THE UNIVERSITY OF QUEENSLAND

AUSTRALIA

Advanced Query Representation and Feedback Methods for Neural Information Retrieval

Hang Li

Bachelor of Science in Computer Science

0000-0002-5317-7227

A thesis submitted for the degree of Doctor of Philosophy at

The University of Queensland in 2025

School of Electrical Engineering and Computer Science

Abstract

Modern information retrieval systems rely on ranking models to determine the relevance of documents to user queries. Traditional models, often grounded in bag-of-words (BOW) representations, provide computational efficiency but are limited in their capacity to capture deeper semantic relationships. Recent developments in neural information retrieval, particularly dense retrievers and large language models (LLMs), have significantly advanced the ability to estimate relevance, leading to improved retrieval quality. However, these improvements frequently introduce new challenges, including increased inference latency, computational overhead, and reduced robustness to noisy or ambiguous inputs. This thesis explores Pseudo-Relevance Feedback (PRF) as a mechanism to enhance retrieval performance while addressing these trade-offs, focusing on three core aspects: effectiveness, efficiency, and robustness.

The first part of this thesis begins by applying a classical Text-Based PRF approach on top of a BERT-based reranker, demonstrating that naively appending feedback passages can yield modest gains but also exposes limitations in scalability and generalizability. We then investigate the reproducibility and effectiveness of ANCE-PRF, an existing dense feedback method built upon training a new query encoder. This analysis reveals both the potential and fragility of vector-based feedback when applied in neural settings. Building on these insights, we propose Vector-Based PRF (VPRF), a lightweight and model-agnostic method that interpolates the original query vector with those derived from top-ranked feedback passages. VPRF significantly improves retrieval effectiveness across a range of dense retrievers and datasets, without requiring retriever-specific training or incurring additional query-time complexity. Further extensions show that VPRF generalizes well to hybrid reranking pipelines and achieves competitive performance relative to more computationally demanding feedback methods.

The second part of this thesis explores the reliability of feedback signals and the practical limitations of existing PRF methods. We begin by analyzing how different feedback quality can affect PRF performance with different dense retrievers. Additionally, our experience with VPRF reveals the need for manual tuning to balance query and feedback vectors, while findings from ANCE-PRF highlight the inefficiency of large models in resource-constrained settings. Motivated by these challenges, we propose Transformer-based Pseudo-Relevance Feedback (TPRF), a lightweight feedback encoder that learns to dynamically adjust the contribution of each signal. By employing compact transformers, TPRF achieves high effectiveness with minimal overhead. To support broader adoption and systematic evaluation, we also introduce a modular PRF framework designed to simplify integration, experimentation, and reproducibility.

The third part investigates PRF in the context of LLM-based dense retrieval. We begin by applying VPRF to state-of-the-art LLM dense retrievers to assess its generalisability. While effectiveness gains are still observed, the improvements diminish compared to earlier settings, highlighting the need for feedback mechanisms better aligned with the generative nature of LLMs. To address this, we introduce Prompt-based Pseudo-Relevance Feedback (PromptPRF), a novel PRF method that leverages LLMs to extract query-independent features from feedback passages and incorporate them into enriched

prompts. PromptPRF enables zero-shot query refinement through in-context learning and operates without retriever-specific tuning. Our experiments demonstrate that PromptPRF substantially improves retrieval effectiveness, particularly for smaller LLMs, bridging the gap with larger models while maintaining computational efficiency.

The fourth part of this thesis presents a set of case studies that examine the failure cases and practical limitations of the proposed PRF methods. Through these case studies, we highlight the trade-offs inherent in PRF design and provide diagnostic insights into when and why specific approaches may underperform. The thesis concludes with a synthesis of the main findings and contributions, along with a discussion of future research directions to advance the development of more adaptive and generalizable feedback methods.

In summary, this thesis advances the understanding and application of PRF in neural information retrieval by proposing several novel approaches and systematically evaluating their effectiveness, efficiency, and robustness. The findings demonstrate that PRF still remains a viable mechanism for enhancing retrieval performance across modern retrieval architectures.

Declaration by author

This thesis is composed of my original work, and contains no material previously published or written by another person except where due reference has been made in the text. I have clearly stated the contribution by others to jointly-authored works that I have included in my thesis.

I have clearly stated the contribution of others to my thesis as a whole, including statistical assistance, survey design, data analysis, significant technical procedures, professional editorial advice, financial support and any other original research work used or reported in my thesis. The content of my thesis is the result of work I have carried out since the commencement of my higher degree by research candidature and does not include a substantial part of work that has been submitted to qualify for the award of any other degree or diploma in any university or other tertiary institution. I have clearly stated which parts of my thesis, if any, have been submitted to qualify for another award.

I acknowledge that an electronic copy of my thesis must be lodged with the University Library and, subject to the policy and procedures of The University of Queensland, the thesis be made available for research and study in accordance with the Copyright Act 1968 unless a period of embargo has been approved by the Dean of the Graduate School.

I acknowledge that copyright of all material contained in my thesis resides with the copyright holder(s) of that material. Where appropriate I have obtained copyright permission from the copyright holder to reproduce material in this thesis and have sought permission from co-authors for any jointly authored works included in the thesis.

Publications included in this thesis

1. [102] **Hang Li**, Shengyao Zhuang, Ahmed Mourad, Xueguang Ma, Jimmy Lin and Guido Zuccon, Improving Query Representations for Dense Retrieval with Pseudo Relevance Feedback: A Reproducibility Study, *In Proceedings of the 44th European Conference on Information Retrieval (ECIR)*, pp. 599–612, 2022

2. [100] **Hang Li**, Shuai Wang, Shengyao Zhuang, Ahmed Mourad, Xueguang Ma, Jimmy Lin and Guido Zuccon, To Interpolate or Not to Interpolate: PRF, Dense and Sparse Retrievers, *In Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 2495–2500, 2022

3. [98] **Hang Li**, Ahmed Mourad, Bevan Koopman and Guido Zuccon, How Does Feedback Signal Quality Impact Effectiveness of Pseudo Relevance Feedback for Passage Retrieval, *In Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 2154–2158, 2022

4. [101] **Hang Li**, Shengyao Zhuang, Xueguang Ma, Jimmy Lin and Guido Zuccon, Pseudo-Relevance Feedback with Dense Retrievers in Pyserini, *In Proceedings of the 26th Australasian Document Computing Symposium (ADCS)*, pp. 1–6, 2022

5. [104] **Hang Li**, Ahmed Mourad, Shengyao Zhuang, Bevan Koopman and Guido Zuccon, Pseudo-Relevance Feedback with Deep Language Models and Dense Retrievers: Successes and Pitfalls, *In ACM Transactions on Information Systems (TOIS)*, pp. 1–40, 2023

6. [107] **Hang Li**, Shengyao Zhuang, Bevan Koopman and Guido Zuccon, LLM-VPRF: Large Language Model Based Vector Pseudo Relevance Feedback, *In arXiv:2504.01448*, `https://arxiv.org/abs/2504.01448`, 2024

7. [105] **Hang Li**, Chuting Yu, Ahmed Mourad, Bevan Koopman and Guido Zuccon, TPRF: A Transformer-based Pseudo-Relevance Feedback Model for Efficient and Effective Retrieval, *In arXiv:2401.13509*, `https://arxiv.org/abs/2401.13509`, 2025

8. [106] **Hang Li**, Xiao Wang, Bevan Koopman and Guido Zuccon, Pseudo-Relevance Feedback Can Improve Zero-Shot LLM-Based Dense Retrieval, *In arXiv:2503.14887*, `https://arxiv.org/abs/2503.14887`, 2025

Submitted manuscripts included in this thesis

No manuscripts submitted for publication.

Other publications during candidature

1. [247] Shengyao Zhuang, **Hang Li**, Shuai Wang and Guido Zuccon, IELAB at TREC Deep Learning Track 2021, *In Proceedings of the 2021 TREC Deep Learning Track*, 2020

2. [96] **Hang Li**, Harrisen Scells, Guido Zuccon, Systematic Review Automation Tools for End-to-End Query Formulation, *In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 2141–2144, 2020

3. [29] Sebastian Cross, **Hang Li**, Shengyao Zhuang, Ahmed Mourad, Bevan Koopman and Guido Zuccon, IELAB for TREC Conversational Assistance Track (CAsT) 2020, *In Proceedings of the 2020 TREC Conversational Assistance Track (CAsT) 2020*

4. [248] Shengyao Zhuang*, **Hang Li*** and Guido Zuccon, Deep Query Likelihood Model for Information Retrieval, *In Proceedings of the 43rd European Conference on Information Retrieval (ECIR)*, pp. 463–470, 2021

5. [231] Zhijun Yang, Yang Wang, Jianhou Gan, **Hang Li** and Ning Lei, Design and Research of Intelligent Question-Answering (Q&A) System Based on High School Course Knowledge Graph, *In Mobile Networks and Applications*, pp. 1–7, 2021

6. [210] Shuai Wang, **Hang Li**, Harrisen Scells, Daniel Locke and Guido Zuccon, MeSH Term Suggestion for Systematic Review Literature Search, *In Proceedings of the 25th Australasian Document Computing Symposium (ADCS)*, pp. 1–8, 2021

7. [249] Shengyao Zhuang, **Hang Li** and Guido Zuccon, Implicit Feedback for Dense Passage Retrieval: A Counterfactual Approach, *In Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 18–28, 2022

8. [97] **Hang Li**, Guido Zuccon, Bevan Koopman and Ahmed Mourad, Agvaluate: A New Test Collection for Both Passage and Document Retrieval in the Agriculture Domain, `https://rdm.uq.edu.au/files/6f3741d0-e541-11ec-8b7d-ffdfa0f38746`, `doi:10.48610/0160dc7`, 2022

9. [212] Shuai Wang, **Hang Li** and Guido Zuccon, MeSH Suggester: A Library and System for MeSH Term Suggestion for Systematic Review Boolean Query Construction, *In Proceedings of the 16th ACM International Conference on Web Search and Data Mining (WSDM)*, pp. 1176–1179, 2023

10. [103] **Hang Li**, Bevan Koopman, Ahmed Mourad and Guido Zuccon, AgAsk: A Conversational Search Agent for Answering Agricultural Questions, *In Proceedings of the 16th ACM International Conference on Web Search and Data Mining (WSDM)*, pp. 1140–1143, 2023

11. [86] Bevan Koopman, Ahmed Mourad, **Hang Li**, Anton van der Vegt, Shengyao Zhuang, Simon Gibson, Yash Dang, David Lawrence and Guido Zuccon, AgAsk: An Agent to Help Answer

Farmer's Questions From Scientific Documents, *In International Journal on Digital Libraries*, 25(4), pp. 569–584, 2024.

Contributions by others to the thesis

This thesis includes several published works co-authored with other researchers. In accordance with the University of Queensland's authorship policy (PPL 4.20.04), I provide below a detailed account of my individual contributions to each of these publications.

For all publications listed in the front matter and included in this thesis, for which I am the first author, I was primarily responsible for the conceptual design of the studies, development of methods, implementation of experiments, data analysis, and manuscript drafting. In addition to these contributions, I also led the submission and revision processes for each manuscript.

My co-authors contributed in the following capacities:

- **Ahmed Mourad**, **Bevan Koopman**, and **Guido Zuccon** provided supervision, conceptual guidance, and critical feedback on manuscript revisions for all included publications.
- **Shuai Wang**, **Shengyao Zhuang**, **Xiao Wang**, **Chuting Yu**, **Xueguang Ma**, and **Jimmy Lin** contributed to selected publications through discussions on methodology, assistance with experiment implementation, evaluation strategies, or by providing editorial feedback on the manuscripts.

I affirm that the work presented in each publication reflects my original contributions. I accept full responsibility for the content of the versions included in this thesis.

Statement of parts of the thesis submitted to qualify for the award of another degree

No works submitted towards another degree have been included in this thesis.

Research involving human or animal subjects

No animal or human subjects were involved in this research.

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my principal supervisor, Prof. Guido Zuccon, for your unwavering support, intellectual generosity, and steadfast guidance throughout this PhD journey. Your high standards, critical insights, and passion for research have profoundly shaped my scholarly thinking and academic discipline. But more than that, I am especially grateful for the care and compassion you showed during my most difficult times. In moments of personal and professional struggle, your support was a steady anchor. You did not just mentor my research, you helped me stay grounded and resilient. Thank you for believing in me even when I found it hard to believe in myself.

I am also deeply indebted to Dr. Ahmed Mourad, who has been far more than a collaborator or co-supervisor. Ahmed, your presence during my hardest moments meant more than words can express. Whether through your generous academic guidance or the everyday gestures, joining me for exercise, swimming, and hiking, especially those unforgettable treks in Cairns. You lifted my spirits when I needed it the most. Your friendship, patience, and genuine care were crucial to helping me move forward. I am proud to call you not only a colleague but a lifelong friend.

I also extend sincere thanks to A/Prof. Bevan Koopman, Dr. Joel Mackenzie, and Dr. Teerapong Leelanupab for your valuable collaboration, thoughtful feedback, and support throughout various stages of my research. Each of you contributed meaningfully to the development and refinement of this thesis.

To my colleagues at the IElab, Shengyao Zhuang, Shuai Wang, Ismail Sabei, Shuyi Wang, Ekaterina Khramtsova, Bing Liu, Suresh Pokharel, Sebastian Cross, Linh Le, Jonathan J Ross, Daniel Locke, Harrisen Scells, Xinyu Mao, and Sitthichoke Subpaiboonkit, thank you for being an integral part of this journey. The stimulating discussions, the shared frustrations and breakthroughs, and the sense of camaraderie made my time at the lab intellectually fulfilling and personally meaningful. Special thanks to Ismail, and Shuai, who were always there when I needed help, I am sincerely grateful for your support and kindness. A heartfelt thanks also goes to Shengyao, whose collaboration, dedication, and friendship were invaluable and irreplaceable to me throughout this journey.

I would also like to acknowledge my peers in the broader IR research community, Xueguang Ma, Luyu Gao, Xiao Wang, and Jimmy Lin. While we may not have collaborated on papers directly, your work, your toolkits, and our discussions have significantly influenced the direction and progress of my research.

To everyone in my family, especially my parents and grandparents, thank you for your endless love, patience, and encouragement throughout this long and challenging path. Your belief in me gave me strength when I felt lost, thank you for your consistent support and the warmth of family that never faded, no matter the distance.

To my love, Chuting Yu, thank you for being my greatest source of strength, joy, and comfort. Your belief in me, even when I doubted myself, gave me the courage to persist through the darkest and most uncertain moments. Your patience, your care, and your presence brought light to the many long

days and nights. You have shared in every struggle and triumph of this journey, and I could not have done this without you. This achievement is as much yours as it is mine.

Finally, to everyone, mentors, peers, reviewers, and friends, who has supported my work, challenged my thinking, or stood by me through unseen ways: thank you. This thesis is not only a product of academic labor but a reflection of the compassion, wisdom, and generosity of the people around me.

Financial support

This research was supported by an Australian Government Research Training Program Scholarship. In addition, this research was also financially supported by the Grain Research and Development Corporation project AgAsk (UOQ2003- 009RTX).

Keywords

Information Retrieval (IR), Neural Information Retrieval (Neural IR), Pseudo-Relevance Feedback (PRF), Query Reformulation, Query Representation, Retrieval Effectiveness, Dense Retrieval, Transformer-Based Retrieval, Encoder-based Dense Retriever, Decoder-based Large Language Model (LLM), Zero-Shot Retrieval, LLM-Based Retrieval, Text-Based PRF, Vector-Based PRF (VPRF), TPRF (Transformer-based PRF), Prompt-Based PRF (PromptPRF)

Australian and New Zealand Standard Research Classifications (ANZSRC)

ANZSRC code: 460508, Information Retrieval and Web Search, 100%

Fields of Research (FoR) Classification

FoR code: 0807, Library and Information Studies, 100%

To my family, for their endless support and love.
To my friends, for keeping me (mostly) sane.
To Monster Energy Drink, for being there when no one else was.
And to curiosity – may it always outrun exhaustion.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Search engines retrieve the most relevant information from a vast document corpus in response to a user's query. To achieve this, search engines rely on algorithms to identify (retrieve) and order (rank) documents based on their estimated relevance.

One of the most common IR tasks is ad-hoc document retrieval [1], where a user submits a short query and the system returns a ranked list of documents. While modern ranking models [2] have made considerable progress in estimating document relevance, their effectiveness is often limited by the quality and specificity of the initial query. In many real-world scenarios, queries are vague, underspecified, or incomplete, resulting in suboptimal retrieval performance. Improving query representation in such settings is critical for accurately capturing the user's information need and retrieving more relevant results.

Pseudo-Relevance Feedback (PRF) has long been employed as a technique to enhance query representations. Instead of relying on explicit user feedback, PRF assumes that the top-ranked documents from an initial retrieval round are likely to be relevant and can therefore serve as a source of expansion or refinement signals. By extracting informative terms or representations from these documents and incorporating them back into a revised query, PRF enables iterative improvements in retrieval effectiveness. Classical PRF methods have demonstrated consistent gains across benchmarks and tasks in traditional IR systems [4, 122, 125, 127, 225, 237, 241].

Despite their historical success, classical PRF methods such as Rocchio [179], RM3 [91], and DFR-based approaches [4, 81, 204] exhibit several limitations when applied on top of term-matching sparse retrievers like BM25 [177] or TF-IDF [182]. These methods rely heavily on exact lexical overlap between query and document terms, which restricts their ability to capture deeper semantic relationships [58, 123, 147]. Consequently, their performance degrades in the presence of *vocabulary mismatch* or when relevant documents use different terminology from the original query [47]. Moreover, feedback term selection is typically based on simple term frequency or co-occurrence statistics, which can be noisy or biased toward dominant but uninformative terms. Techniques like RM3 attempt to

[1] "Passage retrieval" is a related retrieval task in which passages are typically shorter than full documents. In this thesis, we focus on passage retrieval and use the terms "passage" and "document" interchangeably.

[2] Also referred to as "rankers".

mitigate this by estimating term distributions, but they remain sensitive to hyperparameters such as feedback depth, number of expansion terms, and interpolation weights, which often require dataset-specific tuning [125, 237]. Furthermore, these approaches generally lack mechanisms for incorporating contextual information or word dependencies, limiting their effectiveness for complex or ambiguous queries [113]. As a result, while classical PRF methods often offer a principled foundation for query expansion, their reliance on surface-level lexical statistics constrains their effectiveness in modern retrieval scenarios.

1.1 Revisiting Pseudo-Relevance Feedback for Neural Information Retrieval

The landscape of information retrieval has undergone a substantial transformation with the advent of encoder-style neural rerankers [131, 155], encoder-style dense retrievers [51, 73, 83, 223, 238], and more recently, decoder-style large language model (LLMs) based retrievers [128, 250]. These neural information retrieval models rely on high-dimensional vector representations that capture semantic meaning beyond surface-level term overlap, enabling more flexible and expressive matching capabilities. However, this shift in architecture and retrieval mechanism introduces new challenges for incorporating feedback methods. PRF, while well-studied in traditional bag-of-words and probabilistic systems [15, 91, 179, 224, 237], was originally designed for sparse lexical retrieval, and its applicability in neural settings remains underexplored. The assumptions, input formats, and computational characteristics of neural models differ significantly from those of classical systems, raising questions about whether existing PRF methods can be effectively transferred or require fundamental rethinking. This thesis re-examines the design, effectiveness, and limitations of PRF within modern neural information retrieval architectures, with a focus on understanding how these methods must be adapted to align with the evolving representational and operational characteristics of neural pipelines.

1.1.1 Evaluating PRF Adaptation in Transformer-Based Rerankers

To investigate the viability of PRF in neural retrieval, we first explored a straightforward adaptation of classic Text-based PRF technique, which directly reformulates the query text, in the context of encoder-based rerankers, such as BERT [155]. Empirical results indicate that Text-based PRF remains effective in enhancing retrieval quality, even within modern neural pipelines. This observation affirms the broader utility of feedback-driven query refinement beyond traditional term-based systems.

However, several limitations emerge when adapting Text-based PRF to reranking architectures. The constrained input capacity of the underlying rerankers [49, 155] restricts the number of feedback passages that can be accommodated, thereby capping the achievable PRF depth. Additionally, the computational demands of processing long PRF queries result in substantial latency increase [85], which poses challenges for practical deployment. Beyond efficiency concerns, the structural design of

rerankers inherently limits the scope of PRF: as they operate only on a fixed candidate set from the initial retrieval stage, they lack the ability to introduce new relevant documents into the ranked list.

Moreover, we observe that the improvements gained through this integration are often modest, potentially due to a misalignment between model training, typically based on short queries, and inference-time usage involving longer PRF inputs. These challenges highlight the need to reimagine how PRF is integrated into neural architectures, and point toward dense retrievers as a more suitable foundation for feedback mechanisms. Unlike rerankers, encoder-based dense retrieval models support efficient retrieval over large collections and offer greater flexibility in incorporating feedback, presenting an opportunity to achieve both effectiveness and scalability.

1.1.2 A Reproduction Study of Feedback-Aware Query Encoders

Building on the limitations identified by adapting PRF on top of rerankers, we turn to the emerging more effective and efficient encoder-style dense retrievers, where a prominent line of work involves training dedicated query encoders to incorporate feedback signals on top of these dense retrievers.

To better understand the potential and constraints of this approach, we reproduced a representative trained PRF method, namely ANCE-PRF [234], encompassing both its training regime and inference pipeline. Our findings show that learning a specialized query encoder for PRF queries substantially improves retrieval effectiveness. By aligning the model's training with the PRF-augmented queries [102, 234], this ANCE-PRF strategy mitigates the inference-time mismatch observed in untrained PRF approach on top of reranker.

However, despite these improvements, core challenges still persist. The method remains subject to input length constraints, limiting the depth of feedback that can be utilized. Furthermore, while encoder-style dense retrievers are more efficient than rerankers, the requirement for a second round of query encoding introduces additional latency, albeit at a reduced cost. Importantly, we observe that this method suffers from poor generalization across different dense retrievers, often showing strong dependency on the underlying retrieval model. Performance also proves highly sensitive to training configurations, where small variations in hyperparameters can lead to significant degradation in results. These observations suggest that while trained PRF approach on top of dense retrievers offers promising gains, it lacks robustness and scalability for broader application.

1.1.3 Efficient PRF through Vector Manipulation in Dense Retrieval

Motivated by the structure of dense retrievers, which encode queries and passages into a shared vector space, we explored whether query representations could be directly refined through vector-space manipulation, inspired by classical Rocchio-style PRF approach [179]. This direction offers a compelling alternative to both reranking-based and training-intensive PRF methods by sidestepping their core limitations.

Rather than using long textual inputs or retraining dedicated encoders, we provide a vector-space manipulation approach that directly modifies the original query vector representation, which we

name Vector-based PRF (VPRF). This approach enables efficient and scalable feedback integration. By operating entirely in latent space, VPRF eliminates the input length constraints that previously restricted PRF depth, allowing deeper aggregation of feedback without incurring model-specific input limitations. The refined query vectors preserve the original embedding dimensionality, avoiding any training-inference mismatch, and can be used directly in a second retrieval stage with minimal latency, relying solely on fast similarity computations. Importantly, the VPRF approach removes the need for a second round of query encoding, further improving its efficiency.

VPRF is functioning as a general-purpose enhancement that is both lightweight and robust. Its zero-shot, model-agnostic nature allows it to be seamlessly integrated into diverse retrieval systems without retraining, offering a scalable and practical solution for modern neural IR pipelines. Furthermore, we integrate the VPRF approach into a widely adopted neural retrieval toolkit, aiming to provide a free performance boost for all dense retrievers supported within the framework. The implementation is designed to be flexible and extensible, allowing researchers and practitioners to apply the method with minimal overhead across different retrieval backbones.

Despite these advantages, the VPRF approach is not without limitations. First, it requires careful tuning of hyperparameters that control the combination of the original query vector with feedback passage vectors. Although this tuning is significantly cheaper than training new query encoders, it nonetheless introduces a layer of complexity. Second, the quality of feedback is fundamentally constrained by the effectiveness of the underlying dense retriever. While dense retrievers perform well in ranking highly relevant passages, they often struggle to differentiate fine-grained relevance, especially in the presence of ambiguous or low-relevance passages.

As a result, VPRF performance is closely tied to the retriever's ability to identify high-quality feedback. Stronger retrievers tend to yield more informative and semantically aligned feedback passages, leading to effective query refinement, whereas weaker retrievers often provide noisy or off-topic feedback, increasing the risk of query drift. These observations highlight the importance of retriever quality in determining the reliability and effectiveness of VPRF approaches.

1.1.4 Interpolating Sparse and Dense Signals for Enhanced PRF

To further investigate the dense retriever limitations, we turn our attention to interpolation strategies between sparse and dense retrievers. While dense retrievers perform well at ranking highly relevant passages, they often exhibit limitations in fine-grained relevance discrimination, particularly when handling ambiguous or marginally relevant passages [51, 145, 200, 211]. In contrast, traditional sparse retrieval methods such as BM25, offer high recall and remain widely adopted in initial retrieval stages due to their efficiency and robustness. These complementary characteristics suggest an opportunity for integration [113]. Prior works [111, 115, 211] have shown that interpolating the results of dense and sparse retrievers can yield more balanced performance, combining the precision of dense models with the recall strength of sparse counterparts.

Building on this insight, we explore whether such interpolation strategies can also enhance performance within different PRF pipelines. Specifically, we investigate when and how interpolation should be applied in the context of PRF, and whether particular sparse retrievers provide more effective complementary signals. This line of inquiry is motivated by the hypothesis that sparse retrievers may retrieve relevant passages missed by dense models, thus enriching the feedback pool and improving downstream query refinement. By systematically evaluating interpolation across architectures and PRF variants, we aim to identify principled strategies for combining heterogeneous signals in modern retrieval systems.

1.1.5 Understanding the Impact of Feedback Signal Quality on PRF

Across the previous investigations, one recurring observation is the critical role of feedback signal quality in determining the effectiveness of PRF methods. Regardless of the retrieval architecture or feedback mechanism, the informativeness, relevance, and coherence of the feedback passages substantially influence the quality of the refined query and the final retrieval results. However, the specific impact of varying signal qualities across different types of PRF pipelines, ranging from zero-shot approaches to fully trained feedback-aware encoders, remains insufficiently understood. While prior analyses highlight the general importance of high-quality feedback, they do not fully characterize how different pipelines respond to degraded, noisy, or overly general feedback signals.

To address this gap, we undertake a systematic examination of how feedback signal quality affects the behavior and performance of diverse PRF approaches. This includes evaluating robustness under controlled variations in feedback signals, as well as identifying which architectures are more resilient to noisy signals. By disentangling the relationship between signal quality and PRF effectiveness, we aim to develop a deeper understanding of the sensitivity and limitations of the feedback mechanisms, and to inform future designs that can better adapt to varying feedback conditions.

1.1.6 A Lightweight Vector-Space Adapter for Neural PRF

Previously, we have explored a PRF strategy that directly modifies the query representation by aggregating it with feedback passage vectors. While this method is efficient and model-agnostic, it relies on manually tuned hyperparameters to control the weighting between the original query and the feedback signals. These parameters, though lightweight compared to training full models, require careful tuning to avoid query drift and to ensure effective signal integration. On the other hand, trained PRF approaches operate in the text domain, constructing PRF-refined query vectors that are processed by a newly trained query encoder. While these models mitigate training-inference mismatch and achieve strong performance under certain conditions, they remain constrained by the input limits of the underlying dense retrievers. Moreover, the learned query encoders tend to be large and inflexible, often failing to generalize across retriever architectures even when trained with the same procedure.

To overcome these limitations, we propose a lightweight transformer-based feedback adapter TPRF that also operates entirely in the vector space. Instead of relying on manually specified interpolation

weights, TPRF learns to automatically modulate the contribution of each feedback vector, promoting relevant signals while suppressing noise. This design eliminates the need for hyperparameter tuning, it also supports broad compatibility with different dense retriever backbones via fine-tuning, as it manipulates embeddings directly. By taking vector matrices as input, TPRF bypasses the token length constraints of text-based encoders, enabling deeper feedback integration without compromising efficiency or scalability. This TPRF model strikes a balance between the adaptability of trained models and the efficiency of latent-space manipulation, offering a flexible and robust solution for neural PRF.

1.1.7 Transferring Vector-based PRF to Decoder-Style LLM Retrievers

Decoder-style generative LLMs have further reshaped natural language processing by achieving strong performance across a wide range of tasks. Their ability to capture rich semantic information has also made them appealing for information retrieval. Recent decoder-style dense retrievers, built on LLMs, show strong generalization and semantic matching capabilities [128, 250]. Despite their generative design, these retrievers still encode queries and passages into a shared vector space for similarity-based retrieval.

Build upon this similarities, we investigate whether VPRF techniques developed for encoder-based retrievers can be effectively transferred to decoder-style LLM-based dense retrievers, which we name LLM-VPRF. Initial results suggest that while the VPRF mechanism can still be applied, the resulting performance of LLM-VPRF gains are relatively modest. Unlike encoder-based retrievers, where vector aggregation can significantly enrich under-specified queries, decoder-based embeddings already encode semantically rich representations. Therefore, LLM-VPRF, which utilises simple vector aggregation with feedback signals, appears to be insufficient to offer substantial improvements. Moreover, LLM-VPRF does not leverage the generative capabilities of the underlying LLMs, which may be essential to fully exploit their potential in feedback-driven retrieval scenarios. These observations point to a need for more sophisticated PRF strategies that are better aligned with the representational strengths of generative models.

1.1.8 Generative Pseudo-Relevance Feedback with Large Language Models

To more fully leverage the generative capabilities of large language models, we draw inspiration from prior work [139] that explores generating features as a form of feedback signal. These methods demonstrate that LLMs can synthesize rich semantic information to support retrieval, but typically rely on query-dependent feature generation performed at inference time. While effective, this approach introduces substantial latency and computational cost, making it impractical for resource constrained environments. Moreover, in many retrieval settings, model scaling alone can improve performance; however, relying on increasingly large models is often infeasible due to prohibitive inference costs and significant infrastructure demands.

To address these challenges, we propose a novel Prompt-based Pseudo-Relevance Feedback framework PromptPRF that eliminates the need for expensive online feature generation. Instead of

producing query-specific feedback at runtime, PromptPRF pre-computes query-independent features from passages offline at indexing time, allowing efficient reuse across queries. This strategy enables the use of smaller language models in a retrieval pipeline while still achieving competitive performance. By explicitly incorporating rank information into the prompt, PromptPRF guides the LLM to assign appropriate weights to each feedback signal, enhancing its ability to differentiate useful passages from noise. Empirical results show that PromptPRF is more efficient than prior methods and narrows the performance gap between small and large models in zero-shot settings.

Nonetheless, certain limitations still persist. Decoder-based LLMs can also struggle with ambiguous or under-specified queries, such as open-ended questions, often failing to generate coherent or useful features. The inclusion of rank information proves critical in helping LLMs assess the relative importance of each passage, yet the model remains susceptible to misleading or noisy feedback. When exposed to inconsistent signals, PromptPRF may induce query drift, ultimately degrading retrieval quality. These findings highlight both the promise and the current boundaries of generative PRF strategies, motivating future work on improving signal reliability and model robustness.

1.2 Research Questions and Thesis Structure

The central objective of this thesis is to investigate *how Pseudo-Relevance Feedback (PRF), a long-standing and widely studied technique in information retrieval, can be effectively applied within modern neural retrieval architectures.* While PRF has traditionally relied on lexical signals [91] and statistical models [1], the shift toward modern neural information retrieval models introduces both new opportunities for feedback integration and fundamental challenges in system design, efficiency, and robustness. To systematically explore these dimensions, we structure the thesis around three high-level research questions, each addressing a core aspect of PRF in neural retrieval: (1) integration with neural models, (2) practicality and extensibility in deployment, and (3) adaptability to decoder-style large language models.

1.2.1 RQ1: How can Pseudo-Relevance Feedback be integrated into neural models?

While classical PRF methods have often shown effectiveness in sparse, lexical-based systems [15, 18, 91, 125, 179, 237], their application to neural models, such as encoder-style rerankers [155] and encoder-style dense retrievers [73, 223, 238], is under explored. This research question investigates the feasibility and limitations of incorporating PRF into neural IR systems, with an emphasis on understanding how classic and novel feedback strategies perform under neural representations, identifying integration trade-offs, and evaluating the impact of underlying retrieval components. Part 1 of this thesis (Chapters 3–6) systematically explores these directions, covering adaptations of text-based PRF to neural rerankers, latent-space PRF methods for dense retrievers, and hybrid approaches that

interpolate sparse retriever and dense retriever signals. Together, these chapters aim to establish a foundational understanding of how PRF can be effectively reimagined for neural architectures.

1.2.2 RQ2: How to make Pseudo-Relevance Feedback models more practical and extensible?

While PRF methods have demonstrated considerable promise in controlled research settings, deploying them in more practical retrieval systems presents new challenges related to robustness, efficiency, and extensibility. Practical retrieval environments often involve frozen retrievers, limited computational budgets, and noisy or inconsistent feedback signals. To be viable in such contexts, PRF approaches must adapt to varying signal quality, scale efficiently, and remain compatible with evolving retrieval infrastructures. This research question explores how to make PRF methods more practical and extensible by investigating their sensitivity to feedback noise, analyzing trade-offs between effectiveness and computational cost, and developing a modular framework that supports flexible integration with diverse retrievers and feedback strategies. Part 2 of the thesis (Chapters 7–9) focuses on these directions, including empirical studies on signal robustness, the design of lightweight PRF mechanisms with minimal latency overhead, and the implementation of an extensible PRF framework that supports plug-and-play components and cross-model compatibility. These investigations collectively aim to bridge the gap between theoretical effectiveness and practical applicability, enabling PRF to serve as a reliable and scalable enhancement across a wide range of retrieval systems.

1.2.3 RQ3: Are Pseudo-Relevance Feedback models still applicable to decoder-style Large Language Models?

The rise of decoder-style LLMs introduces both new possibilities and open challenges for integrating PRF techniques. Unlike encoder-based retrievers that generate fixed embeddings for similarity-based retrieval, decoder-style LLMs operate through auto-regressive generation, making the incorporation of PRF methods more complex. This research question explores whether established PRF techniques, such as vector manipulation strategies, can be effectively extended to LLM-based dense retrievers, and whether feature-based prompting approaches can leverage feedback signals in a way that is both effective and computationally efficient. In addition, this line of work critically examines the limitations that persist across different PRF methods throughout the thesis. Part 3 and the beginning of Part 4 (Chapters 10–12) investigate these questions in depth. Together, these chapters address RQ3 by evaluating the viability and identifying the constraints of adapting PRF to the evolving class of LLM-based retrieval architectures, as well as identifying the recurring limitations.

1.3 Overview of Thesis Contributions

In this thesis, we propose a set of methods to enhance the performance of neural ranking systems through the systematic integration of Pseudo-Relevance Feedback (PRF). Our work addresses three central challenges in modern information retrieval: improving effectiveness, reducing computational overhead, and ensuring robustness across different retriever architectures and feedback conditions [30, 120, 121, 190, 203]. The proposed methods span encoder-style dense retrievers and decoder-style large language models (LLMs), and include novel query reformulation strategies, feedback signal modeling, and prompt-based retrieval enhancements. These methods are designed to be modular and compatible with existing retrieval pipelines, enabling practical deployment without extensive retraining or manual supervision. Together, the contributions presented in this thesis are supported by a series of published and submitted works, and are organized across multiple retrieval paradigms and system configurations.

Firstly, this thesis begins with the application of simple PRF methods on top of neural rerankers. The experiments show that such integration can improve effectiveness; however, we also identify important limitations. To better understand the landscape of learned PRF, we reproduce and extend a representative method and uncover key gaps. In response to these challenges, we introduce novel Vector-based PRF (VPRF) approaches that can be applied to any dense retriever without retraining. Finally, we study the integration of sparse and dense feedback signals in PRF and analyze how signal interpolation affects retrieval performance. These contributions resulted in the following publications:

- Hang Li, Ahmed Mourad, Shengyao Zhuang, Bevan Koopman and Guido Zuccon, Pseudo-Relevance Feedback with Deep Language Models and Dense Retrievers: Successes and Pitfalls, *In ACM Transactions on Information Systems (TOIS)*, pp. 1–40, 2023
- Hang Li, Shengyao Zhuang, Ahmed Mourad, Xueguang Ma, Jimmy Lin and Guido Zuccon, Improving Query Representations for Dense Retrieval with Pseudo Relevance Feedback: A Reproducibility Study, *In Proceedings of the 44th European Conference on Information Retrieval (ECIR)*, pp. 599–612, 2022
- Hang Li, Shuai Wang, Shengyao Zhuang, Ahmed Mourad, Xueguang Ma, Jimmy Lin and Guido Zuccon, To Interpolate or Not to Interpolate: PRF, Dense and Sparse Retrievers, *In Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 2495–2500, 2022

Furthermore, this thesis investigates how the quality of feedback signals affects the effectiveness of PRF, showing that PRF performance is highly sensitive to noisy feedback signals, and different PRF methods vary in their robustness to such signals. Motivated by the limitations identified earlier, we propose a Transformer-based PRF model (TPRF) that directly learns to extract and weight feedback signals under system constraints. This model achieves strong effectiveness while maintaining low latency and minimal computational cost. To further support research and deployment, we also develop a unified PRF framework within the Pyserini toolkit, enabling reproducible and extensible experimentation across dense retrievers. These contributions are supported by the following publications:

- Hang Li, Ahmed Mourad, Bevan Koopman and Guido Zuccon, How Does Feedback Signal Quality Impact Effectiveness of Pseudo Relevance Feedback for Passage Retrieval, *In Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 2154–2158, 2022

- Hang Li, Chuting Yu, Ahmed Mourad, Bevan Koopman and Guido Zuccon, TPRF: Transformer-based Pseudo-Relevance Feedback Model for Efficient and Effective Retrieval, *In arXiv:2401.13509, https://arxiv.org/abs/2401.13509*, 2025

- Hang Li, Shengyao Zhuang, Xueguang Ma, Jimmy Lin and Guido Zuccon, Pseudo-Relevance Feedback with Dense Retrievers in Pyserini, *In Proceedings of the 26th Australasian Document Computing Symposium (ADCS)*, pp. 1–6, 2022

Finally, this thesis examines the applicability of VPRF in decoder-style LLMs, systematically evaluating whether this PRF technique, which shows promising effectiveness with encoder-style models, still remains effective in LLM-based retrieval scenarios. To better leverage the generative nature of LLMs, we further propose a feature-based and prompt-based PRF approach (PromptPRF) that refines query representations using extracted feedback features rather than raw passages. This method shows strong effectiveness, particularly for smaller LLMs in zero-shot retrieval settings. The relevant contributions include:

- Hang Li, Shengyao Zhuang, Bevan Koopman and Guido Zuccon, LLM-VPRF: Large Language Model Based Vector Pseudo Relevance Feedback, *In arXiv:2504.01448, https://arxiv.org/abs/2504.01448*, 2024

- Hang Li, Xiao Wang, Bevan Koopman and Guido Zuccon, Pseudo-Relevance Feedback Can Improve Zero-Shot LLM-Based Dense Retrieval, *In arXiv:2503.14887, https://arxiv.org/abs/2503.14887*, 2025

By developing and evaluating a series of PRF methods across rerankers, encoder-style dense retrievers, and decoder-style dense retrievers, this thesis advances the effectiveness, efficiency, and robustness of neural retrieval systems. The proposed approaches contribute practical solutions, empirical evidence, and theoretical insights that deepen our understanding of feedback modeling in modern information retrieval. Together, these contributions form a coherent framework for scalable and adaptable PRF integration, and provide a strong foundation for future research in adapting PRF in neural information retrieval.

Chapter 2

Background and Literature Review

This chapter provides foundational background for understanding the retrieval architectures and feedback methods examined throughout this thesis. We begin by defining the ad hoc retrieval setting [143], in which a query q is used to retrieve a ranked list of documents based on an estimated relevance score produced by a scoring function $s(q,d)$, where d represents each document in the collection. The ranking is determined by sorting passages in descending order of their relevance scores. A high-level view of the task is presented in Figure 2.1. A key distinction in modern retrieval systems lies in how queries and documents are represented, leading to two broad categories: sparse and dense retrieval. In sparse retrieval, representations are typically lexical, high-dimensional, and aligned with a fixed vocabulary, as seen in traditional models such as BM25 [177]. In contrast, dense retrieval encodes both queries and documents into continuous, low-dimensional vectors, often learned through neural encoders, allowing for semantic matching beyond exact term overlap.

This chapter is organized as follows. Section 2.1 introduces traditional retrieval models and provides an overview of their mechanisms and limitations. Section 2.2 explores the role of transformer-based

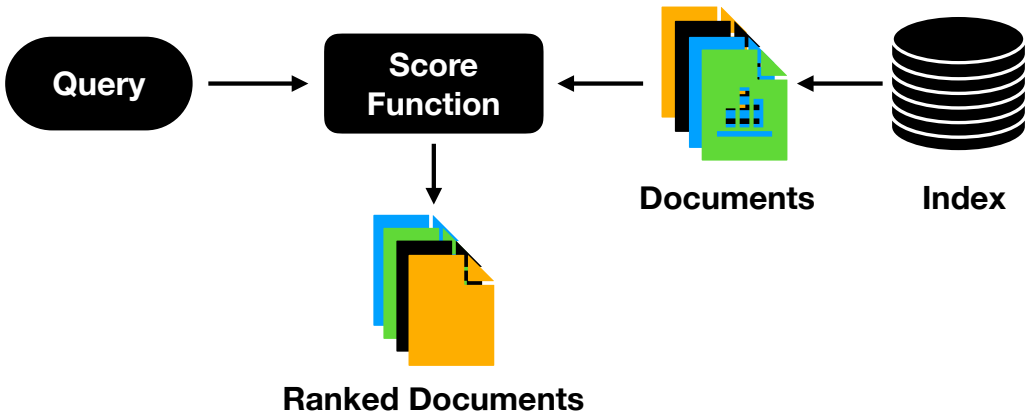


Figure 2.1: A high-level view of ad hoc retrieval task.

neural language models in modern retrieval systems, covering architectural design, multi-stage retrieval pipelines, and techniques for expanding document or improving sparse representations using these models. Section 2.3 focuses on dense retrieval paradigms, distinguishing between single-representation and multi-representation dense retrievers, by discussing their respective design choices and retrieval behaviors. Section 2.4 presents a detailed review of Pseudo-Relevance Feedback (PRF) techniques, categorizing them into sparse PRF and neural PRF approaches, with attention to how these methods are adapted across retrieval architectures.

2.1 Traditional Retrieval Models

In traditional retrieval systems, both queries and documents are represented using high-dimensional sparse vector representations, where each dimension in the vector corresponds to a distinct term in the vocabulary (which is a ordered list consists of all unique terms from the document collection). These vectors typically encode whether or not specific terms from the vocabulary are present in the query/document text and, in some cases, how frequently they occur. The overall dimensionality of these vectors equals the size of the vocabulary, making them inherently sparse since most terms are absent in any given document or query. This binary or weighted encoding, commonly referred to as *one-hot* or *term-frequency-based* representation, allows retrieval models to compute relevance scores based on lexical overlap between query and document terms. In the following section, we introduces widely used traditional retrieval methods.

Vector Space Retrieval Models (VSMs): Vector space models represent both queries and documents as high-dimensional sparse vectors within a shared term space, allowing relevance to be estimated through geometric similarity. A common approach is to use cosine similarity to compute the relevance between the query vector $\mathbf{q}$ and the document vector $\mathbf{d}$:

$$s(q,d) = \text{cosine}(\mathbf{q},\mathbf{d}) = \frac{\mathbf{q}\cdot\mathbf{d}}{\|\mathbf{q}\|\|\mathbf{d}\|} \tag{2.1}$$

where $\mathbf{q}\cdot\mathbf{d}$ is the dot product of the two vectors, and $\|\mathbf{q}\|$ and $\|\mathbf{d}\|$ are the Euclidean norms (i.e., magnitudes) of $\mathbf{q}$ and $\mathbf{d}$, respectively. VSMs became foundational in early information retrieval systems due to their intuitive formulation and computational efficiency [185]. Nevertheless, they exhibit several limitations. The high-dimensional term-based representations do not capture semantic or contextual relationships between terms and rely heavily on exact lexical matches. Additionally, the vocabulary in large-scale collections often contains a significant number of non-informative or low-quality terms, known as *stopwords*, which can degrade retrieval effectiveness. To mitigate this, pre-processing techniques such as stopword removal are commonly applied to reduce vocabulary size and improve efficiency [184, 191, 201].

Term Frequency–Inverse Document Frequency (TF-IDF): Building on the vector space model, TF-IDF [80] introduces a more effective term weighting scheme that accounts for both local and global term importance. Term frequency (TF) reflects how frequently a term appears in a document, while inverse document frequency (IDF) down-weights terms that are common across the entire collection,

thereby emphasizing more informative and discriminative terms. This helps reduce the influence of common or non-informative words in relevance estimation.

The IDF component is defined in Equation (2.2), where N denotes the total number of documents in the collection and $\mathrm{DF}(t)$ is the number of documents containing term t. The additive smoothing ensures stability when $\mathrm{DF}(t) = 0$.

$$\mathrm{IDF}(t,d) = \log\left(\frac{N+1}{\mathrm{DF}(t)+1}\right) \tag{2.2}$$

The final TF-IDF weight for term t in document d is given in Equation (2.3), where the local importance (TF) is scaled by the informativeness of the term across the corpus (IDF):

$$\text{TF-IDF}(t,d) = \mathrm{TF}(t,d) \cdot \mathrm{IDF}(t,d) = \mathrm{TF}(t,d) \cdot \log\left(\frac{N+1}{\mathrm{DF}(t)+1}\right) \tag{2.3}$$

Here, $\mathrm{TF}(t,d)$ refers to the frequency of term t in document d, though it may be normalized by document length or other factors depending on the implementation.

While TF-IDF remains a strong and interpretable baseline, it has notable limitations. It does not incorporate document length normalization, which can lead to biased relevance scores for longer documents. Moreover, its reliance on exact lexical matching limits its robustness in handling vocabulary mismatch and semantic variation, particularly for scenarios involving synonymy, paraphrasing.

Okapi BM25 Retrieval Model: While TF-IDF improves upon the basic vector space model by incorporating global term importance as in Equation (2.3), it does not account for document length or diminishing returns of term frequency. To address these limitations, probabilistic retrieval models were introduced, grounded in the probabilistic relevance framework [176, 177]. Among these, BM25 [178] remains one of the most effective and widely used sparse retrieval models. It builds upon TF-IDF by incorporating non-linear term frequency scaling and document length normalization, improving robustness across documents of varying verbosity.

The BM25 score for a query q and document d is defined in Equation (2.4), where each query term contributes to the overall relevance score based on its frequency in the document and its global distribution across the corpus:

$$\mathrm{BM25}(q,d) = \sum_{t \in q} \mathrm{IDF}(t) \cdot \frac{\mathrm{TF}(t,d) \cdot (k_1+1)}{\mathrm{TF}(t,d) + k_1 \cdot (1 - b + b \cdot \frac{|d|}{\mathrm{avgdl}})} \tag{2.4}$$

Here, $\mathrm{TF}(t,d)$ is the frequency of term t in document d, same as in TF-IDF, $|d|$ is the document length, and `avgdl` is the average document length across the collection. The hyperparameters k_1 and b control the degree of term frequency saturation and the extent of length normalization, respectively. The IDF component in BM25, shown in Equation (2.5), adjusts the term's contribution based on its rarity in the collection:

$$\mathrm{IDF}(t) = \log\left(\frac{N - \mathrm{DF}(t) + 0.5}{\mathrm{DF}(t) + 0.5} + 1\right) \tag{2.5}$$

This formulation ensures that rare terms contribute more to the relevance score than common ones. BM25 is effective because it follows simple and reasonable principles for ranking, such as reducing the

influence of very frequent terms and adjusting for differences in document length, while maintaining computational efficiency. In this thesis, BM25 is employed extensively as a strong sparse retrieval baseline in empirical evaluations throughout the chapters. However, it addresses several limitations from TF-IDF, like other lexical matching models, BM25 remains limited by its reliance on exact term overlap and inability to model semantic similarity or contextual dependencies.

In addition to the classic sparse retrieval models, several other approaches have been proposed to further advance retrieval effectiveness. Notably, information-theoretic methods such as Divergence From Randomness (DFR) [2–4], and probabilistic frameworks like the Query Likelihood Model (QLM) [165, 236], have contributed significantly to the development of information retrieval by introducing alternative ranking principles grounded in statistical modeling and language generation. These models extend the capabilities of traditional term-matching approaches and have demonstrated improved effectiveness across various retrieval tasks.

However, the effectiveness of these models still remains bounded by their reliance on surface-level lexical signals and fixed vocabulary representations. With the advent of deep learning and large-scale pretraining, Pre-Trained Language Models (PLMs) have emerged as a powerful alternative that can capture richer semantic and contextual information, marking a paradigm shift in modern information retrieval.

2.2 Neural Language Models

In this thesis, we focus particularly on transformer-based natural language models. This section begins with an overview of the general transformer architecture, explaining its structure and underlying mechanisms. We then introduce various types of pre-trained language models (PLMs), categorized by their architectural designs, in Section 2.2.1. A clear distinction between PLMs and LLMs can be hard to pin down, and different practices are followed in the literature. For the purpose of this thesis, we generally categorise PLMs as encoder-based or encoder-decoder based models, smaller in parameter size (<500 million parameters) and released with just pretraining steps. PLMs include models like BERT [37], T5 [171], and RoBERTa [119]. In contrast, we generally categorize LLMs as decoder-only, designed for autoregressive generation, larger in parameter size and released following additional post-training (e.g., instruction fine-tuning or reinforcement learning). LLMs include models like GPT-4 [161], QWEN [227] and LLaMA [202]. We acknowledge that this is not a strict criteria and there maybe some overlap. We then discuss the integration of PLMs into information retrieval pipelines, including their use in multi-stage retrieval (Section 2.2.2), document expansion, and learned sparse retrieval (Section 2.2.3).

2.2.1 Transformer-based Neural Language Models

Transformer-based Neural Language Models have emerged as a foundational component in modern Natural Language Processing (NLP), offering state-of-the-art performance across a wide range of

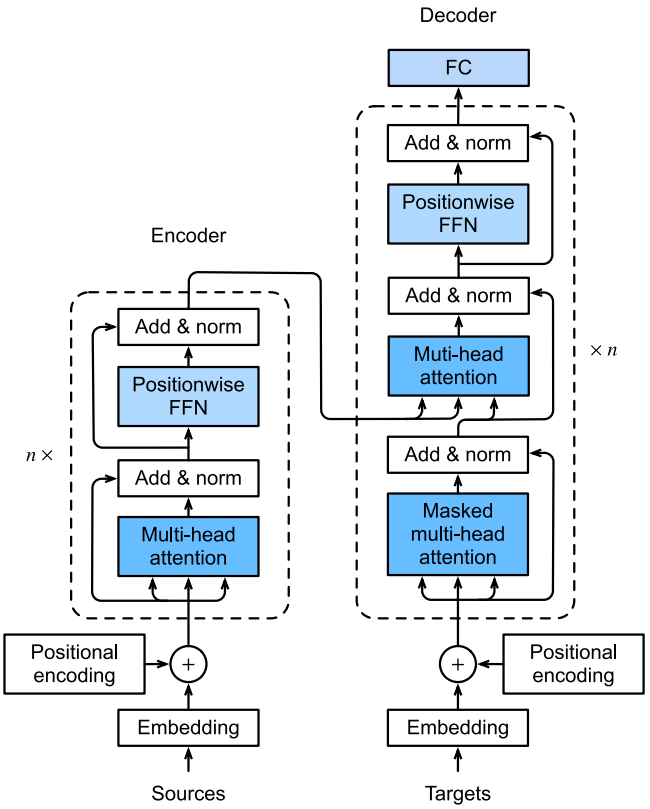


Figure 2.2: Overview of the transformer architecture.

downstream tasks. Built upon the self-attention mechanism introduced in the Transformer architecture [205], these models are capable of capturing complex contextual dependencies and encoding richer semantic representations of text. Unlike earlier recurrent [21, 43, 70] or convolutional [68, 87, 92] architectures, transformer-based Neural Language Models enable parallel computation over entire sequences, leading to significant improvements in scalability and efficiency.

Transformer Architecture

The Transformer architecture, proposed by Vaswani et al. [205], introduced a self-attention mechanism that enables the model to capture dependencies among all tokens in a sequence, irrespective of their positional distance, thus allowing more effective modeling of long-range contextual relationships compared to earlier architectures.

An overview of the Transformer architecture is shown in Figure 2.2. The architecture consists of two main components: an encoder and a decoder, each composed of a stack of identical layers. The encoder transforms the input sequence into a sequence of continuous representations using layers that include multi-head self-attention and position-wise feed-forward networks. Since the architecture does not inherently encode order, which has a significant impact on the output, positional encodings are added to the input embeddings to provide information about the token positions.

The other main component, decoder, generates output sequences by attending to both the previously generated tokens and the encoder outputs. Each decoder layer includes three sub-components: masked multi-head self-attention to prevent positions from attending to future tokens, encoder-decoder attention

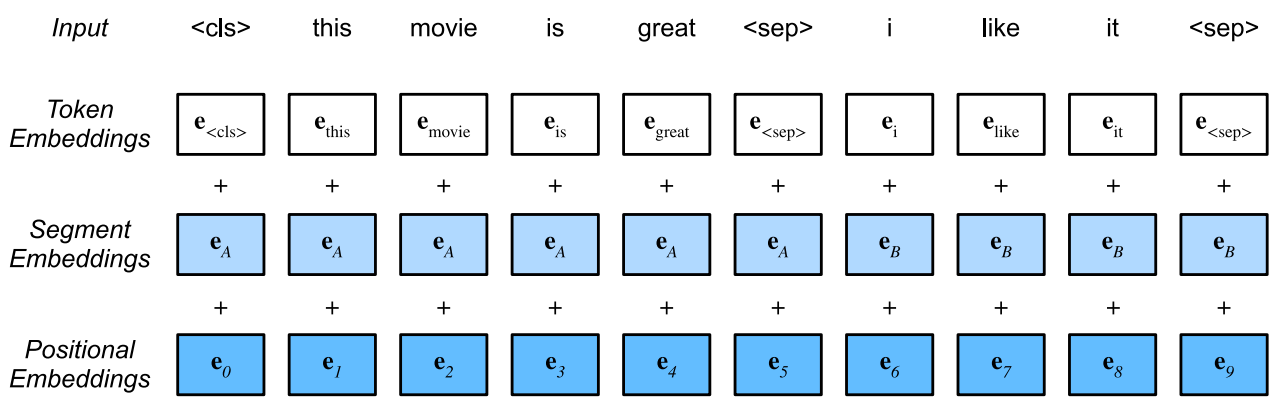


Figure 2.3: The input embedding sequence of BERT model.

to incorporate information from the encoder, and a feed-forward network. This structure enables the decoder to condition each output token on both the source input and the prior outputs.

Compared to earlier sequence-to-sequence models based on recurrent neural networks (RNNs), such as LSTM [70] and GRU [21], the Transformer removes the need for sequential computation by replacing recurrence with self-attention. This change allows the model to process entire sequences in parallel, significantly improving training efficiency. The parallelism of the Transformer architecture aligns well with the computational characteristics of modern GPUs, enabling efficient training over large-scale datasets and facilitating the development of much larger models.

As a result, Transformer-based architectures have become the foundation of most modern pre-trained language models. Their scalability, representational capacity, and compatibility with parallel computation have established the Transformer as the dominant framework in contemporary NLP [34, 37, 59, 113, 170, 171, 232]. The following sections present representative Neural Language Models categorized by their architectural variants.

Encoder-Based PLMs

Encoder-based pre-trained language models are designed to produce deep contextualized representations by encoding entire input sequences through stacked self-attention layers. These models are particularly suited for tasks that require a comprehensive understanding of the input. Among encoder-based PLMs, BERT (Bidirectional Encoder Representations from Transformers) [37] is the most prominent example and has played a foundational role in shaping subsequent research in this space.

The BERT model can be considered as an encoder-only model, which employs a multi-layer bidirectional Transformer encoder that simultaneously attends to both left and right contexts of each token within a sequence. This bidirectional attention mechanism enables the model to capture more comprehensive semantic and syntactic information compared to unidirectional language models. Input sequences are tokenized and then embedded with positional information before being passed through the encoder layers.

To structure the input, BERT introduces two special tokens: [CLS] and [SEP]. The [CLS] token is added to the beginning of every input sequence and serves as an aggregate representation of the entire

sequence. Its final hidden state is typically used as the input to a classification layer for tasks such as sentence classification or next sentence prediction. The [SEP] token, on the other hand, is used to delimit segments within the input. In single-sentence tasks, a single [SEP] is appended at the end of the sequence to mark its boundary. In sentence-pair tasks, a second [SEP] is inserted between the two segments to indicate their separation. This segmentation allows BERT to distinguish between different parts of the input and enables it to model inter-sentence relationships. The overall embedding structure of a BERT input sequence is illustrated in Figure 2.3.

BERT is pre-trained using two self-supervised tasks: masked language modeling (MLM), which helps the model learn contextual word representations by predicting missing tokens, and next sentence prediction (NSP), which trains it to understand relationships between sentences. These objectives enable BERT to capture both fine-grained and broader semantic information. Once pre-trained, BERT can be easily adapted to various downstream NLP tasks through lightweight fine-tuning, without modifying the core architecture.

The success of BERT has led to the development of many enhanced encoder-based PLMs. RoBERTa [119] improves training stability and performance by removing the next sentence prediction (NSP) objective, adopting larger training corpora, and employing dynamic masking during pretraining. ELECTRA [24] introduces a more sample-efficient pretraining approach by replacing MLM with a discriminative objective, in which the model learns to distinguish between real tokens and those generated by a small auxiliary generator.

Other models have focused on architectural and objective refinements. DeBERTa [69] enhances the attention mechanism by disentangling positional and content information, while MPNet [193] integrates permuted language modeling with masked pretraining to better capture dependency information. Additional variants such as XLNet [232], which leverages permutation-based autoregressive objectives, and ALBERT [90], which reduces model size through parameter sharing and factorized embeddings, further contribute to improving representation quality and efficiency across a range of NLP tasks.

Encoder-Decoder Based PLMs

Encoder–decoder based PLMs adopt the full Transformer architecture, consisting of both an encoder and a decoder stack. This design is well-suited for sequence-to-sequence tasks where the input and output differ in length and structure, such as machine translation, summarization, and question answering. The encoder first processes the input sequence into a contextualized intermediate representation, which the decoder then uses to autoregressively generate the target sequence. By leveraging both sides of the Transformer architecture, encoder–decoder models can jointly model understanding and generation.

One of the most prominent encoder–decoder PLMs is T5 (Text-to-Text Transfer Transformer) [171], which reformulates all NLP tasks into a unified text-to-text format. In T5, both the input and the output are treated as textual sequences, allowing the model to handle a wide range of tasks within the same architecture. During pretraining, T5 is trained with a span-corruption objective, in which contiguous spans of text are masked and replaced with a unique sentinel token. The model learns

to generate the masked spans based on the corrupted input, encouraging it to capture both local and global dependencies.

Another influential encoder–decoder model is BART (Bidirectional and Auto-Regressive Transformers) [93], which combines the bidirectional encoding of BERT [37] with the autoregressive decoding of GPT [169]. BART is pretrained using a denoising autoencoder objective, where the input text is first corrupted by one of several noising functions (such as token masking, deletion, or sentence permutation), and the model is trained to reconstruct the original sequence. This approach allows BART to be highly effective in generation tasks, especially in scenarios requiring robust handling of noisy or incomplete inputs.

In contrast to encoder-only models, which produce contextualized representations for classification, encoder–decoder PLMs are inherently more flexible. They can condition generation directly on input content, making them particularly useful in tasks that require a tight coupling between input and output sequences.

Decoder-Based PLMs and LLMs

Decoder-based PLMs and LLMs are built exclusively on the decoder component of the Transformer architecture (Figure 2.2). These models operate in a unidirectional or autoregressive manner, generating each token by conditioning only on previously generated tokens. This architectural constraint enables natural language generation and forms the basis for a wide range of generative tasks.

The most well-known decoder-based models are the Generative Pre-trained Transformer (GPT) series [14, 161, 169, 170]. GPT models are trained using a causal language modeling (CLM) objective, where the model predicts the next token in a sequence given all preceding tokens. This objective is well-aligned with generative applications, as it allows for autoregressive text generation in a left-to-right manner. Input sequences are pre-pended with a start-of-sequence token, and no future tokens are visible during training due to the application of causal masking in the self-attention layers.

GPT adopts the Transformer decoder architecture that omits the encoder–decoder attention layers, as it does not require conditioning on a separate input sequence. Instead, all attention is self-contained within the decoder, constrained by causal masking. Despite this simplification, GPT models have demonstrated remarkable performance in a wide range of zero-shot, few-shot, and fine-tuned settings. GPT-2 [170] highlighted the scalability of decoder-only models for generating coherent long-form text, while GPT-3 [14] further expanded model capacity and demonstrated strong generalization across tasks with minimal task-specific supervision.

Decoder-based LLMs have also been adopted in instruction-tuned and alignment-focused models. Recent variants such as InstructGPT [162] and GPT-4 [161] incorporate human feedback and instruction-following capabilities to align model outputs with user intent. These models use the same decoder-only architecture but introduce additional supervised fine-tuning and reinforcement learning steps to improve alignment and usability in real-world applications.

Unlike encoder-based models, decoder-only PLMs and LLMs are inherently generative and are not naturally suited for bidirectional context modeling. However, their ability to perform open-ended

generation and their compatibility with autoregressive decoding make them highly effective for tasks that require coherent and context-aware language generation. Due to their strong generative capabilities and scalability, decoder-based models continue to be central to the development of general-purpose language models and have played a significant role in shaping current trends in large-scale NLP systems.

2.2.2 Multi-Stage Retrieval Using Neural Language Models

With the advancements of Transformer-based Neural Language Models described in Section 2.2.1, many of these models have demonstrated strong performance across a wide range of downstream tasks, particularly due to their ability to capture deep semantic representations from text. This has motivated researchers to explore their application in IR. However, due to their substantial computational demands and high inference latency, directly applying these models to full-corpus retrieval is often impractical [113, 230].

As a result, one of the earliest and most common applications of neural language models in IR is in the reranking stage, where they are used to reorder a smaller set of candidate documents retrieved by an initial, more efficient first-stage retriever. These initial retrievers are typically traditional sparse retrieval models such as BM25 (see Section 2.1), which efficiently retrieve a subset of potentially relevant documents from the entire corpus. These models, often referred to as rerankers, then computes fine-grained relevance scores over this smaller candidate set to produce a more accurate final ranking.

This architecture forms the basis of the widely adopted multi-stage retrieval pipeline, commonly known as *Retrieve-and-Rerank*. In this setup, an efficient sparse or dense retriever generates an initial candidate pool from the full corpus based on lexical term overlap, and a Transformer-based reranker refines the ranking based on deeper semantic matching. This combination balances efficiency and effectiveness, enabling high retrieval effectiveness while managing resource constraints.

In the following sections, we introduce three major categories of reranking models based on the architectural design of the underlying PLMs: encoder-based, decoder-based, and encoder–decoder based rerankers.

Encoder-Based PLMs for Reranking: One of the earliest and most influential applications of encoder-based PLMs in reranking task was introduced by Nogueira and Cho [155], who employed BERT [37] as a cross-encoder for passage-level relevance estimation. Unlike traditional BERT usage where individual inputs are encoded independently, the cross-encoder formulation processes the query and candidate document jointly as a single input sequence. The input format follows the structure:

$$[[\text{CLS}], q_1, q_2, q_3, ..., q_i, [\text{SEP}], d_1, d_2, d_3, ..., d_i, [\text{SEP}]], \tag{2.6}$$

This setup allows the self-attention mechanism to capture token-level interactions across both segments, enables the model to attend jointly over the query and the document, capturing richer semantic interactions between the two.

The BERT reranker is fine-tuned on labeled relevance data, where each training instance consists of a query, a relevant (positive) document, and one or more non-relevant (negative) documents. The

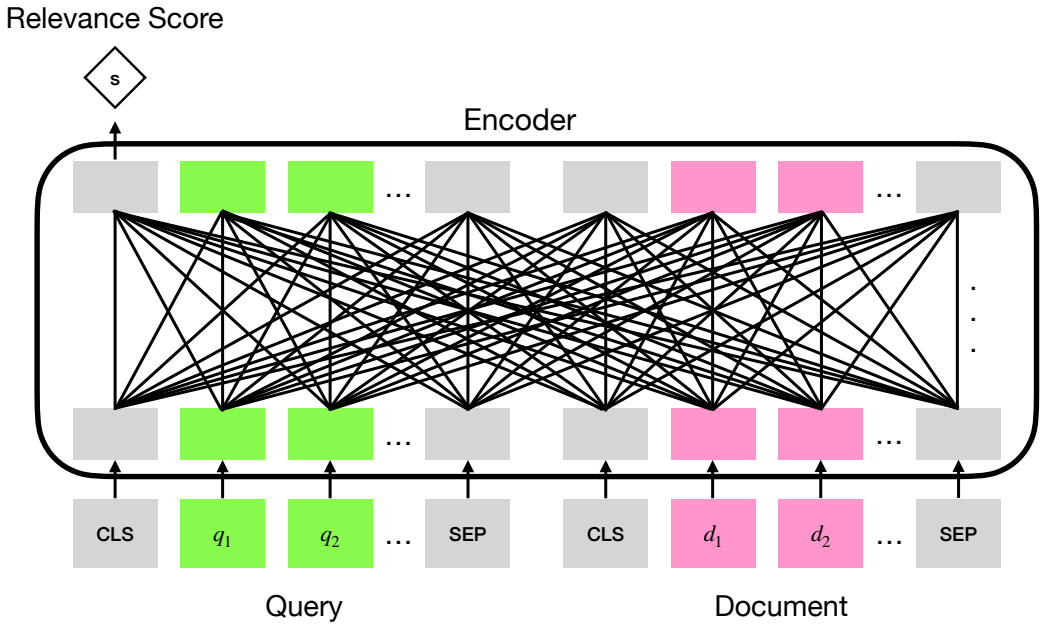


Figure 2.4: Overview of cross-encoder architecture.

model is trained as a binary classifier: the final hidden representation of the [CLS] token is passed through a linear classification head to produce a scalar relevance score between 0 and 1. A sigmoid activation function is applied to map the raw score into a probability space, where values closer to 1 indicate high relevance. The training objective is binary cross-entropy (BCE) loss, which penalizes incorrect predictions based on the labeled relevance of each query–document pair.

As illustrated in Figure 2.4, the cross-attention enables the model to capture fine-grained token-level dependencies between query and document, making it highly effective for relevance estimation. However, this architecture is computationally expensive, as it requires re-encoding both the query and each candidate document pair together, which limits scalability to large candidate sets. Despite this limitation, cross-encoder rerankers based on encoder PLMs remain a strong baseline in reranking tasks due to their ability to model deep query–document interactions.

As illustrated in Figure 2.4, the cross-attention mechanism allows the model to deeply integrate contextual information across the query and document. While highly effective in estimating fine-grained relevance, the BERT cross-encoder reranker is computationally expensive at inference time, since it requires pairwise joint encoding of each query with every candidate document. Despite this limitation, it remains a widely adopted approach due to its strong performance in reranking tasks. In this thesis, we use BM25+BERT as a competitive baseline for empirical evaluations.

Encoder-Decoder Based PLMs for Reranking: Encoder–decoder models, particularly the Text-to-Text Transfer Transformer (T5) [171], have also been applied to the reranking task by leveraging their strong generative and contextual modeling capabilities. Unlike encoder-only models, which produce relevance scores through classification heads over contextual embeddings, encoder–decoder models treat reranking as a sequence-to-sequence problem where the model generates an output token or textual label indicating the relevance of a query–document pair.

T5 follows a unified text-to-text framework, where both inputs and outputs are represented as natural language text. For reranking, each input is formatted as a textual prompt, such as:

```
Query: <query text> Document: <document text> Relevant:
```

The model also functions as a cross-encoder by jointly attending to both the query and the document within the encoder–decoder framework. It generates an output string such as "relevant", "irrelevant", or a numerical score (e.g., "3.4"), depending on the fine-tuning objective. This formulation casts reranking as a conditional text generation task, allowing T5 to fully leverage its pretraining across diverse natural language understanding and generation objectives.

The encoder processes the combined query–document input and generates contextual representations, which are then consumed by the decoder to autoregressively produce the output token. During fine-tuning, a cross-entropy loss is computed between the generated token sequence and the ground-truth label. This training strategy encourages the model to learn token-level relevance patterns and jointly model interactions between the query and document through both the encoder and decoder components.

T5-based rerankers have been shown to be highly effective on standard benchmarks, such as MonoT5 [159] and DuoT5 [166] extend this architecture for reranking individual query–document pairs or document pairs respectively. MonoT5 operates as a pointwise reranker that scores each query–document pair independently, while DuoT5 considers document pairs and predicts preference orderings, enabling pairwise ranking capability.

Although encoder–decoder rerankers typically require more computation than encoder-only models due to the decoder's autoregressive nature, they offer greater flexibility. Their generative formulation also facilitates applications beyond binary relevance prediction, such as multi-label classification or answer-aware reranking.

Decoder-Based LLMs for Reranking: With the emergence of decoder-only generative models [14, 78, 161, 202, 227], researchers also have explored their application to document reranking. Unlike encoder-based models, which typically encode the query and document independently, or encoder–decoder models, which use bidirectional encoding of inputs followed by cross-attentive generation, decoder-only architectures process all input and output in a single autoregressive stream. They generate output tokens one at a time, conditioning only on previously generated tokens due to causal masking, without input from encoder-side representations.

In the context of reranking, decoder-only models or LLMs are typically prompted with a combined textual input that includes the query and one or more candidate documents. A common approach is to frame the task as a question–answering or instruction-following prompt, such as:

```
Given the following query and document, determine if the document
is relevant to the query.
Query: <query text>
Document: <document text>
Answer:
```

The model generates a response, similar to the encoder-decoder models, by continuing the input prompt as a natural language sequence. Decoder-only models, such as those in the GPT family [14, 161, 169, 170], operate within a single Transformer stack and treat the entire input and output as one continuous autoregressive sequence. They are trained to generate one token at a time, conditioning only on previously generated tokens via causal masking.

However, unlike encoder–decoder models (e.g., T5 [171]), which separate input encoding from output generation and allow the decoder to attend to bidirectional encoder representations, decoder-only models lack a dedicated encoder and rely entirely on left-to-right token generation. As a result, they must learn all dependencies between query, document, and response within a single sequence without cross-attention mechanisms.

Although decoder-only models lack cross-attention mechanisms between query and document, they can be used in reranking tasks without the need for explicit classification heads or architectural modifications. Recent studies [154, 160, 167, 245] demonstrate that large-scale decoder-only models can achieve competitive performance on reranking tasks, even in zero-shot or few-shot settings.

Choi et al. [22] further exploit decoder-only models in in-context learning settings, where the model is presented with a small number of annotated query–document pairs followed by an unlabeled example. This allows the model to infer relevance by extrapolating from the in-context examples, without requiring weight updates.

2.2.3 Document Expansion and Learned Sparse Retrieval with Neural Language Models

Beyond using neural language models as rerankers, researchers have explored their use in improving first-stage retrieval via document expansion and representation refinement. These methods aim to resolve the 'vocabulary mismatch' problem [88], where the effectiveness of term-overlap models like BM25 [178] is limited because short, concise queries often fail to match the vocabulary of longer, more informative documents. To mitigate this, neural models are employed to predict and inject potential query terms into documents, a series of methods [11, 32, 33, 45, 46, 51, 111, 142, 156, 158] have emerged that use neural language models to expand or reweight document representations to better align with potential queries. A common strategy is to enrich document content by predicting and injecting query-relevant terms, a technique known as document expansion. The central idea is to make relevant documents more lexically similar to the types of queries that users are likely to issue, thereby improving retrieval effectiveness under lexical matching.

One of the early models in this direction is Doc2Query [158], which uses a sequence-to-sequence model fine-tuned on query–document pairs to generate synthetic queries for each document. These queries are then appended to the original document text, enriching the index with more relevant query terms. DocTTTTTQuery [156] further improves this idea by applying T5 [171] to generate high-quality synthetic queries at scale. Both approaches aim to reduce query–document mismatch by bridging the lexical gap from the document side.

Document expansion methods add predicted query terms to documents, but they do not explicitly adjust how documents are represented for retrieval task. Therefore, DeepCT [32] and HDCT [33] propose to re-weight document terms using contextualized representations from BERT [37]. These models predict a scalar weight for each term based on its contextual relevance within the document, indicating its importance for retrieval. Instead of altering the document text, they retain the original terms and associate each with a learned weight. These weights are stored in the inverted index and used at retrieval time to influence the relevance score. This design allows the incorporation of semantic signals from neural language models into standard sparse retrieval pipelines while preserving compatibility with traditional indexing infrastructure.

Further improvements were introduced by DeepImpact [142] and DeeperImpact [11], which extend earlier term-weighting approaches by incorporating supervised training objectives that directly relate term importance to document relevance. Instead of predicting term importance in isolation, these models learn to assign weights based on supervised relevance signals by comparing relevant and non-relevant document pairs. This training strategy encourages the model to produce more discriminative term representations that better separate relevant documents from non-relevant ones during retrieval.

Another line of research focuses on enhancing sparse retrieval by learning contextualized term-level representations through late interaction mechanisms. Models such as COIL [51] and uniCOIL [111] represent queries and documents at the token level, leveraging contextual embeddings derived from neural language models. Rather than relying on traditional lexical overlap, these models perform token-wise matching based on contextual similarity, enabling semantic interactions while maintaining the inverted index structure of sparse retrieval. This approach is commonly referred to as *learned sparse retrieval*, where sparse representations are learned to capture richer semantics than traditional term frequencies. In this thesis, we adopt uniCOIL as a representative strong learned sparse retriever due to its ability to capture and represent rich semantic signals from documents.

More recently, SPLADE [46] and its improved variant SPLADE-v2 [45] have significantly advanced learned sparse retrieval by introducing a method that combines the strengths of sparse representations with semantic understanding from language models. SPLADE encodes documents using a transformer-based encoder and maps contextualized token embeddings into a sparse vocabulary space. It applies a max-pooling operation over token-level activations to construct a bag-of-words-like representation, where each term is scored based on its importance in context. This allows SPLADE to capture both exact terms and semantically related expansions, effectively performing implicit document expansion.

Importantly, SPLADE maintains compatibility with traditional inverted indexes, making it highly practical. SPLADE-v2 builds on this foundation with improved training strategies and regularization techniques, resulting in both higher retrieval effectiveness and better efficiency.

In summary, this section provided a foundational overview of transformer-based neural language models and their applications in IR. It began by introducing the original Transformer architecture that act as the basis for most modern neural language models. We then examined representative models built on different Transformer variants, including encoder-only (e.g., BERT [37]), encoder–decoder

(e.g., T5 [171]), and decoder-only (e.g., GPT [169]) architectures along with their roles in text ranking tasks. Finally, we explored how these models are leveraged beyond reranking, particularly in document expansion and learned sparse retrieval, where they help mitigate vocabulary mismatch and enhance first-stage retrieval. Together, these discussions establish a strong conceptual basis for the retrieval models and to equip the readers with necessary background knowledge on neural language models for this thesis.

2.3 Dense Retrieval

In the previous sections, we discussed sparse retrieval models and the integration of transformer-based neural language models into various retrieval pipelines and models, including retrieve-and-rerank frameworks, document expansion, and learned sparse retrieval approaches. Sparse retrieval models are fundamentally built upon inverted indexes derived from fixed-vocabulary sparse term representations. In retrieve-and-rerank pipelines, the effectiveness of the overall system depends on a high-recall first-stage retriever, which typically operates over such inverted indexes to efficiently generate candidate document sets for reranking. Learned sparse retrieval models enhance traditional indexing by incorporating contextualized term importance into the representation; however, their underlying index structure remains sparse and vocabulary-based. While these methods offer measurable improvements over classical sparse models, they continue to rely on sparse representations, which may still lose important semantic relationships between the terms despite the efficiency in storage and computation for such indices.

Due to the limitations of sparse representations, dense retrieval has emerged as a compelling alternative for leveraging neural language models in IR. Sparse methods represent text as high-dimensional vectors with most entries set to zero (due to absence of terms), while dense retrieval encodes queries and documents into more fine-grained low-dimensional, continuous vectors. These semantically rich dense representations enable retrieval even in the absence of shared terms between query and documents. Structurally, dense retrievers typically adopt a *bi-encoder (or dual-encoder)* architecture (Figure 2.5), where queries and documents are encoded independently, and their relevance is computed via a similarity function over their vector representations. This is different from cross-encoder architectures, such as BERT-based rerankers [131, 155], which jointly encode query-document pairs but are computationally expensive at scale. By decoupling encoding and allowing efficient similarity computation, dense retrieval facilitates semantic matching while remaining scalable. In the following sections, we introduce two major categories of dense retrievers based on their encoding strategies: single-representation and multi-representation approaches.

2.3.1 Single-Representation Dense Retrieval

The single-representation approach is the most fundamental and widely adopted paradigm in dense retrieval, such as DPR [83], ANCE [223], RepBERT [238], TCT-ColBERT [114], TCT-ColBERT-

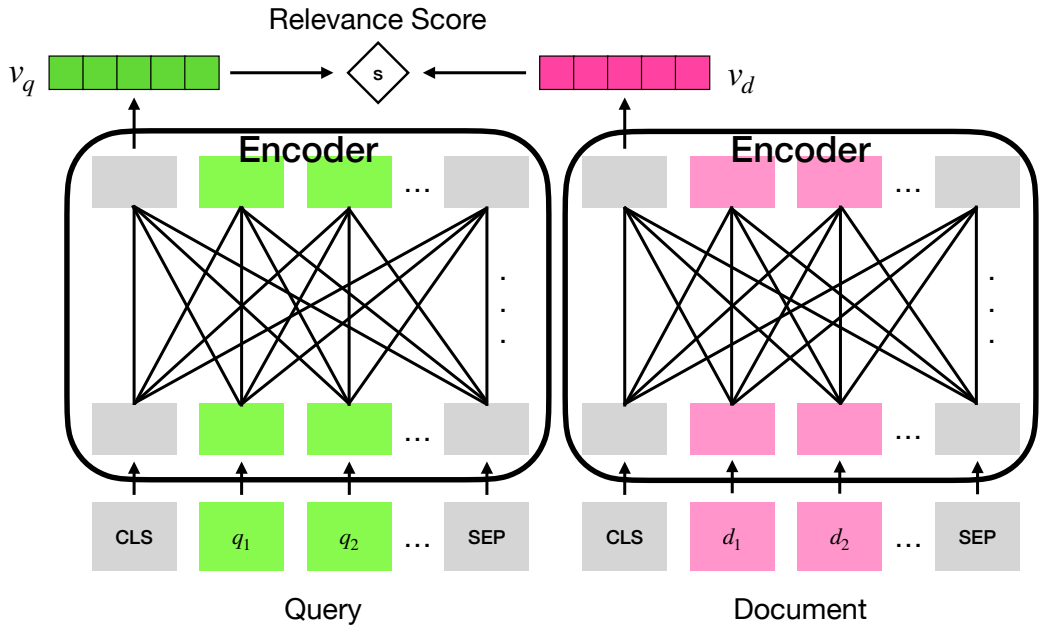


Figure 2.5: Overview of bi-encoder architecture.

v2 [115], DistillBERT KD [72], DistillBERT Balanced TASB [73], ADORE [239], PAIR [174], RocketQA [168], RocketQA-v2 [175], CLEAR [53], Condenser [48], coCondenser [50], SimLM [208], AR2 [240], RetroMAE [222]. In this architecture, a query and a document are each encoded independently into a fixed-size dense vector using two separate but often parameter-shared encoders. These vectors, typically referred to as *query embedding* and *document embedding*, represent semantic information in continuous space.

To put it formally, given a query q and a document d, the encoder $f(\cdot)$ maps each input to a dense vector:

$$v_q = \mathbf{q} = f_q(q), \quad v_d = \mathbf{d} = f_d(d) \tag{2.7}$$

The relevance score between the query and document is then computed using a similarity function, most commonly the dot product or cosine similarity:

$$s(q,d) = \mathbf{q}^\top \mathbf{d} \quad or \quad \frac{\mathbf{q}\cdot\mathbf{d}}{\|\mathbf{q}\|\|\mathbf{d}\|} \tag{2.8}$$

This formulation allows for pre-computing and indexing document vectors using Approximate Nearest Neighbor (ANN) libraries such as FAISS [41], enabling efficient and scalable retrieval in large corpora.

Most dense retrievers adopt encoder-only transformer models like BERT [37], RoBERTa [119], DistilBERT [186], or ERNIE [194]. Representations are typically derived from the [CLS] special token embedding or pooled from the final-layer hidden states. Encoder-decoder models such as T5 [171] have also been adapted for single-vector retrieval [152, 153].

Dense retrievers using this approach include a broad range of models with distinct training objectives and strategies:

Dense Passage Retrieval (DPR) [83]: is a seminal dual-encoder architecture developed for open-domain Question Answering (QA) and serves as a foundational model in dense retrieval research. DPR independently encodes queries and passages into dense vectors using two BERT-based encoders, enabling efficient retrieval through ANN search in vector space.

During training, DPR is optimized using a contrastive loss that encourages the similarity between a query and its corresponding positive passage to be higher than that with negative passages. The negatives are sampled from other passages within the same mini-batch (i.e., in-batch negatives), which improves training efficiency and provides harder negative samples than random selection.

The training objective is defined as:

$$\mathcal{L}_{DPR} = -\log \frac{\exp(\mathbf{q} \cdot \mathbf{d}^+)}{\exp(\mathbf{q} \cdot \mathbf{d}^+) + \sum_{i=1}^{n} \exp(\mathbf{q} \cdot \mathbf{d}_\mathbf{i}^-)} \tag{2.9}$$

where $\mathbf{q}$ is the query embedding, $\mathbf{d}^+$ is the embedding of the corresponding positive passage, and d_i^- are the embeddings of negative passages. The dot product is used as the similarity function, though cosine similarity may also be applied.

One of DPR's key contributions lies in its use of independently encoded dense representations for both queries and documents, enabling large-scale retrieval without the need for computationally expensive cross-attention mechanisms. It also introduced a practical pipeline for training dense retrievers using QA datasets and demonstrated significant improvements over sparse baselines such as BM25. This architecture has since inspired a broad array of dense retrievers that build upon its bi-encoder framework with various enhancements to pretraining, sampling, and distillation strategies.

RepBERT [238]: builds upon the DPR framework and focuses on improving the generalization ability of dense retrievers through more effective fine-tuning strategies. Specifically, RepBERT is trained on a large-scale passage ranking dataset using a similar BERT-based bi-encoder architecture as DPR. The representation for each input is obtained by extracting the embedding of the [CLS] token from the final transformer layer.

A central innovation of RepBERT is its emphasis on hard negative mining to enhance training robustness. Rather than relying solely on randomly sampled or in-batch negatives as in DPR, RepBERT incorporates harder negatives that are semantically similar to the query but non-relevant, thereby making the discrimination task more challenging. This approach reduces overfitting to surface-level lexical signals and encourages the model to capture finer-grained semantic signals.

The model is trained using a contrastive loss similar to that of DPR (Eq. 2.9), which optimizes the similarity between a query and its relevant documents while minimizing similarity to negatives. By using high-quality hard negatives and a stable BERT backbone, RepBERT achieves improved retrieval performance and better generalization across domains, making it a strong first-stage retriever for neural ranking pipelines. Because of its simple architecture and effectiveness, we adopt RepBERT as a baseline dense retriever model in this thesis.

ANCE (Approximate Nearest Neighbor Negative Contrastive Learning) [223]: introduces a novel training paradigm for dense retrieval that addresses a key limitation of earlier models such as DPR and RepBERT: reliance on static, potentially uninformative negative samples. ANCE proposes

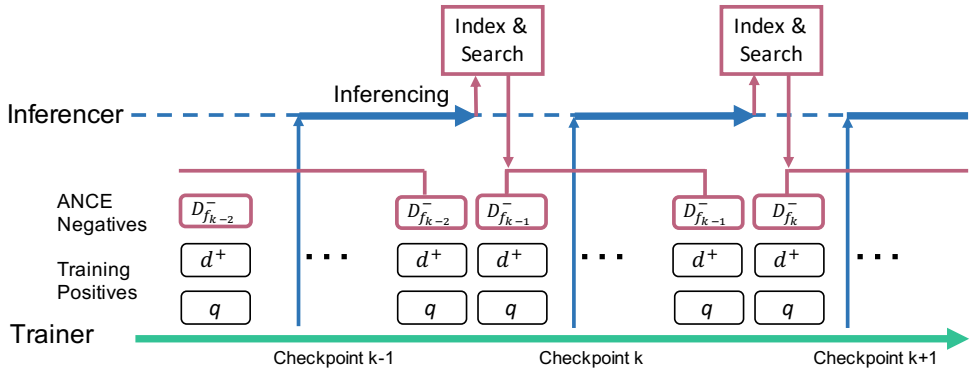


Figure 2.6: ANCE training pipeline [223].

an iterative training framework in which hard negatives are dynamically mined during training using ANN search over a continuously updated document index as illustrated in Figure 2.6.

In each training iteration, the model encodes all passages into dense vectors and builds an ANN index. Given a query, the top-ranked passages retrieved from the index, excluding the ground-truth positives, are treated as hard negatives. These negatives are semantically close to the query in embedding space, making them substantially more challenging than random or in-batch negatives. The retrieval model is then updated using contrastive loss, and the ANN index is refreshed periodically to reflect the model's improved representation space.

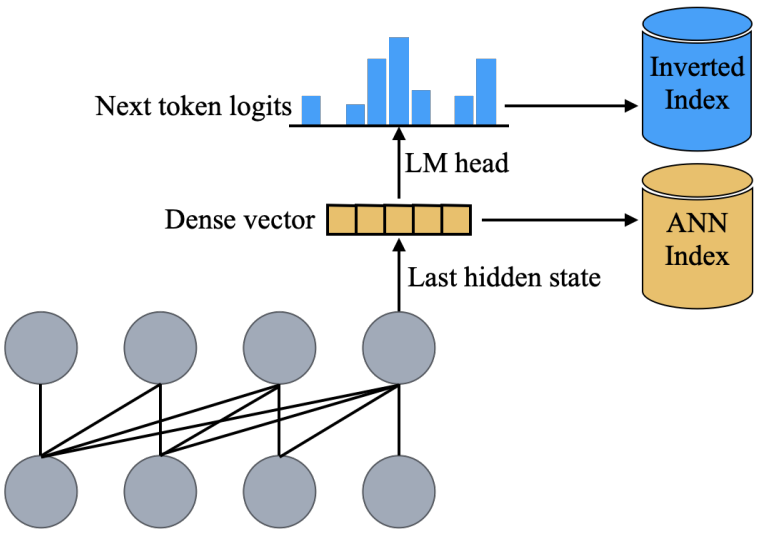
Formally, the training loss function remains a contrastive loss function as Eq. 2.9, but the key difference lies in how the negative samples are selected dynamically through ANN rather than from fixed batches.

This online hard negative mining strategy significantly mitigates the risk of overfitting to a narrow set of training negatives and further encourages the model to refine its embedding space toward finer semantic distinctions. ANCE was among the first to demonstrate that hard negatives retrieved via the model itself lead to stronger performance and faster convergence, particularly on large-scale benchmarks. It has since influenced many subsequent retrieval models that incorporate similar index-refreshing and self-improving feedback loops.

Beyond these core models, many variants have emerged, emphasizing knowledge distillation, unsupervised pretraining, or improved sampling, such as [48, 114, 115, 168, 175]. While these methods differ in their training strategies, they share the same inference pipeline: independently encoding the query and documents, and then performing similarity search via ANN. This design ensures retrieval efficiency comparable to traditional sparse methods, but with significantly improved effectiveness due to their ability to capture semantic similarity in the embedding space.

The proliferation of generative, decoder-only language models has advanced the development of *single-vector* dense retrievers grounded in LLM representations, as exemplified by PromptReps [250] and RepLLaMA [128].

PromptReps [250] presents a zero-shot approach to transform decoder-only generative LLMs into dense retrievers. Without further fine-tuning, PromptReps can effectively produce single-vector



<System> *You are an AI assistant that can understand human language.*
<User> *Passage: "***[text]***". Use one word to represent the passage in a retrieval task. Make sure your word is in lowercase.*
<Assistant> *The word is: "*

Figure 2.7: Overview of PromptReps [250].

embeddings for queries and documents with a single forward pass via a frozen LLM backbone (Figure 2.7).

PromptReps adopts a simple and straightforward prompt (e.g., *"Passage: 'passage'. Use one word to represent the passage in a retrieval task. The word is:"*) to guide the underlying LLM for generating representations. From the single forward pass, two types of embeddings are extracted: (1) a dense vector from the final hidden state of the last input token (e.g., the '"' token in this case), and (2) a sparse vector from the output logits, further filtered based on the input tokens. These two types of vectors can be used as vector representations independently or together (fused) for hybrid retrieval.

Since PromptReps is zero-shot and extracts embeddings from the last token's final hidden state via prompting, this framework can be applied to any instruction-tuned LLM of varying sizes. Due to its zero-shot nature, it requires no training or architectural modifications. One key observation is that prompt phrasing critically affects performance; simple prompts like *"The word is:"* help stabilize outputs.

In this thesis, we extensively adopt PromptReps as a zero-shot, LLM-based dense retriever baseline in Chapters 10 and 11.

RepLLaMA [128] introduces a framework for adapting decoder-only large language models, specifically LLaMA[202], into dense dual-encoder retrievers through contrastive fine-tuning. Unlike LLM-based zero-shot dense retrieval methods such as PromptReps, RepLLaMA is fine-tuned to generate dense vector representations for both queries and documents. The core idea is that decoder-style LLMs, despite their autoregressive nature, can be adapted to effectively encode query and documents that leads to improved retrieval performance when optimized with suitable training objectives.

RepLLaMA encodes queries and documents separately, each using a full LLaMA model with fine-tuned LoRA [74] modules. The inputs to the model are appended with a special end of sequence

token $< /s >$, and the embedding is extracted from the final hidden state of this last token. These embeddings are used to represent query and documents, then used to compute the similarity scores between query and document embeddings. The model is fine-tuned using a contrastive loss (Eq. 2.10) with in-batch negatives and additional hard negatives mined from an efficient retriever such as BM25. For each training batch, one query is paired with multiple relevant and non-relevant documents. To put it formally, the loss function is defined as:

$$\mathcal{L} = -\log \frac{\exp\left(\text{sim}(\mathbf{q}, \mathbf{d}^+)\right)}{\exp\left(\text{sim}(\mathbf{q}, \mathbf{d}^+)\right) + \sum_{i=1}^{k} \exp\left(\text{sim}(\mathbf{q}, \mathbf{d}_i^-)\right)} \tag{2.10}$$

where $\mathbf{q}$ is the query embedding, $\mathbf{d}^+$ is the embedding of the positive (relevant) document, $\mathbf{d}_i^-$ are embeddings of k negative (irrelevant) documents, $\text{sim}(\cdot,\cdot)$ denotes a similarity function.

One of the key features of RepLLaMA is that it requires minimal architectural modification. RepLLaMA retains the autoregressive decoder-only architecture and uses further fine-tuning to produce embeddings, effectively transforming the generative LLM into a dense retriever. During inference, both documents and queries are encoded independently, and retrieval is performed via Approximate Nearest Neighbor (ANN) search using the pre-computed document embedding index.

In this thesis, RepLLaMA serves as a representative decoder-style dense retriever and is included as a baseline in Chapter 10 to evaluate the VPRF methods (Chapter 5) to trained LLM retrievers. Compared to zero-shot approaches like PromptReps, RepLLaMA relies on supervised training to produce embeddings for query and documents, making it a valuable comparative baseline for assessing the generalizability and efficiency of our proposed PRF frameworks.

There are other decoder-only single-representation models that have been explored, such as LLM2Vec [95], where a frozen LLM is prompted to generate dense vector representations by averaging the hidden states of selected tokens (e.g., content-bearing tokens or special positions) from its final layer. Unlike PromptReps, which relies purely on prompt design without any post-processing, LLM2Vec applies lightweight aggregation strategies to enhance representation quality while maintaining training-free deployment. We also adopted it as a baseline in Chapter 10 to evaluate the effectiveness of our proposed feedback methods under zero-shot, decoder-based settings. In the next section, we explore multi-representation dense retrievers, which extend beyond the limitations of single-vector encoding by learning token-level embeddings per query or document, enabling more expressive and fine-grained similarity matching.

2.3.2 Multi-Representation Dense Retrieval

Despite their scalability and efficiency, single-representation dense retrievers suffer from a key limitation: because queries and documents are encoded independently into single embeddings, the model is inherently constrained by its ability to capture fine-grained semantic interactions between the tokens from query and document. This bottleneck can lead to representational loss, hence to degraded performance.

To address this shortcoming, researchers have explored multi-representation dense retrieval models, which enhance the interaction granularity. Among these, the most prominent and foundational approach is ColBERT (Contextualized Late Interaction over BERT) [85], followed by its enhanced variant ColBERTv2 [187].

Unlike single-vector models that aggregate token information into a single embedding, ColBERT uses BERT-based encoders to produce contextualized per-token representations for both queries and documents. Formally, given a query (list of tokens from vocabulary V) $q = \{q_1, q_2, ...q_m\}$ and a document (list of tokens from vocabulary V) $d = \{d_1, d_2, ..., d_n\}$, the encoders produce token-level embeddings:

$$\mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_m], \quad \mathbf{D} = [\mathbf{d}_1, \mathbf{d}_2, ..., \mathbf{d}_n] \tag{2.11}$$

where $\mathbf{q}_i, \mathbf{d}_j \in \mathbb{R}^h$ are contextualized embeddings for the i^{th} query token and j^{th} document token, respectively.

To enable efficient yet expressive matching, ColBERT introduces a late interaction mechanism that defers token-level interactions to the similarity computation stage, rather than performing full cross-attention during encoding as in cross-encoders [155] (see Figure 2.4). Specifically, the relevance score is computed using the *MaxSim* operator over token pairs:

$$s(q,d) = \sum_{i=1}^{m} \max_{1 \le j \le n} \langle \mathbf{q}_i, \mathbf{d}_j \rangle \tag{2.12}$$

This formulation identifies, for each query token $\mathbf{q}_i$, the document token $\mathbf{d}_j$ with the highest similarity, thus enabling fine-grained matching without the computational cost of joint encoding. By approximating token-level interactions in this manner, ColBERT effectively addresses the limitations of single-representation models and achieves notable gains in retrieval performance.

However, representing each query and document with multiple token embeddings increases index size and query latency compared to single-representation dense retrievers. To address these challenges, ColBERTv2 [187] introduces residual compression and improved indexing techniques, significantly reducing storage and latency overhead. Additionally, COIL [51], discussed in Section 2.2.3, accelerates token-level retrieval by restricting interactions to exact token matches, aligning with classical inverted indexing principles. Together, these advances enhance the scalability and practicality of ColBERT-style multi-representation dense retrieval in large-scale systems.

In summary, dense retrieval models have emerged as a powerful alternative to sparse methods by leveraging transformer-based neural language models to encode queries and documents into continuous vector representations. Unlike sparse retrievers that rely on exact term matching, dense retrievers capture richer semantic similarity and are typically implemented using bi-encoder architectures. This section introduced two major categories: single-representation models, such as DPR [83], RepBERT [238], and ANCE [223], which encode each query or document into a single vector; and multi-representation models like ColBERT [218], which retain token-level granularity through late interaction mechanisms.

2.4 Pseudo-Relevance Feedback (PRF)

As discussed in Section 2.1, classical sparse retrieval models often struggle when there is a mismatch between the lexical surface forms used in queries and those present in relevant documents. These limitations are especially evident in cases of ambiguous queries, where term overlap fails to capture the user's true information need. More recent advances, including the use of transformer-based rerankers (Section 2.2.2) and dense retrieval models (Section 2.3), have shown effectiveness in alleviating semantic mismatch by leveraging the representational capacity of neural language models to model deeper semantic relations between queries and documents. In parallel, document expansion techniques (Section 2.2.3) have been explored as a way to enrich document representations with potentially query-relevant terms, thereby mitigating issues stemming from lexical sparsity.

Despite these advancements, query formulation remains a persistent challenge across all retrieval paradigms. Users often submit short, vague, or under-specified queries due to a lack of domain knowledge or search expertise [17, 188, 189]. Even with dense retrievers that have shown notable improvements by representing semantic similarity in a learned vector space, their effectiveness remains closely tied to the expressiveness of the initial query [243]. To address this, a range of query reformulation strategies have been explored, encompassing techniques such as query rewriting [6, 117], query reduction [252], intent detection [150], external knowledge-guided reformulation [19, 62], and query expansion [10, 12, 18, 65, 67, 132, 133, 141, 224]. Among these, Pseudo-Relevance Feedback (PRF) [36, 63, 76, 110, 130, 183, 207, 235], a widely studied form of query expansion, has proven to be one of the simplest and most effective methods for improving retrieval performance [1, 94, 134, 149, 179, 218, 244]. PRF operates under the assumption that the top-ranked documents retrieved in an initial search are likely to contain relevant information, even in the absence of explicit relevance judgments.

The standard PRF framework typically involves three main stages. First, an initial retrieval step retrieves a ranked list of documents based on the original query. Second, feedback signals [66], such as salient terms or representations, are extracted from this top-ranked set (treated as pseudo-relevant usually without ground truth judgements), and used to construct an expanded or reformulated query. Third, this new query is used either to rerank the initial candidates or to perform a second round of retrieval over the entire corpus. This iterative refinement process enables the system to move closer to the user's true intent, often recovering relevant documents that would have been missed due to vocabulary mismatch or incomplete queries.

PRF serves as an effective mechanism to enrich the query, alleviate the vocabulary mismatch, thereby improving the retrieval performance [9]. Originally developed in the context of classical information retrieval [4, 91, 179, 237], PRF has since been widely adopted and reimagined in neural settings [77, 94, 138, 139, 172, 217, 218, 233, 234], where it plays a key role in enhancing retrieval performance.

In the remainder of this section, we present a comprehensive review of PRF methods. We begin with traditional approaches integrated with sparse retrieval models in Section 2.4.1, and then turn to

neural PRF methods ranging from those designed for retrieve-and-rerank pipelines to those developed for dense retrievers in Section 2.4.2. This structure reflects the historical progression of PRF research while highlighting the methodological shifts that have accompanied the transition to neural and dense retrieval paradigms.

2.4.1 Sparse PRF Approaches

To better understand the development of PRF techniques, we begin with traditional approaches originally designed for sparse retrieval systems (Section 2.1), namely Sparse PRF approaches. These methods expand or reweight the original user query using term-based statistics derived from top-ranked documents to address the vocabulary mismatch problem. By modifying the original query, sparse PRF introduces pseudo-relevant feedback signals that help reformulate the query into a more effective form. This enriched query thus helps the model to retrieve more relevant content align with user's intent. Despite their simplicity, sparse PRF methods have demonstrated strong empirical performance and serve as the foundation for more advanced feedback strategies.

Rocchio Algorithm: The Rocchio algorithm, originally introduced by Rocchio [179], is one of the earliest and most influential relevance feedback methods developed within the vector space model framework. Its core objective is to refine the original query vector by incorporating information from known relevant and non-relevant documents to enhance retrieval effectiveness. Specifically, the algorithm adjusts the query vector by shifting it toward the centroid of relevant document vectors while simultaneously distancing it from the centroid of non-relevant document vectors. This process can be formally represented as:

$$\mathbf{q}_{\text{rocchio}} = \alpha \cdot \mathbf{q}_{\text{original}} + \frac{\beta}{|D_R|} \sum_{\mathbf{d}_i \in D_R} \mathbf{d}_i - \frac{\gamma}{|D_N|} \sum_{\mathbf{d}_j \in D_N} \mathbf{d}_j \tag{2.13}$$

In this formulation, $\mathbf{q}_{\text{original}}$ denotes the initial query vector, $\mathbf{q}_{\text{rocchio}}$ is the updated query vector, and D_R and D_N represent the sets of relevant and non-relevant documents, respectively. The parameters α, β, and γ are scalar weights that modulate the influence of the original query, relevant documents, and non-relevant documents. A higher value of β increases the impact of relevant documents, while a higher γ penalizes similarity to non-relevant documents.

While the Rocchio algorithm offers a conceptually simple and computationally efficient approach to relevance feedback, it presumes the availability of explicit relevance judgments to construct the sets of relevant (D_R) and non-relevant (D_N) documents. In practical retrieval scenarios, however, such judgments are seldom available. To address this, PRF heuristically assumes that the top-ranked documents retrieved in an initial search are relevant, while lower-ranked documents are considered non-relevant. This approximation allows the Rocchio algorithm to be deployed in fully automated settings. In Chapter 5, we extend this idea to neural retrieval by employing a Rocchio-inspired vector manipulation strategy within dense retriever architectures, aiming to preserve both retrieval effectiveness and computational efficiency. The next section explores RM3, a probabilistic term-

weighting model that has been widely adopted in PRF for its robust performance in lexical retrieval frameworks.

RM3: The RM3 PRF model, introduced by Abdul-Jaleel et al. [1], is a probabilistic term weighting approach grounded in the language modeling framework for information retrieval. RM3 enhances the initial query by estimating a relevance model from the top-ranked documents retrieved in the first-stage retrieval. These documents are treated as samples from an implicit relevance distribution, and their aggregated term statistics are used to construct an expanded query model. This enriched model incorporates terms that are statistically associated with the original query across the feedback set, thereby improving retrieval effectiveness through contextual expansion.

Formally, RM3 operates as follows. The smoothed document language model is estimated using Dirichlet prior smoothing [236]:

$$P(w \mid D) = \frac{c(w,D) + \mu, P(w \mid C)}{|D| + \mu} \tag{2.14}$$

where $c(w,D)$ is the count of term w in document D, $|D|$ is the document length, $P(w \mid C)$ is the term probability in the collection, and μ is the smoothing parameter. The likelihood of the query Q given a document D is computed as:

$$P(Q \mid D) = \prod_{i=1}^{|Q|} P(q_i \mid D), \quad \text{often evaluated via} \quad \log P(Q \mid D) = \sum_i \log P(q_i \mid D) \tag{2.15}$$

The original query model $P(w \mid Q)$ is a maximum likelihood estimate over query terms:

$$P(w \mid Q) = \frac{c(w,Q)}{|Q|} \tag{2.16}$$

A relevance model $P(w \mid R)$ is then estimated by aggregating the document models $P(w \mid D)$ across the feedback set R, weighted by the query likelihood scores $P(Q \mid D)$:

$$P(w \mid R) = \sum_{D \in R} P(w \mid D) \cdot \frac{P(Q \mid D)}{\sum_{D' \in R} P(Q \mid D')} \tag{2.17}$$

The final expanded query model is obtained by interpolating the original query model with the relevance model:

$$P_{\text{RM3}}(w \mid Q) = \lambda, P(w \mid R) + (1 - \lambda), P(w \mid Q) \tag{2.18}$$

To score documents in the second retrieval stage, the KL-divergence [89] between the expanded query model and each document model is computed:

$$\text{score}(D,Q) = \sum_w P_{\text{RM3}}(w \mid Q) \cdot \log \frac{P_{\text{RM3}}(w \mid Q)}{P(w \mid D)} \tag{2.19}$$

Conceptually, RM3 consists two stages: the initial retrieval step yields a pseudo-relevant set R, from which the relevance model $P(w \mid R)$ is constructed. This model captures term-level associations inferred from documents that are most likely to explain the original query, with $P(Q \mid D)$ acting as a

soft signal of document relevance. The document models $P(w \mid D)$ are smoothed using the Dirichlet prior [89] to ensure robustness against sparse term statistics.

In the second stage, the expanded query model $P_{\text{RM3}}(w \mid Q)$ is derived via interpolation, where λ controls the relative influence of feedback-derived terms with the original query terms. This blending is essential for mitigating query drift: it introduces contextually relevant vocabulary without fully discarding the user's original intent. In practice, $P(w \mid R)$ is often pruned to retain only top-weighted terms and exclude meaningless words (i.e., stopwords [124]), enhancing stability and interpretability.

The final scoring step views retrieval as a divergence minimization problem, where documents are preferred if their smoothed language models closely match the expanded query distribution. The parameter μ controls the degree of smoothing in $P(w \mid D)$ while λ regulates the feedback strength. Together, these parameters make RM3 a robust and effective PRF strategy, capable of improving early precision by enriching queries with semantically aligned terms derived from high-confidence feedback documents.

2.4.2 Neural PRF Approaches

While traditional sparse PRF methods such as Rocchio and RM3 introduced in Section 2.4.1 have proven effective in sparse retrieval settings [1, 140, 164, 179, 214], their reliance on lexical matching and term frequency statistics limits their applicability in neural architectures. As neural model based contextual representations become increasingly prevalent, new PRF strategies have emerged to better align with these models' representational and learning paradigms. Table 2.1 provides an overview of representative neural PRF approaches, highlighting their core mechanisms and key attributes.

CEQE

Contextual Embedding-based Query Expansion (CEQE) proposed by Naseri et al. [149] offers an instructive precedent for adapting PRF to transformer architectures. CEQE relies on BERT [37] to encode candidate expansion terms *in situ* within the pseudo-relevant passages retrieved at the first stage. By ranking those terms according to their contextual similarity to the original query, which is estimated with a neural ranker trained to predict per-term performance gains, CEQE departs from frequency-driven approaches such as RM3 [1] and Rocchio [179]. Empirical results on standard benchmarks confirm that exploiting local context produces sizeable effectiveness improvements over bag-of-words PRF, thereby substantiating the broader claim that neural representations can mitigate query–document vocabulary mismatch.

While CEQE still operates at the *term* level and thus incurs the overhead of generating, ranking, and appending individual expansions, it anticipates one of the central arguments of this thesis: feedback signals expressed in dense, context-sensitive representations are more informative than those derived from surface statistics alone. The ensuing chapters generalise this intuition beyond discrete terms. Vector-based PRF (Chapter 5) and Transformer-based PRF (Chapter 8) directly manipulate or adapt query embeddings themselves, eliminating the term-selection bottleneck and reducing inference latency,

Model	Description	Key Characteristics
CEQE [149]	Formulates query expansion as a ranking task, using contextualized token embeddings from pretrained transformers (e.g., BERT) to select semantically aligned expansion terms from top-ranked documents.	BERT; sparse retrieval; reranking; **term-based**
PGT [233]	models candidate expansion terms and their interactions using a graph neural network enhanced with a transformer encoder, allowing relevance signals to propagate across terms in a structured manner.	graph Transformer; BERT; sparse retrieval; **interaction-based**
NPRF [94]	Aggregates top-ranked documents with neural attention to form a feedback-aware query vector used in a second retrieval stage.	reranking; sparse retrieval; **interaction-based**
BERT-QE [20, 244]	Jointly encodes the query with feedback document chunks in BERT and incorporates list-wise ranking context to model interactions between the query and multiple feedback documents in a single end-to-end re-ranking pass.	BERT; efficiency; reranking; sparse retrieval; **term-based**
ANCE-PRF [234]	Concatenates the query with top-k passage texts and re-encodes the sequence using a fine-tuned ANCE encoder to yield a stronger dense query embedding.	dense retrieval; BERT; PRF query encoder; fine-tuning; **embedding-based**
ColBERT-PRF [214, 218]	Clusters token embeddings from pseudo-relevant documents and appends representative vectors to the multi-vector query, enhancing dense retrieval.	dense retrieval; multi-representation; k-means clustering; reranking or retrieval; efficiency; **embedding-based**
GRF [138, 139]	Uses an encoder–decoder LLM to generate expansion text conditioned on pseudo-relevant feedback, benefiting sparse, dense and learned-sparse retrieval.	generative; LLM; dense retrieval; sparse retrieval; feature-based; efficiency; prompt; zero-shot; **term-based**
Query2doc [209]	Few-shot prompts a LLM to generate a short pseudo-document appended to the query, boosting both sparse and dense retrieval.	generative; LLM; few-shot; sparse retrieval; prompt; **term-based**
DRQR [213]	Frames query reformulation as a reinforcement-learning task in which an agent iteratively rewrites the query to maximise retrieval reward.	reinforcement learning; two-stage training; paraphrasing; **term-based**
EPRF [215]	Augments queries with dense embeddings from an external corpus before applying PRF, markedly improving zero-shot dense retrieval.	dense retrieval; external knowledge; zero-shot; **term-based (sparse)**; **embedding-based (dense)**
CWPRF [216]	Learns a contrastive weighting network to select and weight discriminative feedback embeddings, mitigating drift in dense query expansion.	dense retrieval; contrastive weighting; **embedding-based**
OPRF [220]	Pre-generates pseudo-queries offline so that single-pass dense retrieval retains PRF gains without extra runtime latency.	dense retrieval; offline feedback; single-pass retrieval; efficiency; pseudo-query index
ReDE-RF [77]	Injects relevance-feedback embeddings into the query at inference time, enabling fast zero-shot dense retrieval with feedback signals.	dense retrieval; LLM; zero-shot; **embedding-based**

Table 2.1: Overview of representative PRF models.

while a prompt-driven variant (Chapter 11) extends the paradigm to decoder-style large language models. In this sense CEQE serves as a conceptual bridge between classical expansion methods and the representation-level feedback strategies advanced in the present work.

PGT

Pseudo-Relevant Graph Transformer (PGT) by Yu et al. [233] recasts expansion into a heterogeneous graph learning problem, in which candidate terms harvested from the initial ranking are vertices and semantic, positional, or co-occurrence relations form typed edges. A graph transformer infers contextual embeddings over this structure, allowing inter-term dependencies to be propagated via multi-hop attention rather than through simple term–document statistics. The resulting node representations are scored to select expansion terms that reflect the higher-order interactions latent in the pseudo-relevant set, yielding consistent gains over frequency-based baselines.

PGT is of particular relevance to this thesis because it demonstrates that richer structural priors, here, graph topology, can be injected into PRF without abandoning transformer encoders. Chapters 8 and 11 build on the same premise, but move from discrete term graphs to fully continuous embedding spaces and decoder-style language models. In doing so, the thesis explores whether the benefits of relational reasoning observed in PGT can be retained while streamlining inference and aligning feedback mechanisms with modern dense and prompt-based retrievers.

NPRF

Neural Pseudo-Relevance Feedback (NPRF) proposed by Li et al. [94] reimagines feedback usage by fusing signals from pseudo-relevant documents directly into the ranking network rather than altering the query string. The model first derives a baseline query–document relevance score, then refines that estimate through a secondary matching phase in which the query is paired with the top-k feedback documents. Convolutional encoders and attention layers distill layered interaction features, which are aggregated to produce a final ranking score. Because all computations occur in the representation space, NPRF captures nuanced contextual cues that term-level expansion often overlooks, and empirical studies show clear improvements over frequency-based baselines such as RM3 and Rocchio as well as earlier neural rankers lacking feedback channels.

For this thesis, NPRF exemplifies a shift from surface-level query reformulation toward embedding-level feedback, aligning with the broader argument that semantic signals are most effectively leveraged where the retriever operates—within dense representations. Subsequent chapters extend this insight by directly modifying the query vector itself (Chapter 5), adapting transformer encoders to consume feedback passage embeddings (Chapter 8), and employing decoder-style prompting mechanisms (Chapter 11), thereby generalising NPRF's core principle across model architectures and feedback granularities.

BERT-QE and Enhancement

BERT-QE of Zheng et al. [244] treats expansion as a token-level matching problem mediated by BERT. Terms harvested from the pseudo-relevant set are appended to the original query, and a second BERT encoder evaluates candidate documents given this enriched input. By aligning query tokens, expansion terms, and document content through self-attention, the model surpasses frequency-driven methods such as RM3. Chen et al. [20] advance this idea by conditioning the same encoder on local ranking context—document rank positions and neighbouring results—so relevance estimation is refined in a single end-to-end pass rather than through a separate expansion stage. The list-wise formulation yields further gains, demonstrating that contextual signals derived from both feedback passages and their relative order can be synergistically captured within a transformer re-ranker.

These works foreshadow several themes explored in this thesis. First, they confirm that transformer attention provides a natural mechanism for integrating feedback cues beyond raw term frequencies. Second, the enhancement illustrates the advantages of embedding feedback directly in the ranking architecture, a design principle shared by the embedding-level and prompt-based feedback frameworks developed in Chapters 5 and 8. Finally, their findings motivate our investigation into whether similar context integration can be achieved while reducing inference overhead through representation-space manipulation rather than token-level concatenation.

DRQR

Deep Reinforcement Query Reformulation (DRQR) by Wang et al. [213] casts expansion as a sequential decision process, training a policy network with reinforcement learning [195] to select terms from pseudo-relevant passages over multiple steps. The agent optimises an explicit retrieval reward, thereby learning to balance exploration of novel vocabulary against exploitation of known-useful terms while maintaining a memory of prior choices to avoid redundancy. This dynamic policy yields query-specific reformulations that adapt to topic difficulty and term ambiguity, producing competitive effectiveness relative to static PRF baselines.

DRQR is pertinent to this thesis because it frames feedback utilisation as an optimisation problem rather than a deterministic heuristic, reinforcing the broader contention that adaptive mechanisms outperform one-shot term selection. The representation-level feedback strategies developed in Chapters 5, 8 and 11 pursue a similar objective, tailoring the reformulated query to maximise downstream retrieval quality, yet operate directly in embedding space, reducing the latency inherent in iterative term selection while preserving the adaptability that DRQR exemplifies.

Improving Zero-Shot Retrieval Using Dense External Expansion

Wang et al. [215] tackle cross-domain retrieval by enriching queries with knowledge drawn from a large, *external* corpus rather than the unlabeled target collection. A dense retriever, DPR [83] or ColBERT [85], first gathers semantically related passages whose embeddings are then aggregated into a single expansion vector and fused with the original query representation. The resulting composite

embedding is submitted to the downstream retriever, substantially boosting effectiveness on zero-shot benchmarks without any fine-tuning. By sourcing feedback from a domain-rich repository, the method injects diverse lexical and conceptual cues that mitigate vocabulary mismatch and domain shift.

This strategy resonates with the thesis's argument that representation-level feedback can operate independently of in-domain labels. Chapters 5 and 11 further explore how external, or query-independent, signals can be pre-computed or prompt-generated to enhance low-resource retrieval while maintaining computational efficiency. Dense external expansion thus exemplifies a scalable path toward domain-adaptive query reformulation, complementing the vector-based and prompt-based frameworks advanced in this work.

CWPRF

Contrastive Weighting PRF (CWPRF) by Wang et al. [216] advances term selection by training a shared encoder with a contrastive objective that separates informative expansion terms from distractors within the pseudo-relevant set. Candidate tokens harvested via ColBERT [85] are embedded alongside the query, after which a loss function promotes terms that co-occur in semantically aligned contexts and suppresses off-topic alternatives. The resulting weights are applied when fusing expansions into the query, yielding higher precision than uniform pooling or frequency-based schemes and delivering consistent gains across diverse collections.

The method is germane to this thesis because it demonstrates that relevance cues can be distilled through representation learning rather than heuristic counts. CWPRF's dynamic importance weights parallel the vector-level adjustment strategies explored in Chapters 5, 8, and 11, where feedback signals are injected directly into dense query representations. By showing that contrastive objectives improve robustness to noisy feedback, CWPRF provides further empirical motivation for the thesis's central claim: feedback effectiveness is maximised when semantic alignment is learned in embedding space instead of imposed through static term-level heuristics.

OPRF

Offline PRF (OPRF) introduced by Wen et al. [220] relocates the computational burden of pseudo-relevance feedback to the indexing phase. Pseudo-relevant passages retrieved from a large source collection such as MS MARCO [151] are encoded once, averaged, and stored as expansion vectors. At inference time the system needs only fuse this pre-computed vector with the incoming query embedding and execute a single forward pass, eliminating repeated document encodings and multi-stage reranking. The design delivers retrieval quality comparable to online PRF while providing markedly lower latency and superior scalability.

OPRF highlights a central premise of this thesis: feedback effectiveness can be preserved even when its generation is decoupled from live queries. Chapter 11 extend the same idea by pre-computing passage-level features and prompt-driven embeddings that are reusable across queries, thereby achieving the dual goals of efficiency and robust semantic enrichment.

ReDE-RF

ReDE-RF by Jedidi et al. [77] advances PRF for cross-domain settings by synthesising a *feedback-aware* query embedding rather than fine-tuning the retriever. An out-of-domain dense model first gathers top-k passages; their transformer encodings (e.g., BERT [155]) are then aggregated, via simple averaging or learned weights, into a pseudo-query vector that is fused with the original embedding before the final retrieval step. The procedure transfers latent relevance cues from the feedback set without incurring domain-specific training, and empirical studies show sizeable zero-shot gains over baseline dense retrievers such as DPR [83] and ANCE [223].

Related to this thesis, ReDE-RF illustrates that representation-level feedback can serve as an *adapter* between domains, aligning with the central claim that semantic enrichment need not rely on supervised fine-tuning. Chapters 5 and 11 extend this principle by a completely zero-shot approach, thereby combining ReDE-RF's domain portability with improved efficiency and broader model coverage.

ColBERT-PRF

ColBERT-PRF [214] is the first framework to inject PRF into a *multi-vector* dense retriever, ColBERT [85], without any additional training or document re-encoding, making it a natural strong baseline for the experiments in Chapter 11. In ColBERT, a query q and document d are represented as sets of contextualised token embeddings, and relevance is scored via late interaction:

$$s(q,d) = \sum_{i=1}^{|q|} \max_{j=1,\ldots,|d|}, \phi_{q_i}^{\top} \phi_{d_j}, \tag{2.20}$$

where ϕ_{q_i} and ϕ_{d_j} denote the i-th query and j-th document token vectors, respectively.

Feedback Vector Induction: To incorporate PRF, the method retrieves the top-f_b pseudo-relevant documents $R(q)$, collects all their token embeddings:

$$\Phi(q, f_b) = \{\phi_{d_j} \mid d \in R(q)\}, \tag{2.21}$$

and clusters them with K-means:

$$\{v_1, \ldots, v_K\} = \mathrm{KMeans}\big(K, \Phi(q, f_b)\big). \tag{2.22}$$

Each centroid v_k is mapped to its nearest token vector in the retriever's vocabulary using FAISS [41]. The centroid's informativeness is estimated by the inverse document frequency σ_k of that nearest token. The f_e most discriminative centroids constitute the expansion set F_e.

Augmented Scoring: ColBERT-PRF augments Eq. (2.20) with these expansion embeddings:

$$s(q,d) = \sum_{i=1}^{|q|} \max_{j=1}^{|d|} \phi_{q_i}^{\top} \phi_{d_j} + \beta \sum_{(v_k,\sigma_k)\in F_e} \max_{j=1}^{|d|} \sigma_k v_k^{\top} \phi_{d_j}, \tag{2.23}$$

where β balances original and feedback signals. Because expansion is effected entirely in the embedding space, the model sidesteps polysemy drift and token repetition issues that can plague text-based expansion schemes such as CEQE [149].

Implications for This Thesis: ColBERT-PRF demonstrates that dense, context-sensitive feedback can be grafted onto an existing multi-vector retriever through purely inference-time operations, yielding large gains over the ColBERT backbone and remaining competitive with specialised neural re-rankers and PRF-enhanced sparse models. Its strong empirical performance makes it an indispensable baseline for evaluating the feedback methods proposed in this dissertation, which pursue similar goals, leveraging semantic feedback while minimising training and indexing overhead, across a broader range of retriever architectures.

ANCE-PRF

ANCE-PRF [234] extends the ANCE dual-encoder architecture [223] with a dedicated feedback encoder that absorbs first-stage pseudo-relevant documents and synthesises an enriched query vector. After the initial retrieval pass, the top-k passages $R = d_1, \ldots, d_k$ are paired with the original query q and fed through a transformer encoder:

$$\mathbf{e}_i = \text{Encoder}_\theta(q, d_i),\ i = 1, \ldots, k. \tag{2.24}$$

Self-attention across the concatenated tokens of q and d_i allows the model to highlight complementary evidence while down-weighting spurious content. The feedback vectors are then merged—via either learned attention pooling or simple averaging—to obtain the PRF-enhanced query embedding:

$$\mathbf{q}' = \text{Aggregate}(\{\mathbf{e}_i\}_{i=1}^{k}) \tag{2.25}$$

During training, $\mathbf{q}'$ is optimised with the same margin-based contrastive objective used by ANCE, but the negative set is drawn from the original document index, enabling the model to learn which feedback passages genuinely increase relevance. Because documents are never re-encoded, inference still involves a single inner-product lookup against the static ANCE [223] index, preserving the latency advantages of bi-encoder retrieval.

Substantial gains over baseline ANCE [223] and contemporary dense baselines with the largest improvements appearing on ambiguous or under-specified queries where additional context is most beneficial. Analysis shows the feedback encoder learns to amplify relevant signals introduced by useful passages, while attenuating generic or off-topic signals, a behaviour verified by attention heat-maps and ablation studies.

Implications for This Thesis: To validate the generalisability and robustness of ANCE-PRF, we reproduced ANCE-PRF in Chapter 4 and confirmed its substantial gains on the original ANCE [223] backbone but also exposed three obstacles to wider use. The feedback encoder is tightly coupled to the underlying ANCE dense retriever, even if fine-tuned with stronger dense retrievers, the improvements are modest. From the reproduction, effectiveness fluctuates sharply with minor changes to training hyperparameters, indicating considerable sensitivities. Finally, the input limit inherited from the underlying transformer-based neural language models cap the number of feedback passages that can be processed, truncating useful signals on long-form PRF queries. These constraints motivate the model-agnostic, embedding-level strategies developed in Chapters 5 and 11, which operate in a

complete zero-shot manner, thus avoiding encoder coupling, hyperparameter fragility, and input-length constraints.

GRF

GRF, introduced by Mackie et al. [139], leverages the generative capabilities of LLMs to produce synthetic documents or summaries relevant to the query. Rather than selecting expansion terms from retrieved documents, GRF synthesizes new text based on the query itself and uses these generations to construct a relevance feedback model.

The GRF framework involves three main components: (1) generation of synthetic documents using a prompt-tuned LLM, (2) estimation of a feedback term distribution over the generated texts, and (3) interpolation of this distribution with the original query model. Let $G = \{g_1, g_2, ..., g_n\}$ denote the set of generated passages conditioned on query Q. A term distribution is computed over these passages to form the generative relevance model:

$$P(w \mid G) = \sum_{g \in G} P(w \mid g) \cdot P(g \mid Q) \tag{2.26}$$

Here, $P(w \mid G)$ is the term probability in the generated passage g, typically based on maximum likelihood estimation or smoothed counts, and $P(g \mid Q)$ is treated as uniform or proportional to generation confidence. The final query model is then interpolated with the original query model $P(w \mid Q)$:

$$P_{\text{GRF}}(w \mid Q) = \lambda P(w \mid G) + (1 - \lambda) P(w \mid Q) \tag{2.27}$$

This updated model can be used within any retrieval architecture, sparse, dense, or learned sparse, by scoring documents using language modeling techniques or computing similarity between vector representations. Early experiments show that GRF substantially increases recall by injecting external knowledge absent from the target corpus [217].

Hybrid Integration with Classical PRF: In follow-up work, Mackie et al. [138] demonstrates that GRF can be combined with traditional PRF in both sparse and dense settings. Let $\mathbf{q}_0$ denote the initial query representation, $R = \{d_1, d_2, \ldots, d_k\}$ the top-k retrieved documents, and G the generated passages. Using the same encoder as the base retriever, the framework constructs:

$$\mathbf{q}_{\text{Gen}} = \frac{1}{|G|} \sum_{g \in G} \text{Encoder}(g), \quad \mathbf{q}_{\text{PRF}} = \frac{1}{k} \sum_{i=1}^{k} \text{Encoder}(d_i), \tag{2.28}$$

and fuses them via:

$$\mathbf{q}_{\text{final}} = \lambda \mathbf{q}_{\text{Gen}} + (1 - \lambda) \mathbf{q}_{\text{PRF}} \tag{2.29}$$

or rank-level aggregation such as Reciprocal Rank Fusion (RRF) when separate retrieval runs are performed. Evaluations on BM25 [178], SPLADE [46], and ANCE [223] backbones show that the generative and corpus-based signals are complementary, delivering additive gains over either component alone.

Limitations and Implications for This Thesis: Despite its effectiveness, GRF generates feedback *at inference time*. Each query prompts multiple LLM calls, introducing substantial latency and computational cost, and the resulting features are inherently query-dependent, limiting reuse across sessions. In Chapter 11 we therefore employ GRF as a strong baseline while investigating prompt-based PRF variants that pre-compute feedback to reduce online overhead. The comparison highlights a core objective of this dissertation: to achieve the semantic richness of LLM-generated feedback while delivering the efficiency and model-agnostic flexibility required for large-scale neural retrieval.

Query2Doc

Query2Doc, proposed by Wang et al. [209], replaces corpus-based feedback with text *generated* on demand by a LLM. Given only the user query q, an instruction-tuned LLM (e.g., GPT-3 [14]) is prompted with a template such as *"Write a concise passage that fully answers the following query:"*. The model returns a coherent paragraph d' whose lexical and semantic content is assumed to approximate what an ideal, highly relevant document might contain. Because no first-pass retrieval is required, the approach avoids the low-recall situation of traditional PRF on ambiguous or under-specified queries.

Integration with Sparse and Dense Retrievers: To preserve the influence of q when the generated passage is long, Query2Doc repeats the original query string n times and concatenates the result with d':

$$q^{+}_{\text{sparse}} = \text{concat}(q \times n, d'), \tag{2.30}$$

where n is tuned (typically 5–10) to match the average length of d'. For dense models the two texts are merged at the token level with a special separator:

$$q^{+}_{\text{dense}} = [q; [\text{SEP}]; d'] \tag{2.31}$$

The resulting input q^{+} is fed directly into the frozen query encoder of a retriever such as DPR [83] or SimLM [208]; no parameter updates or document re-encodings are required, making the method model-agnostic. Qualitative inspection shows that generated passages often introduce synonyms, named entities, and explanatory context missing from the original query, thereby enriching the semantic match space.

Limitations: The approach incurs the latency and monetary cost of multiple autoregressive LLM calls at inference time, a bottleneck for real-time or high-throughput systems. Because each d' is *query-specific*, there is no opportunity to cache or reuse feedback across sessions. Moreover, generations may contain hallucinated facts or off-topic digressions [192], potentially polluting the expanded query. The authors mitigate this risk with prompt constraints and length caps, yet the issue remains salient for safety-critical applications.

Implications for This Thesis: In Chapter 11 we adopt Query2Doc as a baseline when evaluating prompt-based PRF. Its strengths, independence from first-pass retrieval and ease of integration with both sparse and dense models, provide a meaningful contrast to our proposed framework, which

seeks comparable semantic enrichment while amortising generation cost and reducing reliance on query-dependent features.

The literature reveals a clear trajectory: from rigid sparse expansion (Rocchio/RM3) to powerful but inefficient neural generative models (GRF/Query2Doc). However, a critical gap remains: how to achieve the semantic power of neural feedback without the prohibitive cost of generative inference or full-model retraining. This thesis directly targets this efficiency-effectiveness gap. Part 1 introduces vector-based methods to bypass retraining; Part 2 develops lightweight adapters to replace heavy encoders; and Part 3 leverages offline LLM generation to solve the latency bottleneck of generative feedback.

2.5 Datasets

Throughout this thesis, we evaluate the proposed PRF methods on a range of standard information retrieval benchmarks. The selection of datasets for each chapter is driven by the specific research questions being addressed, the nature of the feedback mechanisms under investigation, and the evolution of community standards for evaluating neural retrieval systems. A summary of the datasets used across chapters is provided in Table 2.2.

2.5.1 Standard Retrieval Benchmarks

TREC Deep Learning Track (DL) 2019 and 2020: The DL 2019 [27] and DL 2020 [26] datasets serve as the consistent "anchor" benchmarks across all experimental chapters (Chapters 3 – 11). Derived from the MS MARCO passage ranking corpus [151] (MS MARCO Passage V1), they contain dense, graded relevance judgments that are essential for accurately measuring the subtle improvements in ranking quality introduced by feedback mechanisms. Their status as the standard benchmarks for neural ranking enables easier comparison of the strategies proposed in this thesis against established baselines.

MS MARCO Passage V1: The MS MARCO training set is utilized in chapters involving supervised learning to train feedback-aware query encoders. The development set (Dev) is primarily used for hyperparameter tuning and validation, ensuring that test results on TREC DL remain unbiased.

2.5.2 Specialized and Adversarial Benchmarks

In the initial parts of the thesis (Chapters 3 and 5), we investigate the robustness of foundational PRF mechanisms using specialized datasets designed to challenge retrieval systems:

- **DL HARD:** A subset of queries from TREC DL known to be difficult for neural models. This dataset is included to stress-test feedback strategies under adversarial conditions where the initial retrieval quality is often poor. [137].

Table 2.2: Overview of datasets used across different chapters of this thesis. TREC DL 2019 and 2020 serve as the consistent anchor benchmarks, while other datasets are selected based on the specific requirements of the research questions in each Part.

Dataset	Chapters Used	Role and Rationale
Anchor Benchmarks		
TREC DL 2019/2020	3, 4, 5, 6, 7, 8, 9, 10, 11	**Primary Evaluation:** Selected for high judgment density to allow consistent comparison across all proposed feedback strategies.
Training and Validation		
MS MARCO (Passage)	4, 7, 8, 9	**Supervision:** Used for training learned feedback encoders and performing hyperparameter tuning (Dev set).
Adversarial & Specialized Benchmarks		
DL HARD	3, 5	**Stress Test:** Evaluating feedback robustness on hard queries, stress-test. Excluded in later chapters to isolate algorithmic effects from adversarial baseline failures.
TREC CAsT 2019	3, 5	**Context Test:** Evaluating feedback mechanisms under conversational context shifts.
WebAP	3, 5	**Length Robustness:** Testing performance on documents with varying lengths.
Generalization Benchmarks		
BEIR	10, 11	**Zero-Shot Transfer:** Standard benchmark for evaluating LLM-based retrieval, used to assess out-of-domain generalization capabilities.

- **TREC CAsT 2019:** A conversational search dataset used to evaluate how feedback mechanisms handle context-dependent queries and topic shifts [35].
- **WebAP:** A dataset with diverse passage lengths, used to test robustness against input length variation in reranking tasks. [84].

However, as we demonstrate in our empirical analysis, the adversarial nature of DL HARD often leads to a violation of the fundamental pseudo-relevance assumption that top-ranked passages are mostly relevant. Therefore, in other chapters we focus on the stable environment of TREC DL 19/20 and exclude DL HARD to rigorously isolate specific algorithmic effects without the confounding variables of adversarial failure.

2.5.3 Generalization Benchmarks

In the final part of the thesis (Chapters 10 and 11), the focus shifts to Large Language Models (LLMs) and zero-shot generalization. To align with current evaluation standards in the LLM retrieval literature, we adopt the BEIR benchmark [200]. BEIR is a diverse collection of datasets covering various domains (e.g., biomedical, financial, scientific) and is specifically designed to test the zero-shot transfer

capabilities of retrieval models. Its inclusion allows us to rigorously evaluate whether generative feedback mechanisms can generalize beyond the training distribution of MS MARCO.

Part 1

Pseudo-Relevance Feedback in Transformer-Based and Dense Retrieval Models

Introduction to Part 1

In this part of the thesis, we investigate our first main research question:

RQ1: How can Pseudo-Relevance Feedback be integrated into neural models?

The widespread adoption of neural architectures, particularly transformer-based models [37, 119], in information retrieval has attracted interest in how classical retrieval techniques, such as Pseudo-Relevance Feedback (PRF) [25], can be effectively incorporated into these modern systems. While PRF mechanisms have been proved to be effective in traditional IR pipelines [16, 126, 179, 196, 235], its integration into neural retrieval models presents new challenges and opportunities, especially in balancing effectiveness with computational efficiency.

We begin by exploring how classical text-based PRF [104] can be adapted for transformer-based rerankers. Chapter 3 presents a framework that applies Text-based PRF to expand queries with a classic BM25+BERT [113, 155] reranking pipeline, where BM25 [177] is the initial retriever, BERT [37] is a pre-trained model and further fine-tuned as the reranker (Section 2.2.2 in Chapter 2), the same BERT model is then used for a second reranking step with the expanded query to further improve reranking effectiveness. Although this approach offers consistent effectiveness improvements, due to the need for the second reranking step with long input sequences through large transformer encoders, it incurs huge computational cost. This limits its practicality for real-time or resource-constrained applications.

To overcome this, Chapter 4 investigates dense retrievers based on bi-encoder architectures, which support efficient retrieval using pre-computed passage embeddings [102]. We conduct a reproducibility study of ANCE-PRF [234], a method that integrates PRF into dense retrieval by fine-tuning the query encoder to incorporate feedback through concatenated query-passage pairs. Although ANCE-PRF avoids reranking and offers strong effectiveness, it remains tightly coupled to its specific model design and training procedure, making it less generalizable and harder to deploy across systems.

Chapter 5 introduces a lightweight and model-agnostic alternative: Vector-based PRF (VPRF) [104]. This framework constructs a dense representation of feedback information and reformulates the query embedding using vector-based operations, such as averaging or Rocchio [179]-style interpolation. Unlike text-based PRF, VPRF manipulates the dense representations (vectors) directly and requires no re-encoding or model retraining, and it can be applied to any dense retriever in a plug-and-play manner. To facilitate reproducibility and integration, we also implement VPRF in the Pyserini [101, 112] toolkit.

Finally, Chapter 6 investigates hybrid feedback strategies that combine sparse and dense retrieval signals within the PRF process [100]. We propose and evaluate three interpolation variants: Pre-PRF, Post-PRF, and Both-PRF, which integrate sparse retriever scores either during candidate selection, query reformulation, or both. Our empirical results show that these hybrid strategies significantly enhance performance, particularly when leveraging learned sparse representations such as uniCOIL [111].

Together, this part demonstrate that PRF remains highly effective when adapted to neural retrieval frameworks. From BERT-based reranking to dense retrievers and sparse-dense interpolation, we systematically explore multiple paths for integrating PRF into modern neural retrieval pipelines. These

findings establish a solid foundation that guides the subsequent parts of the thesis, where we turn to the challenges of efficiency, generalization, and LLM-driven feedback mechanisms.

Chapter 3

Text-Based Pseudo-Relevance Feedback For Neural Rerankers

In this chapter, we aim to address the following sub-research questions:

RQ1.1: How can Pseudo-Relevance Feedback be adapted for neural models?

RQ1.2: What are the challenges and limitations of integrating Pseudo-Relevance Feedback into neural models?

To investigate how PRF can be integrated with neural models and the challenges and limitations involved, we begin with a straightforward strategy: concatenating the query text with PRF passages to form an expanded query input for a deep language model (e.g., BERT [37], other models such as RoBERTa [119], query likelihood models [40, 246, 248] can be applied as well). However, this approach faces two key challenges: (i) the concatenated input often exceeds model input length limits [37, 39, 155, 157, 230], and (ii) it incurs high computational costs due to additional model inferences at query time [37, 118, 163, 218]. To address the input length constraint, we propose three text partitioning strategies that segment the input into manageable chunks. To aggregate results across partitions, we explore three score fusion methods: Average, Borda, and Max.

We evaluate our approach on TREC DL 2019/2020 [26, 27], TREC CAsT [35], WebAP [84], and DL HARD [137]. Our method significantly improves over baselines on TREC DL 2019 but yields mixed or negative results on the other benchmarks. These findings suggest that Text-based PRF is most effective when initial rankings contain more relevant initial results. In contrast, noisy initial results lead to query drift and performance degradation. Furthermore, our analysis highlights a core limitation: Text-based PRF roughly doubles inference cost (Section 3.3.5), making it impractical for latency-sensitive applications.

3.1 Text-Based Pseudo-Relevance Feedback

BERT is a computationally expensive model and is therefore typically used as a reranker [5, 52, 56, 113] that operates over a limited set of top-ranked candidates (commonly the top 1,000 results) produced by

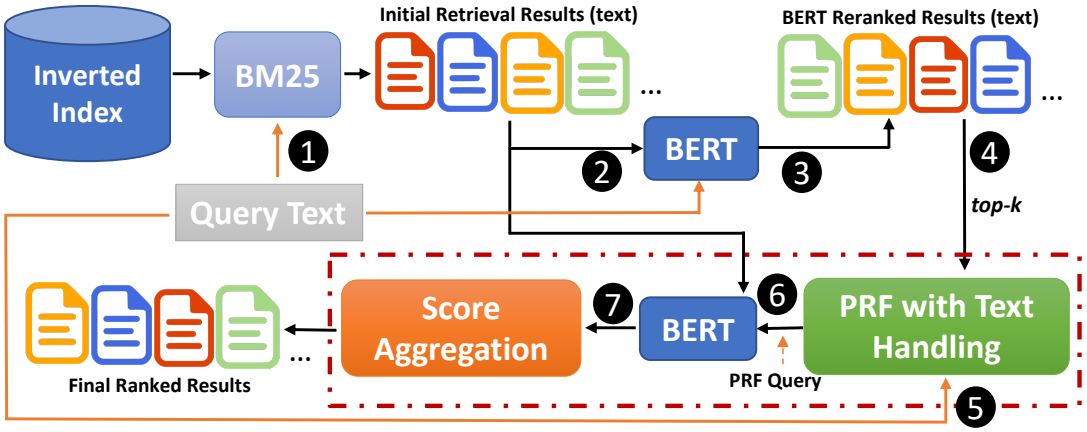


Figure 3.1: The proposed architecture for integrating Text-based Pseudo-Relevance Feedback with BERT reranker. The initial retriever is a traditional bag-of-words BM25.

an initial retrieval stage. In our approach, we integrate the Text-based PRF signal directly with the BERT reranker to refine the final rankings. Unlike traditional PRF methods such as RM3 [125], which extract high-weighted terms from feedback passages to reformulate queries, prior work by Padaki et al. [163] showed that appending such keyword-based expansions can negatively impact the effectiveness of fine-tuned BERT rerankers. To avoid this degradation, we instead utilize the full textual content of the feedback passages to construct Text-based PRF queries, preserving the rich semantic context necessary for BERT to operate effectively.

Figure 3.1 illustrates the architecture we used for integrating Text-based PRF signals into the BERT reranking pipeline. The upper section of the figure represents the standard two-stage reranking workflow (Steps ❶ – ❹), while the lower section demonstrates the integration of the Text-based PRF component (Steps ❺-❼). The process begins with an initial retrieval stage using a traditional BM25 bag-of-words model (❶), which retrieves an initial ranked list of passages from the inverted index. This list, along with the original query, is then passed to a standard BERT reranker [155] for initial scoring (❷). The top-k passages from the reranked list are selected as feedback passages (❹) after being mapped back to their raw text form (❸). These feedback passages are then combined with the original query text to generate Text-based PRF queries (❺), which are again scored with candidate passages using BERT (❻). Finally, the scores produced across the Text-based PRF queries are aggregated to produce the final ranking of candidate passages (❼). The two central components of this pipeline, Text Handling and Score Aggregation, are discussed in detail in the following sections.

3.1.1 Text Handling in Text-Based PRF

To address the challenge that longer Text-based PRF queries may exceed BERT's input limit, we adopt three text handling strategies in our Text-based PRF framework:

Concatenate and Truncate (CT)

To construct the Text-based PRF query, we concatenate the original query text with the top-k *feedback* passage texts, using a space ($\_$) as the delimiter. If the resulting text exceeds 256 tokens, it is truncated to that length [148]. Since BERT accepts a maximum input of 512 tokens, we allocate 256 tokens to the refined query (Q_{new}) and reserve the remaining 256 tokens for each of the to-be-reranked *candidate* feedback passages ($\{p_1, ..., p_k\}$). The final query-passage pair is then passed to the BERT reranker for scoring. The query is constructed as follows:

$$Q_{new,l<=256} = \lceil Q_{original} + \_ + p_1 + ... + \_ + p_k \rceil_{256} \tag{3.1}$$

where Q_{new} and $Q_{original}$ represent the new query text and the original query text, respectively. $l <= 256$ represents the input size limit enforced, which is achieved by truncating the sequence (denoted with $\lceil . \rceil_{256}$). $p_1, ..., p_k$ represent the top-k feedback passages from the BERT reranker. $\_$ is the space in between.

Concatenate and Aggregate (CA)

We generate k new queries by concatenating the original query text with each of the top-k feedback passages, separated by a space ($\_$). Each new query is passed through BERT to produce a separate reranked list, resulting in k scores for each candidate passage. These scores are then aggregated using score aggregation methods (See Section 3.1.2) to compute the final ranking, excluding the original query's ranked list. The new queries are generated as follows:

$$\begin{aligned} Q_{1,new} &= Q_{original} + \_ + p_1 \\ &... \\ Q_{k,new} &= Q_{original} + \_ + p_k \end{aligned} \tag{3.2}$$

where $Q_{1,new}, ..., Q_{k,new}$ represent the k new queries. $Q_{original}$ represents the original query text. $p_1, .., p_k$ represent the top-k feedback passage texts. $\_$ is the separation token in between.

Sliding Window (SW)

In this approach, the top-k feedback passage texts are first concatenated into a single block, then segmented into j overlapping partitions using a sliding window [180, 197]. The window size and stride are adjusted based on passage lengths specific to each dataset [31]. Each partition is then concatenated with the original query to form j new queries, which are processed following the same steps as in the Concatenate and Aggregate method:

$$p_1 + ... + p_k \xrightarrow{SW} p_1, ..., p_j \tag{3.3}$$

where $p_1, ..., p_k$ represent the top-k feedback passage texts, SW represents the sliding window mechanism, $p_1, ..., p_j$ represent the j partitions. Similar to the CA approach, the set of j new queries is generated using Eq. 3.2.

Note that after generating each new query, the query/passage pair may exceed the BERT input size limit for the CT approach. Under this situation, if the length of the new query exceeds 256, we truncate the new query down to be of length 256. For CA and SW approaches, we also applied the same methodology to guarantee all the new queries are below the length of 256.

3.1.2 Score Aggregation in Text-Based PRF

CA and SW text-handling approaches generate k and j scores per candidate passage, respectively. To estimate a final score for each candidate passage, we consider the following aggragation methods.

Average

The final score is aggregated by calculating the mean of all scores:

$$s_{final} = Avg(S_1 + S_2 + ... + S_k) \tag{3.4}$$

where s_{final} represents the final ranking score for each candidate passage, and $S_1, ..., S_k$ represent the k ranking scores for each candidate passage based on each of the k new queries. For the rest of this chapter, we refer to this method as Text-Average, represented by T-A for brevity.

Max

The final score is aggregated by taking the highest score per candidate passage:

$$s_{final} = Max(S_1, S_2, ..., S_k) \tag{3.5}$$

where s_{final} represents the maximum score for each candidate passage, and $S_1, ..., S_k$ represent the k ranking scores for each candidate passage based on each of the k new queries. For the rest of this chapter, we refer to this method as Max, represented by M for brevity.

Borda

The final score is aggregated by using the Borda voting algorithm [8, 135]. The score of a candidate passage w.r.t a ranked list is the number of candidate passages in the ranked list that are ranked lower. Scores are summed over ranked lists as follows:

$$s_{final} = \sum_{L_i : p \in L_i} \frac{n - r_{L_i}(p) + 1}{n} \tag{3.6}$$

where L_i represents the $i\text{-}th$ ranked list produced using the $i\text{-}th$ new query, p represents the candidate passage, r is the rank of the candidate passage, and n represents the number of candidate passages in the ranked list. For the rest of this chapter, we refer to this method as Borda, represented by B for brevity.

3.2 Empirical Evaluation

To guide our empirical investigation of PRF integration with neural rerankers, we pose the following research questions:

RQ1.1.1: What is the impact of PRF depth on the effectiveness of Text-based PRF with neural rerankers?

RQ1.1.2: What is the impact of text handling techniques on the effectiveness of Text-based PRF with neural rerankers?

RQ1.1.3: What is the impact of score aggregation methods on the effectiveness of Text-based PRF with neural rerankers?

RQ1.1.4: What is the overall effectiveness of Text-based PRF models on the task of reranking?

RQ1.1.5: What is the impact of Text-based PRF models on the efficiency of reranking with neural rerankers?

RQ1.2.1: What are the challenges and limitations when integrating Text-based PRF into neural rerankers?

The implementation of Text-based PRF and the baselines evaluated in this Chapter is available at: `https://github.com/ielab/Neural-Relevance-Feedback-Public/tree/master/Text_Based_PRF`.

3.2.1 Datasets and Evaluation Metrics

Our experiments use the TREC Deep Learning Track Passage Retrieval Task 2019 [27] (DL 2019) and 2020 [26] (DL 2020), DL HARD [137], the TREC Conversational Assistance Track 2019 [35] (CAsT 2019), and the Web Answer Passages (WebAP) [84]. We evaluate Text-based PRF using MAP, nDCG@1, 3, 10, Success@1, and Reciprocal Rank (RR)[1]. These metrics are standard for BERT-based reranking models and widely used in prior work on the benchmark datasets, enabling direct comparison with existing and future results. For the TREC DL 2020 dataset, we follow the organizers' guidelines and binarize relevance labels at level 2 across all metrics. Statistical significance is assessed using a two-tailed paired t-test with Bonferroni correction.

3.2.2 Baselines

We consider the following baselines:

- `BM25`: traditional first stage retriever, implemented using the Anserini toolkit [229] with its default settings ($k_1 = 0.9$, $b = 0.4$).

[1] If no relevant passage is retrieved for a query within the standard cut-off (1,000), RR is assigned a value of 0.

Table 3.1: Window size and stride size of the Sliding Window PRF approach for each dataset.

	window size	stride
TREC DL 2019	65	32
TREC DL 2020	65	32
TREC CAsT 2019	69	34
WebAP	75	37
DL HARD	65	32

- `BM25+RM3`: RM3 pseudo relevance feedback method [1] on top of BM25, as implemented in Anserini. We use this approach as a representative bag-of-words PRF method, since previous research has found alternative bag-of-words PRF approaches achieve similar effectiveness [146]. We note that BM25+RM3 is a standard baseline for MS MARCO and TREC DL. We use default settings with 10 feedback terms, 10 feedback documents, original query weight 0.5.

- `BM25+BERT (BB)`: A common two-stage reranker pipeline, first proposed by Nogueira and Cho [155], where the initial stage is BM25, and BERT is used to rerank the results from BM25. BERT is fine-tuned on MS MARCO Passage Retrieval Dataset [151]. In all of our experiments, we use the 12 layer uncased BERT-Base provided by Nogueira and Cho [155], unless stated otherwise, and we simply refer to it as BERT. In Section 3.3.5 we also use BERT-Large for the efficiency analysis.

3.2.3 Applying Text-Based PRF to Rerankers

We refer to our Text-based pseudo relevance feedback for reranking approach as BB+PRF, where BB represents BM25+BERT. For the Sliding Window approach, we use the average passage length as the window size, and half of the window size as the stride. Details of the Sliding Window parameters for each dataset are shown in Table 3.1. We experiment by using the top $k = 1, 3, 5, 10, 15, 20$ passages as pseudo relevance feedback.

3.2.4 Efficiency Experiments

To measure the runtime of each method, we run our experiments on a Unix-based server with the Intel(R) Xeon(R) Gold 6132 CPU @ 2.60GHz for BM25 and BM25 + RM3. For dense retrievers and our approaches, we use a Unix-based server equipped with a single Tesla V100 SMX2 32GB GPU.

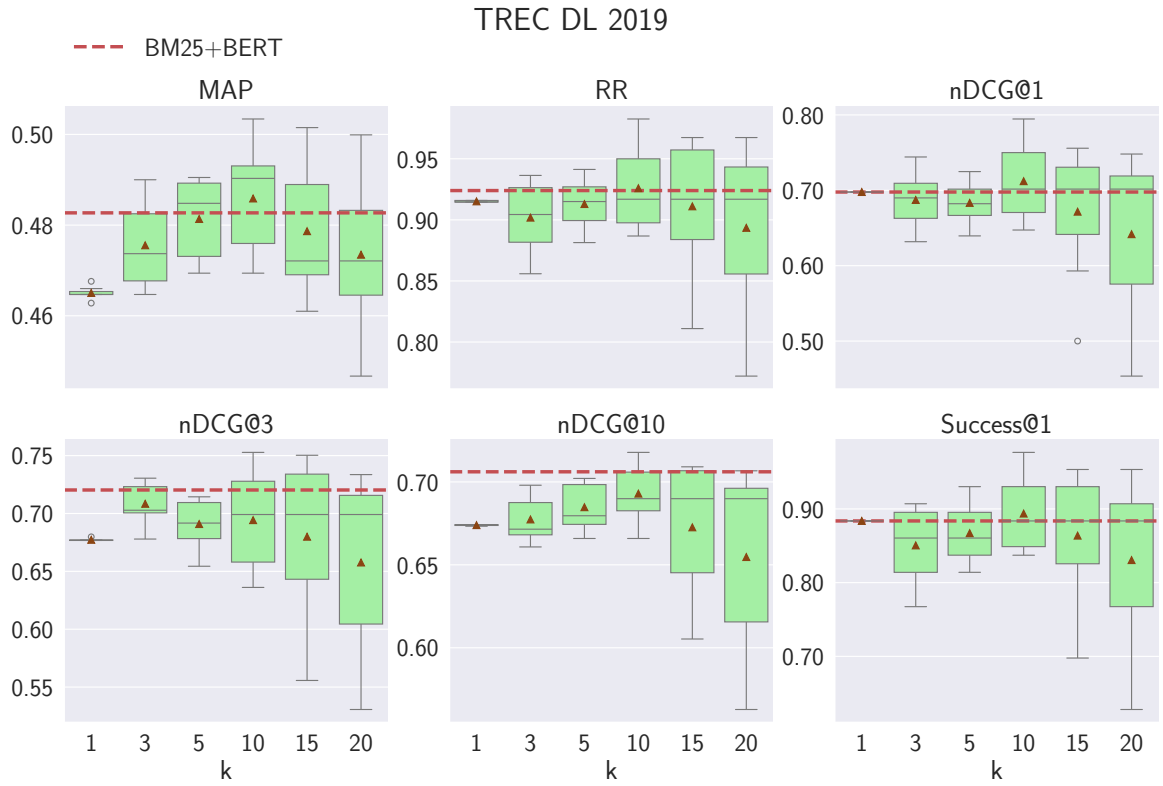


Figure 3.2: Impact of PRF depth on the effectiveness (y-axis) of BM25+BERT+PRF(BB+PRF) for the task of reranking on TREC DL 2019, *k* represents different PRF depths. Baseline BM25+BERT(BB) is marked with a dashed red line.

3.3 Results

3.3.1 Impact of PRF Depth on the Effectiveness of PRF with Neural Rerankers

RQ1.1.1 examines how PRF depth impacts Text-based PRF when integrated with neural rerankers. To address this, we vary the number of top-*k* passages while displaying the distribution of results over other parameters (text handling and score estimation).

Results of *Text-based* PRF (BB+PRF) for reranking are shown in Figure 3.2 – 3.6. For TREC DL 2019 (Figure 3.2), increasing the PRF depth (i.e. the number of top-ranked passages used as pseudo feedback passages) leads to marginal improvements in most evaluation metrics, particularly MAP and RR. These gains indicate that incorporating a limited amount of feedback can help the reranker capture more relevant information when the initial retrieval results are of relatively high quality. However, improvements degrade quickly beyond shallow depths, and certain metrics such as nDCG@10 and nDCG@3 show little to no benefit from deeper feedback.

In contrast, for the remaining datasets: TREC DL 2020 (Figure 3.3), TREC CAsT (Figure 3.4), WebAP (Figure 3.5), and DL HARD (Figure 3.6), increasing PRF depth generally leads to a decline in effectiveness across all reported metrics. For these datasets, none of the PRF configurations produce meaningful or consistent improvements over the baseline BERT-based reranker without feedback. This suggests that deeper feedback introduces more noise than useful signals, amplifying issues such as

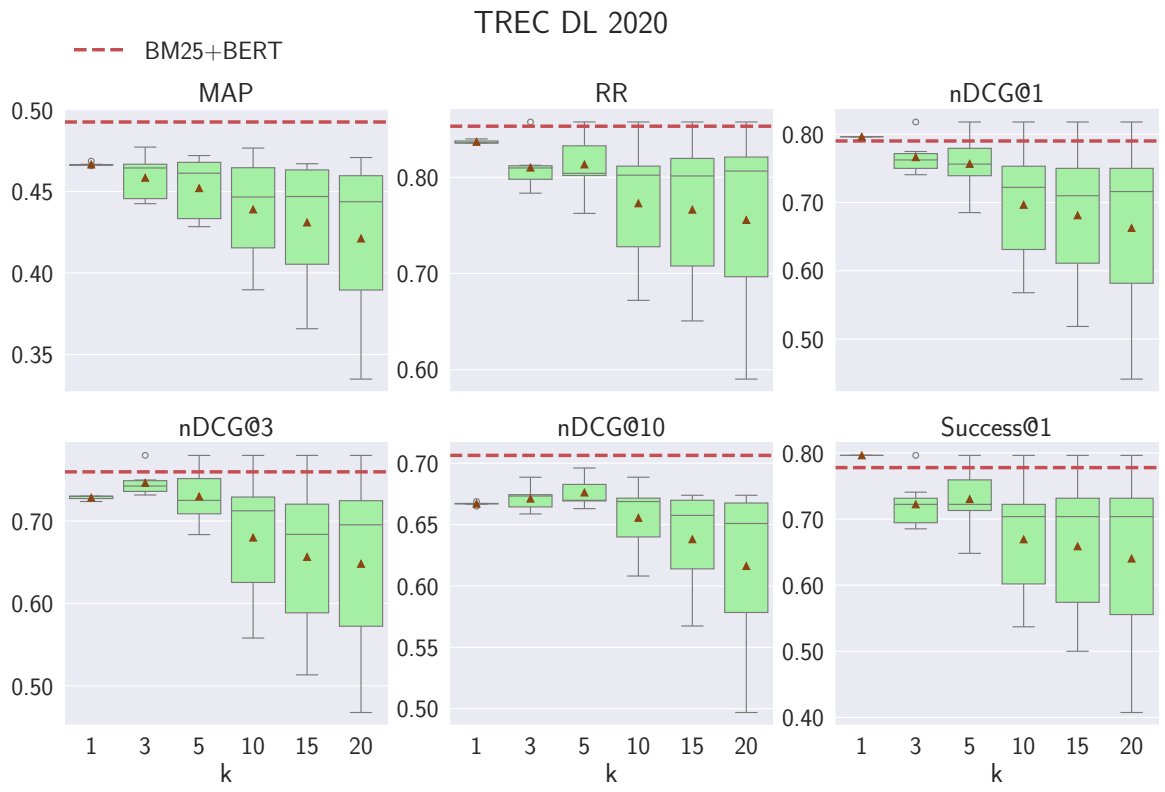


Figure 3.3: Impact of PRF depth on the effectiveness (y-axis) of BM25+BERT+PRF(BB+PRF) for the task of reranking on TERC DL 2020, *k* represents different PRF depths. Baseline BM25+BERT(BB) is marked with a dashed red line.

query drift. The impact is especially observable in datasets with ambiguous or under-specified queries (e.g., DL HARD), where the initial top-ranked passages are less reliable as feedback sources.

To summarize, increasing PRF depth tends to harm the effectiveness of Text-based reranking models. While shallow depths may offer slight improvements under favorable conditions (as observed in TREC DL 2019), deeper feedback typically degrades performance due to the incorporation of irrelevant or misleading information into the reformulated query. These findings highlight a key limitation of Text-based PRF methods: their vulnerability to noise in the feedback set, particularly when integrated with neural rerankers.

3.3.2 Impact of Text Handling Techniques on the Effectiveness of Text-Based PRF with Neural Rerankers

RQ1.1.2 investigates the impact of different text handling techniques on the effectiveness of PRF when integrated with neural rerankers. This question is motivated by the input length limitations of transformer-based models, which restrict the amount of feedback text that can be incorporated into the reranking process. To explore this, we vary the text handling strategies used to construct the reformulated query, while analyzing the distribution of results across different PRF depths and score aggregation methods. Specifically, we evaluate three techniques: Concatenate and Truncate (CT), which directly joins the query and feedback passages but discards overflow tokens; Concatenate and Aggregation (CA), which forms multiple full-length inputs and aggregates their scores; and Sliding

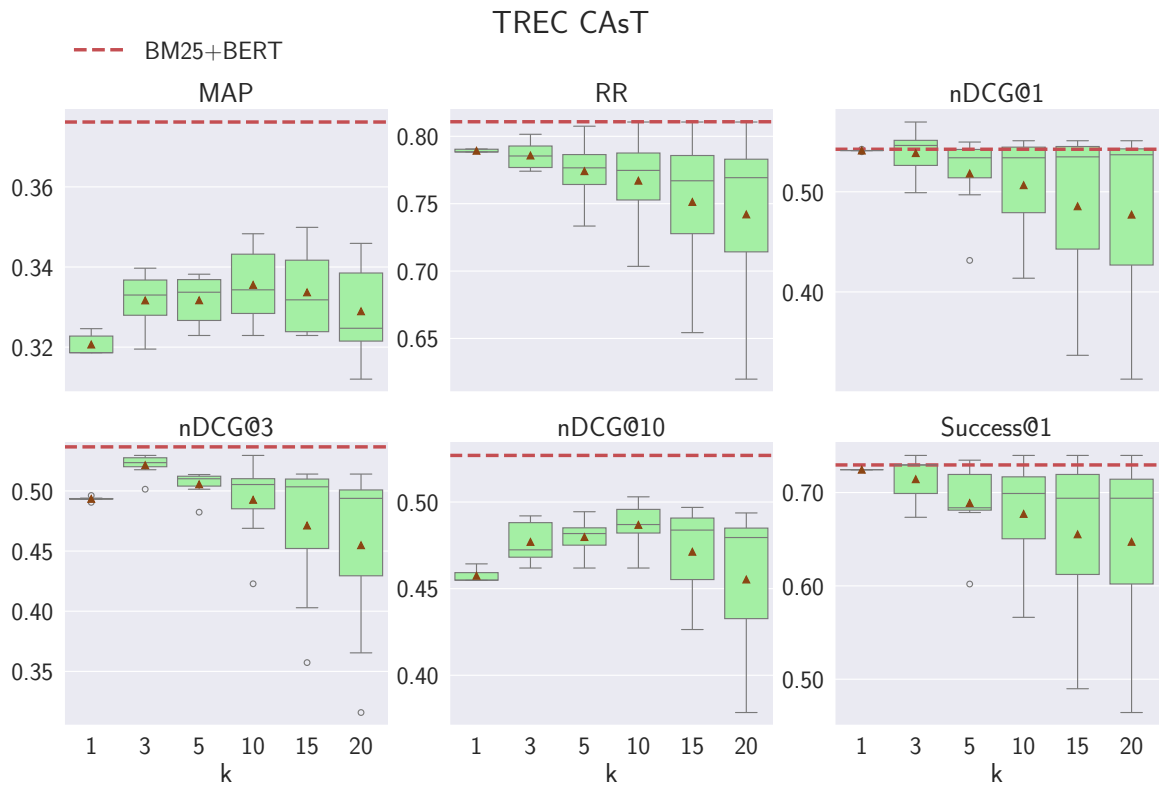


Figure 3.4: Impact of PRF depth on the effectiveness (y-axis) of BM25+BERT+PRF(BB+PRF) for the task of reranking on TREC CAsT, *k* represents different PRF depths. Baseline BM25+BERT(BB) is marked with a dashed red line.

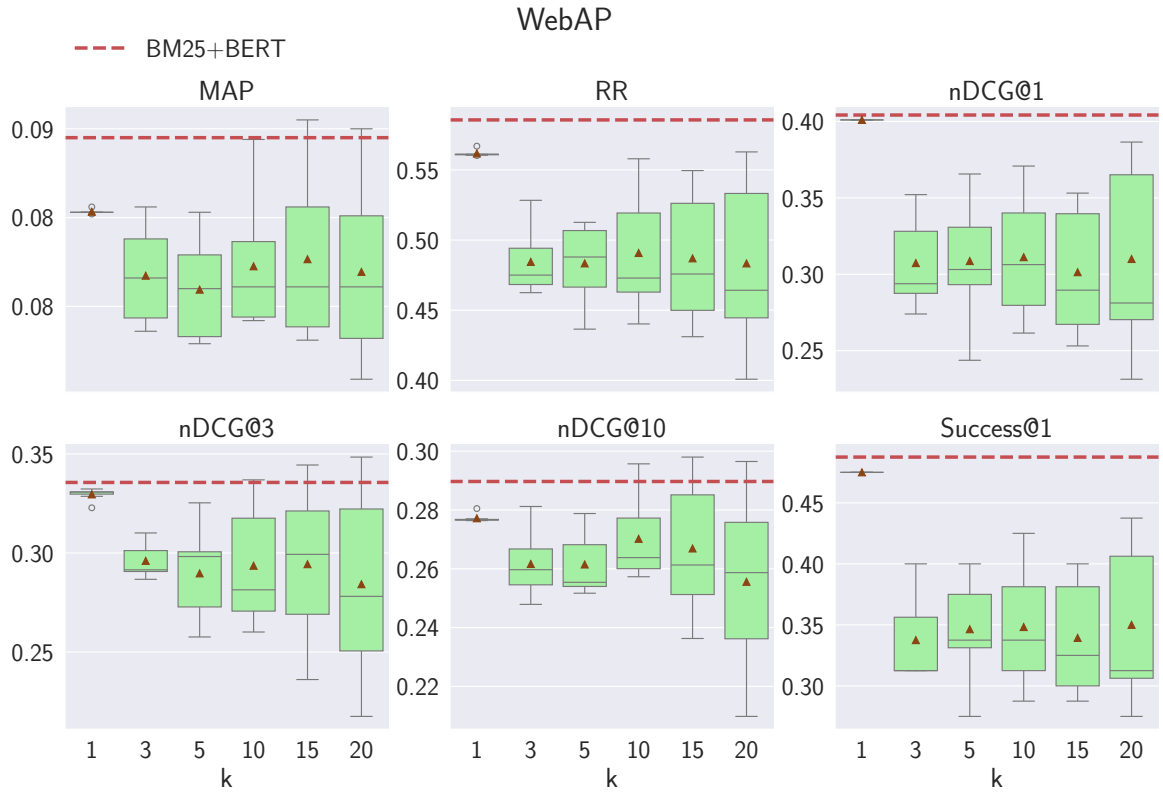


Figure 3.5: Impact of PRF depth on the effectiveness (y-axis) of BM25+BERT+PRF(BB+PRF) for the task of reranking on WebAP, *k* represents different PRF depths. Baseline BM25+BERT(BB) is marked with a dashed red line.

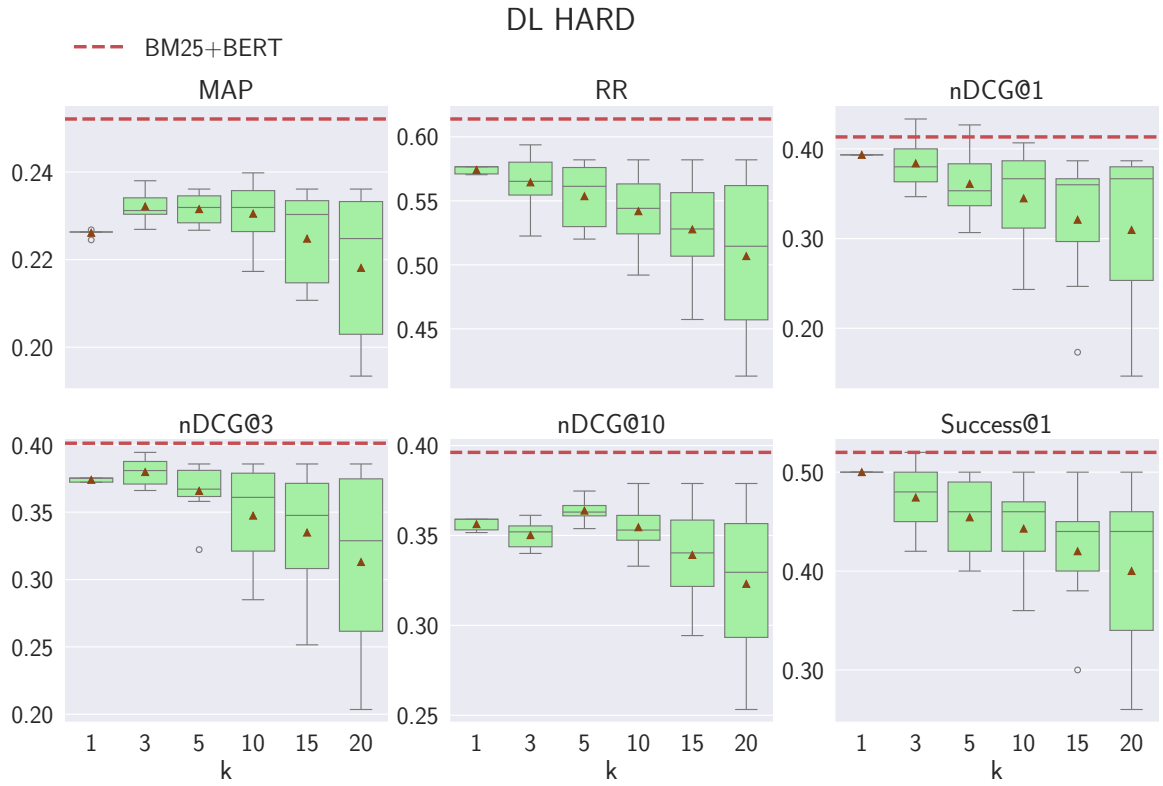


Figure 3.6: Impact of PRF depth on the effectiveness (y-axis) of BM25+BERT+PRF(BB+PRF) for the task of reranking on DL HARD, *k* represents different PRF depths. Baseline BM25+BERT(BB) is marked with a dashed red line.

Window (SW), which creates overlapping text segments to preserve more context. These strategies reflect different trade-offs between information retention and input constraints, and their comparative effectiveness provides insight into how best to integrate PRF signals under architectural limitations.

Results are shown in Figure 3.7 – 3.11. For TREC DL 2019 (Figure 3.7), the Concatenate and Aggregation (CA) method substantially improves MAP, RR, and nDCG@1. This suggests that on datasets with high-quality initial retrieval, preserving the full context of feedback passages is crucial. The CT method likely underperforms here because it truncates useful relevance signals from the feedback set, whereas CA allows the model to vote based on the complete evidence from all top-*k* passages.

However, this advantage diminishes on harder datasets. For TREC DL 2020 (Figure 3.8), CA loses its advantage, and the Sliding Window (SW) method provides only marginal improvements. DL 2020 contains more challenging queries where relevant information might be buried within largely non-relevant passages. SW likely performs slightly better here because it segments the text, potentially isolating relevant snippets from surrounding noise, whereas CA forces the model to process the entire noisy passage.

In the case of TREC CAsT 2019 (Figure 3.9), no text handling technique yields significant gains. The CT method performs comparably to the baseline in RR, which is notable. CAsT involves conversational queries where context sensitivity is high. The CA method, by treating all feedback passages as independent evidence, may introduce too much diverse or conflicting context, leading to

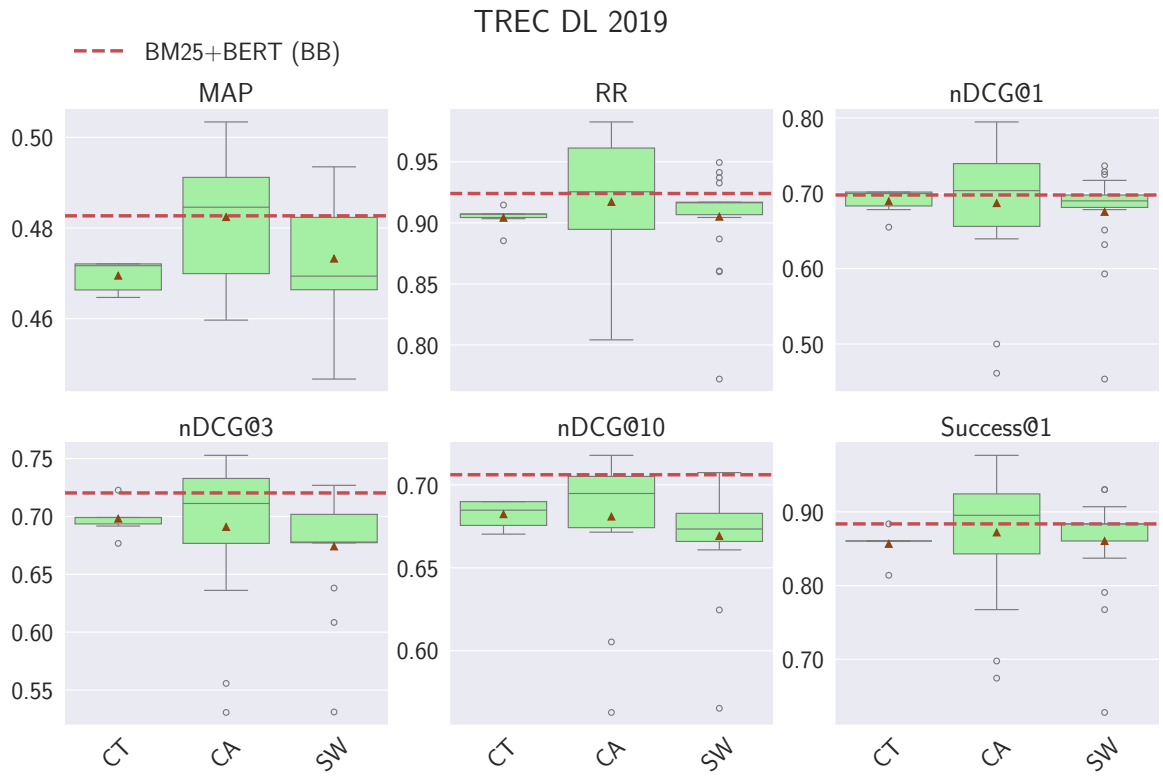


Figure 3.7: Impact of text handling on the effectiveness (y-axis) of PRF approaches for the task of reranking on TREC DL 2019, where CT, CA and SW represent the text handling methods Concatenate and Truncate, Concatenate and Aggregate and Sliding Window, respectively. Baseline BM25+BERT(BB) is marked with a dashed red line.

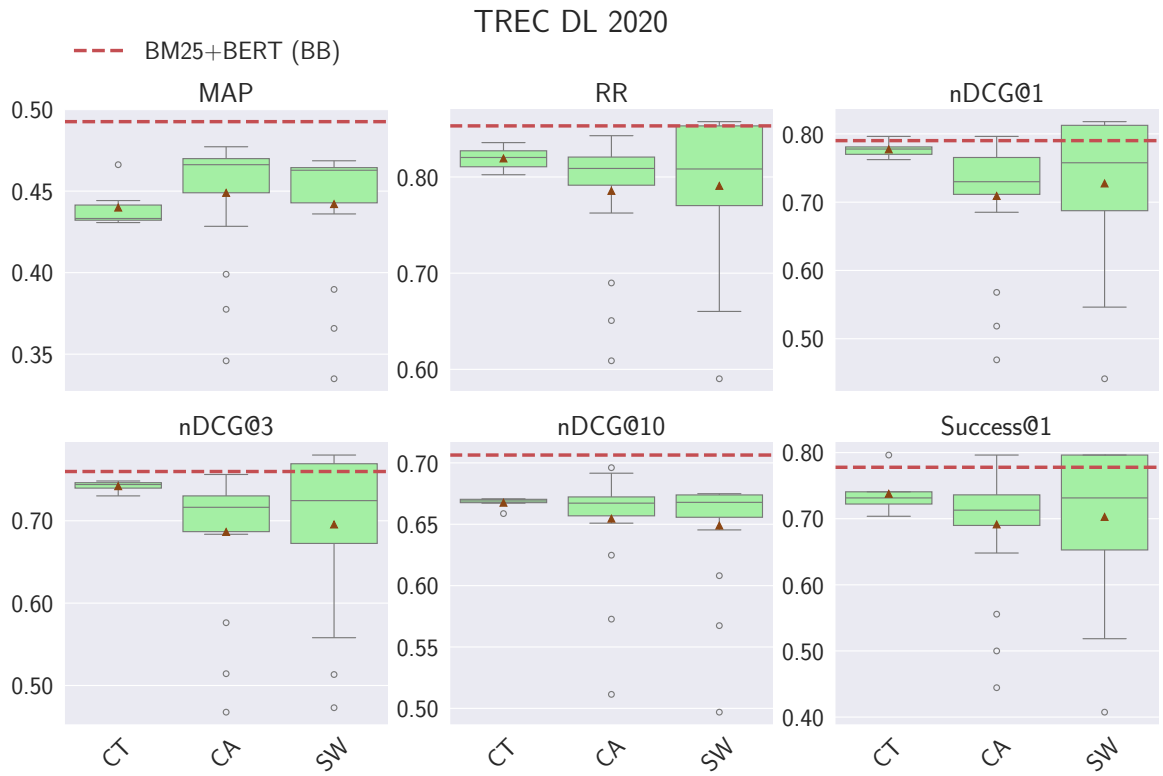


Figure 3.8: Impact of text handling on the effectiveness (y-axis) of PRF approaches for the task of reranking on TREC DL 2020, where CT, CA and SW represent the text handling methods Concatenate and Truncate, Concatenate and Aggregate and Sliding Window, respectively. Baseline BM25+BERT(BB) is marked with a dashed red line.

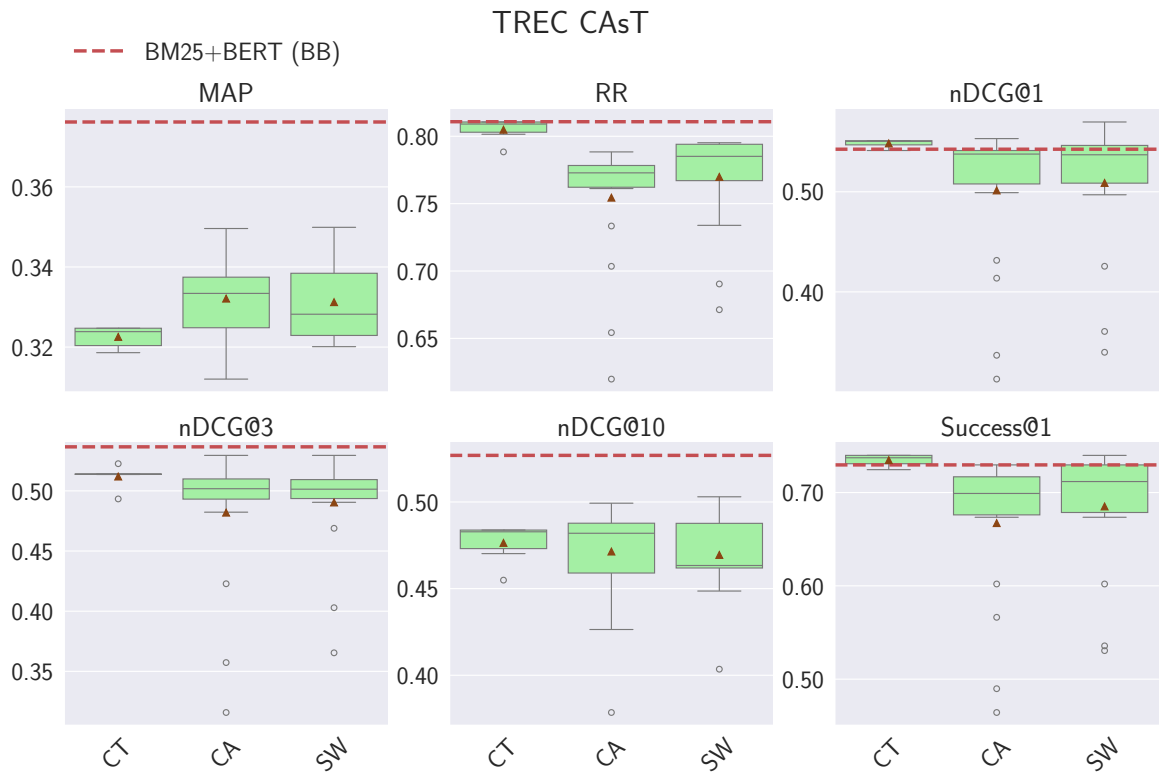


Figure 3.9: Impact of text handling on the effectiveness (y-axis) of PRF approaches for the task of reranking on TREC CAsT, where CT, CA and SW represent the text handling methods Concatenate and Truncate, Concatenate and Aggregate and Sliding Window, respectively. Baseline BM25+BERT(BB) is marked with a dashed red line.

query drift. CT, by truncating the input, restricts the amount of extraneous context added, effectively acting as a noise filter in this case, though it still fails to outperform the baseline significantly.

For WebAP (Figure 3.10) and DL HARD (Figure 3.11), all methods degrade performance, with CA often performing worse than CT. This confirms the noise amplification risk of the CA strategy. In adversarial settings like DL HARD, where top-ranked documents are often misleading, CA maximizes the model's exposure to this misleading information. CT, by truncating the input, coincidentally limits the damage caused by these bad examples.

Overall, the comparison reveals a clear trade-off: Concatenate and Aggregation (CA) is superior when feedback quality is high (e.g., DL 19) because it preserves all signals. However, Concatenate and Truncate (CT) or Sliding Window (SW) can offer more stability in noisy or difficult environments (e.g., DL 20, CAsT) by limiting the model's exposure to irrelevant text. Ultimately, however, none of the text handling strategies can fully overcome the fundamental issue of feedback noise, as evidenced by the lack of consistent gains over the baseline across the board.

3.3.3 Impact of Score Aggregation Methods on the Effectiveness of Text-Based PRF with Neural Rerankers

RQ1.1.3 investigates the impact of score aggregation methods on the effectiveness of Text-based PRF with neural rerankers. To address this question, we vary the score aggregation strategies while

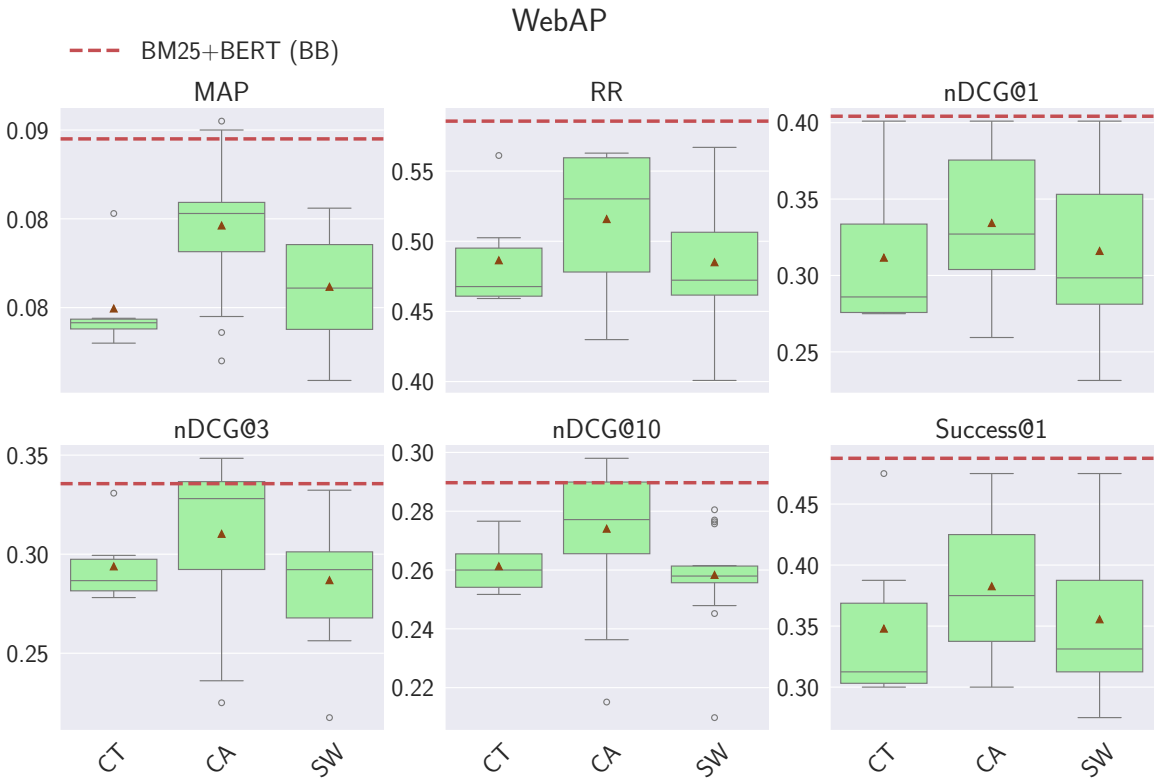


Figure 3.10: Impact of text handling on the effectiveness (y-axis) of PRF approaches for the task of reranking on WebAP, where CT, CA and SW represent the text handling methods Concatenate and Truncate, Concatenate and Aggregate and Sliding Window, respectively. Baseline BM25+BERT(BB) is marked with a dashed red line.

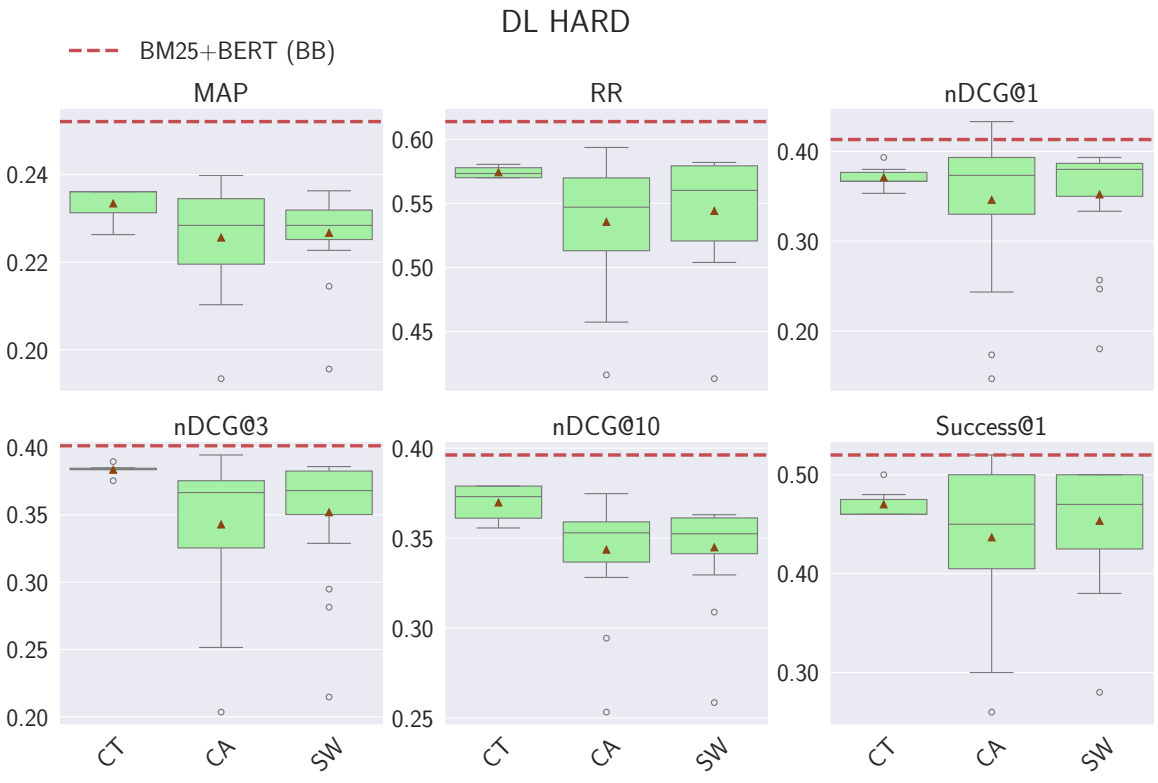


Figure 3.11: Impact of text handling on the effectiveness (y-axis) of PRF approaches for the task of reranking on DL HARD, where CT, CA and SW represent the text handling methods Concatenate and Truncate, Concatenate and Aggregate and Sliding Window, respectively. Baseline BM25+BERT(BB) is marked with a dashed red line.

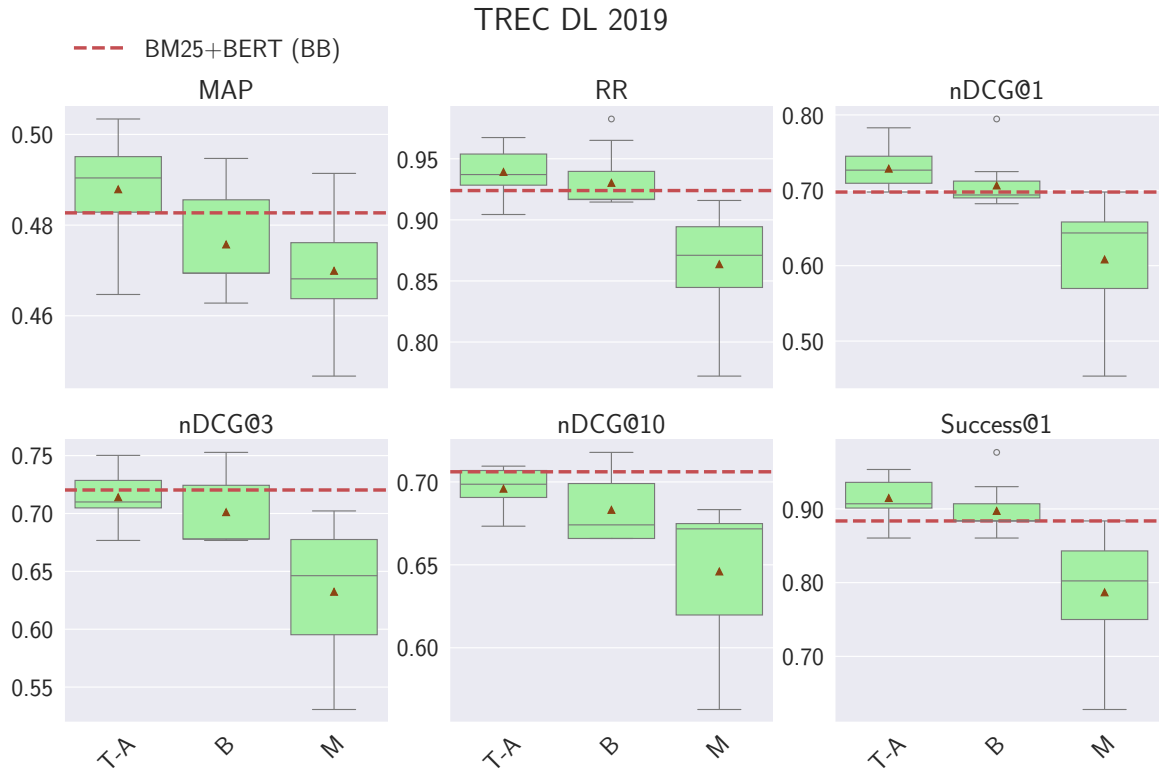


Figure 3.12: Reranking effectiveness (y-axis) by using different score aggregation methods on TREC DL 2019. Where T-A is Text Average, B is Borda, M is Max. Baseline BM25+BERT(BB) is marked with dashed red line.

presenting results across different PRF depths and text handling techniques. Specifically, we evaluate three Text-based score aggregation methods: Average (T-A), Borda (B), and Max (M).

Results are shown in Figure 3.12 – 3.16. For TREC DL 2019 (Figure 3.12), the Average (T-A) aggregation method outperforms the baseline in terms of MAP, RR, and nDCG@1, while Borda (B) performs on par with the baseline for RR. However, the Max (M) method reduces effectiveness across all metrics. This degradation occurs because the Max strategy is highly sensitive to outliers; it propagates the single highest score assigned to a passage across any of the PRF query variations. In PRF, where feedback passages often contain noise, a non-relevant passage might accidentally achieve a high score with one specific feedback passage (a false positive). The Max strategy elevates these false positives to the top of the ranking, displacing truly relevant passages, whereas Average and Borda dampen this noise by requiring consensus across multiple feedback signals.

For TREC DL 2020 (Figure 3.13), the trend worsens. The Max method performs the worst, degrading effectiveness on almost all metrics. Since DL 2020 contains more difficult queries with subtler relevance signals than DL 2019, the feedback set likely contains more "near-miss" or non-relevant passages. The Max strategy amplifies the noise from these passages, leading to significant performance drops.

On TREC CAsT 2019 (Figure 3.14), the Max method again performs poorly. CAsT involves conversational queries where context is crucial. If a feedback passage introduces a topic shift (query drift), the resulting query variation might retrieve passages highly relevant to the drifted topic but

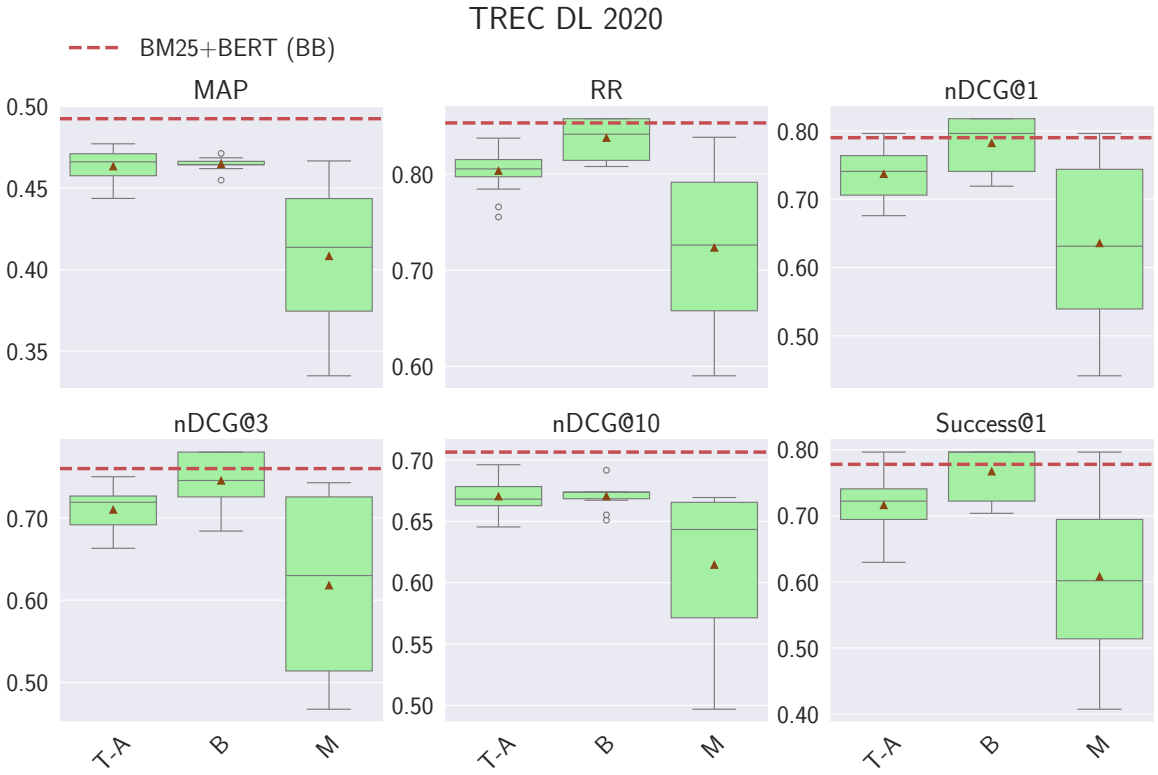


Figure 3.13: Reranking effectiveness (y-axis) by using different score aggregation methods on TREC DL 2020. Where T-A is Text Average, B is Borda, M is Max. Baseline BM25+BERT(BB) is marked with dashed red line.

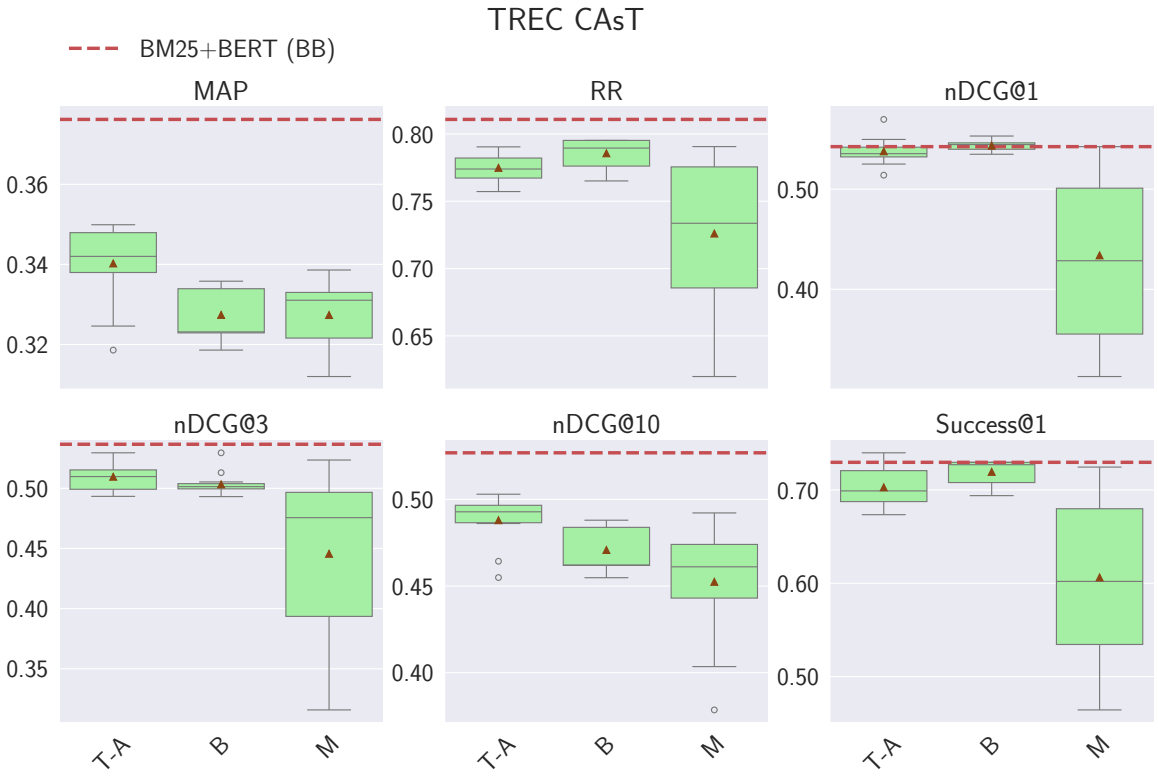


Figure 3.14: Reranking effectiveness (y-axis) by using different score aggregation methods on TREC CAsT. Where T-A is Text Average, B is Borda, M is Max. Baseline BM25+BERT(BB) is marked with dashed red line.

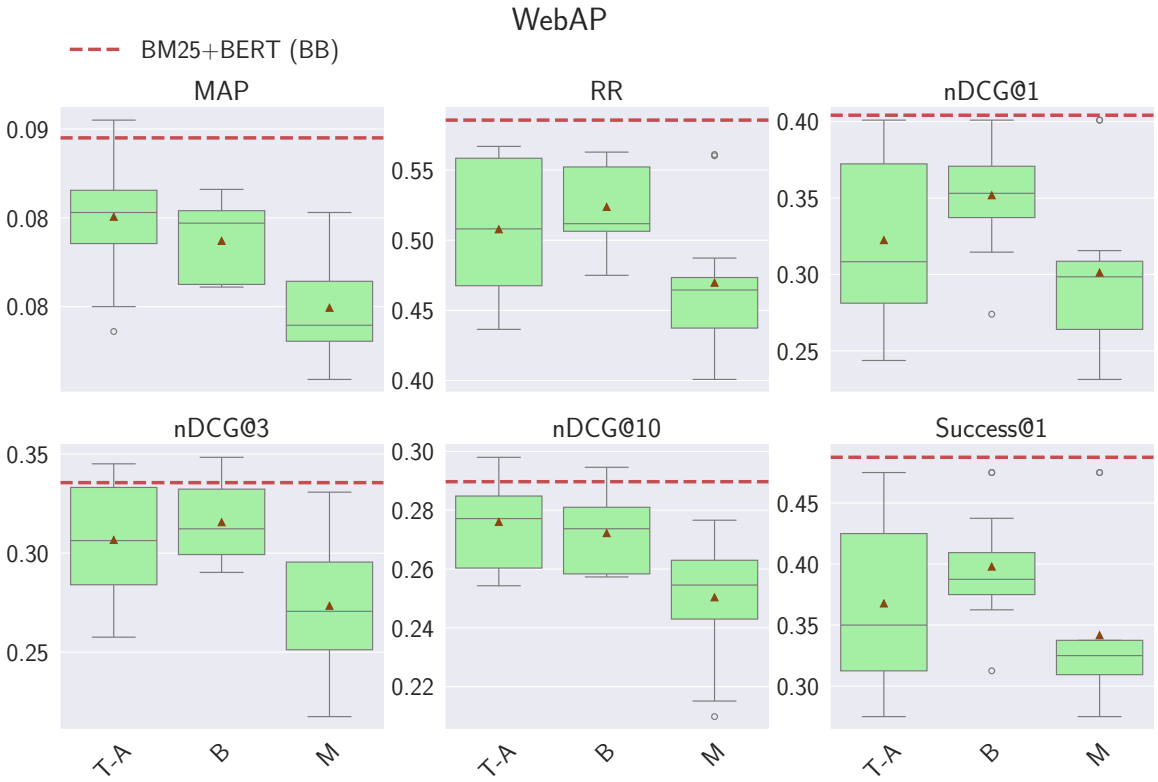


Figure 3.15: Reranking effectiveness (y-axis) by using different score aggregation methods on WebAP. Where T-A is Text Average, B is Borda, M is Max. Baseline BM25+BERT(BB) is marked with dashed red line.

irrelevant to the original user intent. The Max strategy blindly selects these high scores, effectively reinforcing query drift.

For WebAP (Figure 3.15) and DL HARD (Figure 3.16), all score aggregation methods perform worse than the baseline, with Max showing the most severe degradation. DL HARD specifically consists of more challenging queries where the initial top-ranked passages are often non-relevant or misleading. In this adversarial setting, the feedback signal is mostly noise. The Max strategy therefore creates a near "worst-case" scenario by cherry-picking the highest scores generated from this misleading feedback, causing catastrophic failure in ranking effectiveness.

In summary, the Max (M) method consistently underperforms because it lacks a mechanism to filter out noise. Unlike Average (T-A) and Borda (B), which act as smoothing filters by aggregating evidence across multiple feedback signals, Max is vulnerable to individual high-scoring errors. This finding highlights that for Text-based PRF, robust aggregation strategies that prioritize consensus are essential to mitigate the inherent risks of noisy feedback.

3.3.4 Overall Effectiveness of Text-Based PRF Models on the Task of Reranking

RQ1.1.4 investigates the overall effectiveness of Text-based PRF when integrated with neural rerankers. To address this question, we evaluate the best-performing Text-based PRF configurations, using optimal combinations of PRF depth, text handling, and score aggregation parameters identified in previous analyses. Results are presented in Table 3.2. For each dataset, we report the effectiveness of the

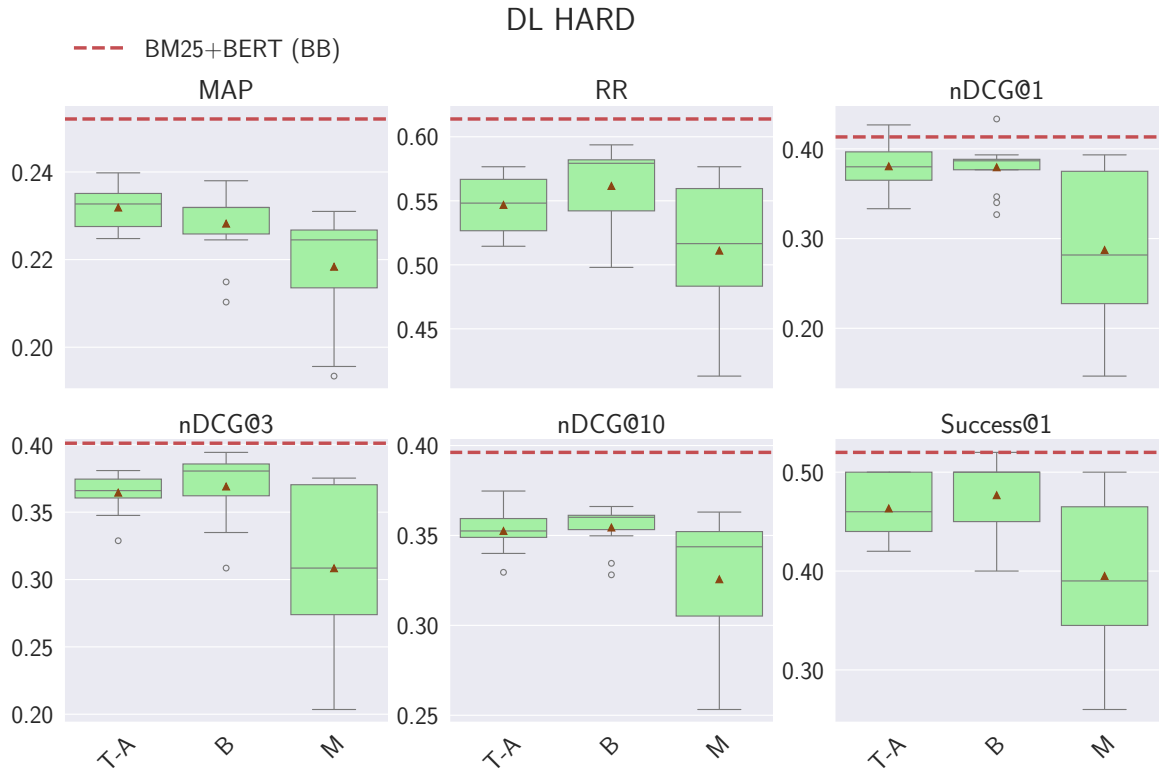


Figure 3.16: Reranking effectiveness (y-axis) by using different score aggregation methods on DL HARD. Where T-A is Text Average, B is Borda, M is Max. Baseline BM25+BERT(BB) is marked with dashed red line.

baseline BERT reranker (BB) alongside the top-performing Text-based PRF variant (BB+PRF). This comparison provides a measure of whether incorporating pseudo-relevance feedback via Text-based methods improves final reranking performance.

For TREC DL 2019, the Text-based PRF model (BB+PRF) improves effectiveness over all evaluation metrics, with statistically significant gains observed primarily in shallow metrics such as RR and nDCG@1 and @3. This indicates that incorporating pseudo-relevant feedback passages can help better rank highly relevant documents at the top positions, particularly when the initial retrieval quality is strong.

For TREC DL 2020, improvements are again observed for shallow metrics, but these gains are not statistically significant. The lack of improvement in deeper metrics and significance may reflect greater noise in the feedback set or increased difficulty in the dataset compared to DL 2019.

For TREC CAsT 2019, BB+PRF does not improve over the BM25+BERT baseline across majority of the metrics, except for a marginal but significant gain in nDCG@1. This limited improvement may be due to the short length of CAsT passages, which aligns well with the training configuration of the BERT reranker. As a result, the model may already be operating near its peak effectiveness without the addition of longer PRF-augmented queries.

On WebAP, improvements are observed for MAP and nDCG@3 and @10, although they are modest. We attribute this to the nature of the dataset: WebAP contains longer passages, which may dilute the effectiveness of a BERT model fine-tuned on shorter text segments. In this case, the added

Table 3.2: Results of Text-based PRF approaches for the task of reranking across different datasets. For each method, the configuration that achieves optimal effectiveness across all metrics is reported. Statistical significance (paired t-test, $p < 0.05$, Bonferroni Correction) is marked as follows: ^{a} indicates significance over BM25, ^{b} over BM25+RM3, and ^{c} over the corresponding baseline without PRF (either better or worse). Best results for each dataset and evaluation metric are shown in **Bold**.

	Model	MAP	RR	nDCG@1	nDCG@3	nDCG@10	R@1000
DL19	BM25	0.3773	0.8245	0.5426	0.5230	0.5058	0.7389
	BM25+RM3	0.4270	0.8167	0.5465	0.5195	0.5180	**0.7882**
	BM25+BERT (BB)	0.4827	0.9240	0.6977	0.7203	0.7061	0.7389
	BB+PRF($k = 10$,CA,BORDA)	**0.4947**^{ab}	**0.9826**^{abc}	**0.7946**^{abc}	**0.7528**^{abc}	**0.7178**^{ab}	0.7389
DL20	BM25	0.2856	0.6585	0.5772	0.5021	0.4796	0.7863
	BM25+RM3	0.3019	0.6360	0.5648	0.4740	0.4821	**0.8217**
	BM25+BERT (BB)	**0.4926**	0.8531	0.7901	0.7598	**0.7064**	0.7863
	BB+PRF($k = 3$,SW,BORDA)	0.4644^{ab}	**0.8575**^{ab}	**0.8179**^{ab}	**0.7798**^{ab}	0.6739^{ab}	0.7863
CAST	BM25	0.2936	0.6502	0.3631	0.3542	0.3526	**0.8326**
	BM25+RM3	0.3132	0.6556	0.3971	0.3829	0.3817	0.8246
	BM25+BERT (BB)	**0.3762**	**0.8108**	0.5425	**0.5366**	**0.5269**	**0.8326**
	BB+PRF($k = 10$,CC)	0.3247^{ac}	0.8106^{ab}	**0.5510**^{abc}	0.5140^{ab}	0.4838^{abc}	0.8326
WEBAP	BM25	0.0436	0.3099	0.1667	0.1604	0.1404	0.2944
	BM25+RM3	0.0536	0.2767	0.1344	0.1316	0.1376	**0.3472**
	BM25+BERT (BB)	0.0845	**0.5856**	**0.4042**	0.3356	0.2897	0.2944
	BB+PRF($k = 15$,CA,AVG)	**0.0855**^{ab}	0.5459^{ab}	0.3271^{ab}	**0.3444**^{ab}	**0.2980**^{ab}	0.2944
DL-HARD	BM25	0.1845	0.5422	0.3533	0.3137	0.2850	0.6288
	BM25+RM3	0.1925	0.4381	0.2467	0.2508	0.2555	**0.6522**
	BM25+BERT (BB)	**0.2521**	**0.6139**	0.4133	**0.4012**	**0.3962**	0.6288
	BB+PRF($k = 3$,CA,BORDA)	0.2380^{a}	0.5937^{b}	**0.4333**^{b}	0.3944^{b}	0.3550^{abc}	0.6288

context from PRF could partially compensate for the mismatch in passage length, improving ranking performance.

For DL HARD, BB+PRF shows a minor improvement in nDCG@1, but it is not statistically significant to the BB baseline. Notably, performance on nDCG@10 is significantly worse than the baseline. This outcome suggests that the PRF mechanism may be incorporating misleading or non-relevant signals, leading to query drift. This is supported by the very low values of shallow metrics on DL HARD compared to TREC DL 2019 and 2020 (which share the same document collection), indicating that the top-ranked feedback passages are less likely to be relevant.

To summarize, the BB+PRF model demonstrates its potential to improve reranking effectiveness, particularly for shallow metrics and on datasets with high-quality initial retrieval results. While it consistently outperforms classical baselines like BM25 and BM25+RM3, its improvements over BM25+BERT are more limited and inconsistent. These mixed results can be largely attributed to the increased input length introduced by PRF, which may misalign with the pretraining and fine-tuning characteristics of the BERT reranker, especially on datasets with shorter or noisier passages.

Table 3.3: Query latency of the investigated methods on TREC DL 2019: the lower latency, the better (faster).

	Models	Latency (ms/q)
Baselines	BM25 (Anserini)	81
	BM25 + RM3 (Anserini)	140
BERT Reranker	BM25 + BERT(BB)	3,246
	BM25 + BERT Large	9,209
Text-based PRF Reranker	BB+PRF($k = 5$)-CT	6,889
	BB+PRF($k = 5$)-CA	17,266
	BB+PRF($k = 5$)-SW	22,314

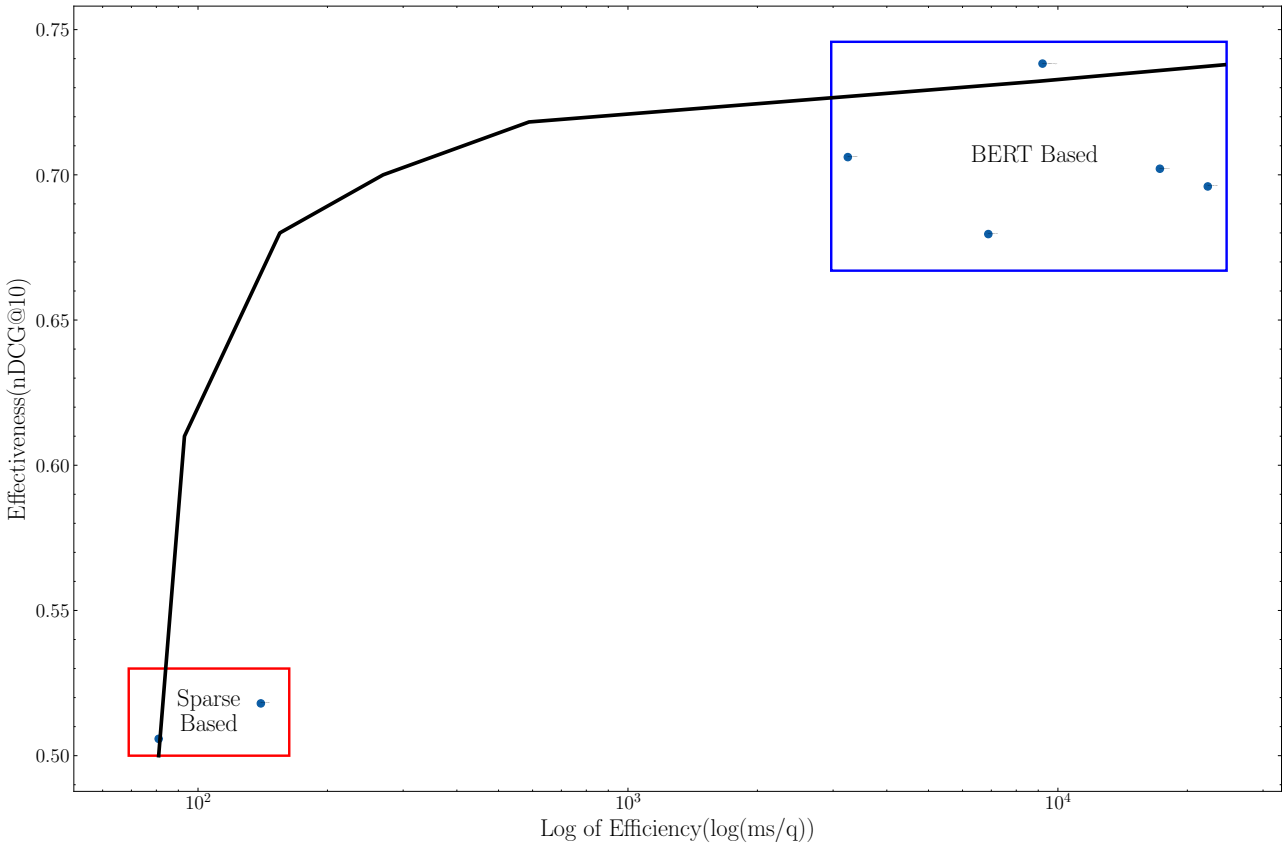


Figure 3.17: Trade-off between effectiveness and efficiency for all methods in our experiments. Effectiveness is measured using nDCG@10, and efficiency is measured using $\log(ms/q)$. The sparse baselines (BM25 and BM25+RM3) cluster on the left bottom corner (red box). All rerankers, i.e., BM25+BERT(base/large), Text-based PRF, cluster on the top right side (blue box); these methods present the worst efficiency compared to others. The black line shows the trade-off trend between effectiveness and efficiency.

3.3.5 Impact of Text-Based PRF Models on the Efficiency of Reranking with Neural Rerankers

To answer **RQ1.1.5**, we examine the impact of Text-based PRF on the efficiency of reranking with neural rerankers. In this context, efficiency is measured in terms of query latency, a critical factor for retrieval systems. Our goal is to quantify the computational overhead introduced by Text-based

PRF methods and compare it against the baseline BERT-based reranker and traditional sparse retrieval methods.

Efficiency results are reported in Table 3.3. Compared to the BM25+BERT baseline (BB), Text-based PRF methods (BB+PRF) exhibit substantially higher query latency. This increase is primarily due to approaches such as Concatenate and Aggregation (CA) and Sliding Window (SW), which generate multiple reformulated queries based on top-ranked feedback passages. For instance, with a PRF depth of 5, CA produces five additional query variants, each of which requires a separate BERT inference alongside the original input. This results in up to six BERT inferences per query, leading to a near-linear growth in latency as PRF depth increases.

To ensure inference consistency and maximize GPU throughput, all query-passage pairs are padded or truncated to a fixed input length of 512 tokens (typically 256 tokens for the query and 256 for the passage). This standardization supports efficient batching and ensures stable per-inference latency, regardless of the original or expanded query length. As a result, input length does not directly influence latency within each BERT pass. Instead, the main bottleneck arises from the number of inference passes required.

The impact is particularly evident for CA and SW methods, which treat each feedback passage independently rather than fusing all PRF content into a single input. This design leads to a linear relationship between PRF depth and the number of BERT inferences, making the method increasingly expensive as more feedback passages are incorporated.

In summary, while Text-based PRF preserves fixed per-inference latency due to standardized input formatting, total query latency grows linearly with PRF depth. This scalability limitation makes high-depth PRF configurations impractical for latency-sensitive applications. These findings highlight the importance of balancing effectiveness gains with computational cost and point to the need for more efficient strategies for integrating PRF into neural reranking pipelines.

3.3.6 Challenges and Limitations When Integrating Text-Based PRF into Neural Rerankers

From our experiments, integrating Text-based PRF with neural rerankers introduces two main challenges: high computational cost and strict input length limits, which map to efficiency and effectiveness concerns, respectively. Prior work has mainly emphasized effectiveness, often applying PRF in a second-stage reranking pipeline [94, 206, 233, 244] to manage computational load, but these rely on weak first-stage retrieval and inherit their limitations.

We evaluated the applicability, effectiveness, and efficiency of Text-based PRF specifically for reranking, and found that while it mitigates input length issues, the resulting effectiveness gains are typically marginal. These gains come at the expense of significant computational cost, making Text-based PRF unsuitable for low-latency search applications, as shown in Figure 3.17.

The BERT reranker used in our experiments is fine-tuned on MS MARCO [155], where queries are shorter than the expanded queries produced by PRF. This mismatch between training and testing

distributions likely impacts performance, limiting the generalizability of the model to PRF-augmented inputs. Despite this mismatch, Text-based PRF sometimes yields marginal improvements in individual metrics across certain datasets. However, in most cases it performs worse than the BM25+BERT baseline, suggesting limited practical benefit under the current setup. It remains an open question whether fine-tuning the BERT model on long PRF-augmented queries would reduce this mismatch and improve effectiveness. Although not investigated in this thesis, this direction still presents an interesting opportunity to better align training and inference conditions for Text-based PRF.

We also acknowledge that pseudo-relevance feedback may introduce query drift, which is a well-known risk in PRF. However, due to differences in inputs and experimental conditions, we could not fairly isolate the effect of query drift in this study.

3.4 Summary

In this Chapter, we conducted a comprehensive investigation into the integration of Text-based PRF with neural rerankers. The proposed approach, designed specifically for the reranking task, leverages relevance signals from feedback passages within a BERT-based reranking framework. We examined the impact of PRF depth, text handling strategies, and score aggregation methods, and evaluated the overall effectiveness and efficiency of the Text-based PRF methods. Our findings reveal several challenges and limitations. These insights also highlight the need for future research into more effective and efficient methods for integrating PRF with neural models.

The Text-based PRF approaches were applicable exclusively to the reranking task. Regarding the depth of PRF, we observed that using 3, 5, or 10 feedback passages can led to marginal improvements in effectiveness, while deeper configurations (15 or 20 passages) often degraded performance, this is likely due to increased query drift introduced by excessive feedback. For text handling methods, the Concatenate and Aggregation (CA) approach performed best in most cases, although in terms of overall effectiveness, these methods provided only marginal and inconsistent improvements over the baseline across datasets and metrics.

However, the most notable limitation lies in their efficiency, particularly in terms of query latency. Text handling methods such as Concatenate and Aggregation (CA) introduce substantial computational overhead, as they require multiple BERT inferences per query—one for each feedback passage. This significantly increases total query latency, making Text-based PRF impractical for deployment in real-time search environments. The experiments conducted in this chapter highlight the trade-off between modest effectiveness gains and the prohibitive cost of latency, offering insights for the design of neural reranking systems in latency-sensitive applications.

Building on the limitations identified in this chapter, the next chapter shifts focus from integrating PRF into computationally expensive rerankers to exploring more efficient dense retrievers. Specifically, it investigates the reproducibility and performance characteristics of ANCE-PRF [234], a previously proposed method that integrates PRF signals directly into the query encoder of dense retrieval models to achieve a better balance between effectiveness and efficiency.

The implementations of our Text-based PRF methods are publicly available at `https://github.com/ielab/Neural-Relevance-Feedback-Public/tree/master/Text_Based_PRF`, along with the complete set of empirical results.

Chapter 4

ANCE-PRF Reproducibility and Analysis

In the previous chapter, we examined Text-based PRF approaches applied to neural reranking tasks. These methods incorporated various text handling and score aggregation strategies on top of BERT-based rerankers. However, due to the significant computational overhead of the underlying reranker, the marginal effectiveness gains from PRF integration render such approaches impractical for real-world deployment. With the advent of dense retrievers, which employ transformer-based encoders to encode queries and passages into vector representations, relevance estimation is performed through similarity computations (e.g., dot product, cosine similarity, or Euclidean distance) between these embeddings to generate a ranked list of passages. These types of dense retrieval methods offer vastly improved efficiency and competitive effectiveness compared to traditional two-stage pipelines (e.g., BM25 reranked by BERT) [60, 72, 73, 75, 83, 85, 173, 223, 238]; there has been growing interest in integrating PRF directly within dense retrievers.

In this chapter, we focus on a representative and state-of-the-art approach to integrate PRF into the dense retrievers: the ANCE-PRF method proposed by Yu et al. [234]. This method builds on top of the ANCE dense retriever [223], which is used to retrieve the top-k passages for a given query. The original query is then concatenated with the text of these top-ranked passages to form an expanded query. This expanded query is passed to a dedicated ANCE-PRF encoder, which is trained to incorporate feedback signals into the query representation, to produce a new dense query embedding (refer to Figure 4.1). This embedding is then used to compute similarity with pre-encoded passage representations for final ranking. Unlike the original ANCE model, which encodes only the queries, the ANCE-PRF encoder is trained to include PRF passages with the original queries, thereby enabling the model to learn to contextualize queries based on retrieved evidence.

Our objective is to reproduce the ANCE-PRF method and evaluate its training and retrieval effectiveness. Beyond reproduction, we analyze the sensitivity of the method to hyperparameter configurations and assess its robustness under varied training settings. We also investigate the extent to which the underlying ANCE-PRF mechanism generalizes to other dense retrievers. To support future research, we release an open-source implementation of the ANCE-PRF pipeline, including trained checkpoints for ANCE and selected alternative dense retrieval backbones.

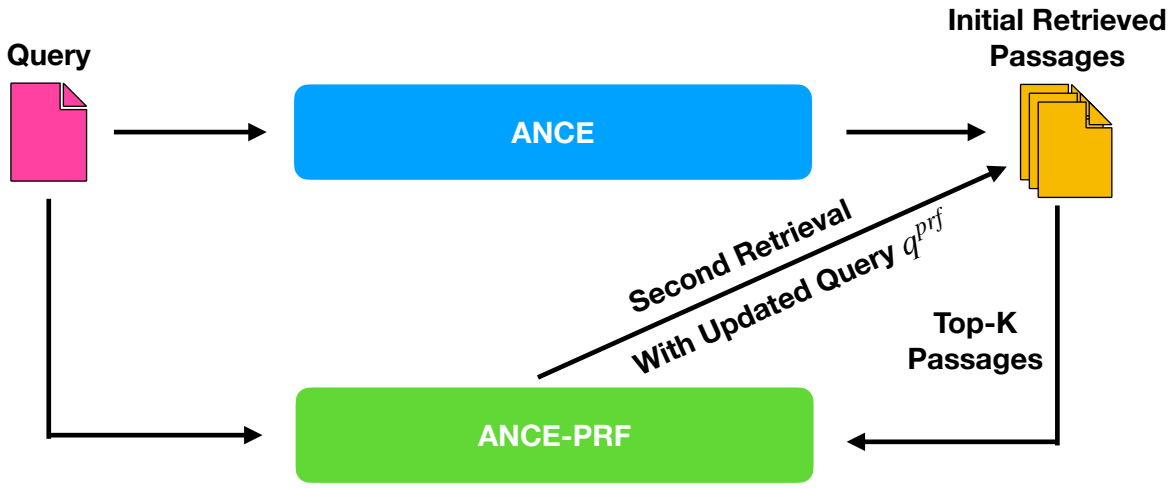


Figure 4.1: Pipeline of the state-of-the-art ANCE-PRF model.

By identifying the practical challenges and modeling limitations of ANCE-PRF, this chapter again addresses our high level research question from Chapter 3.

RQ1.2: What are the challenges and limitations of integrating Pseudo-Relevance Feedback into neural models?

These findings further motivate the exploration of lightweight and training-free alternatives to overcome the drawbacks identified in both text-based and trained feedback-integrated dense retrieval methods, which we will pursue in the later chapters.

4.1 ANCE-PRF: Improving Query Representations for Dense Retrievers with Pseudo Relevance Feedback

Next we briefly describe the ANCE-PRF method [234], which extends ANCE [223] to accept PRF signals from the top-k passages to be encoded in combination with the query to form a new query representation.

In ANCE, the score of a passage p for a query q is computed by separately encoding q and p using the RoBERTa [119] pre-trained deep language model, and then calculating the inner product between the resulting dense representations:

$$s_{\text{ANCE}}(q,p) = \text{ANCE}^{\text{q}}(\langle s\rangle q\langle /s\rangle) \cdot \text{ANCE}^{\text{p}}(\langle s\rangle p\langle /s\rangle) \tag{4.1}$$

where ANCE^{q} and ANCE^{p} represent the query and the passage encoders, respectively, and $\langle s\rangle$ and $\langle /s\rangle$ represent the `[CLS]` and `[SEP]` tokens in ANCE. Both encoders use the final layer of the $\langle s\rangle$ token embedding as the query and passage dense representations. In ANCE, the passage embeddings are pre-computed offline and stored in an index, while the query embeddings are encoded at inference (query) time [223]. For fine-tuning Equation 4.1, ANCE adopts noisy contrastive estimation loss and employs a negative sampling strategy where negative samples are dynamically retrieved from an asynchronously updated ANCE passage index [223].

ANCE-PRF uses a similar schema to score passages for retrieval:

$$s_{\text{ANCE-PRF}}(q,p) = \text{ANCE}^{\text{prf}}(\langle s\rangle q\langle /s\rangle p_1\langle /s\rangle ... p_k\langle /s\rangle) \cdot \text{ANCE}^{\text{p}}(\langle s\rangle p\langle /s\rangle) \tag{4.2}$$

where ANCE^{prf} is the newly trained PRF query encoder and $\langle s\rangle q\langle /s\rangle p_1\langle /s\rangle ...\ p_k\langle /s\rangle$ is the text concatenation of the original query text q with the feedback passage texts $p_1, p_2, ..., p_k$ (in addition to CLS and separator tokens). We denote $\mathbf{q}^{\text{prf}}$ as the query embedding generated through PRF by ANCE^{prf}.

For the training of the PRF query encoder $\text{ANCE}^{\mathbf{prf}}$, ANCE-PRF uses the standard noisy contrastive estimation loss:

$$\mathcal{L} = -log\frac{exp(\mathbf{q}^{\text{prf}} \cdot \mathbf{p}^{+})}{exp(\mathbf{q}^{\text{prf}} \cdot \mathbf{p}^{+}) + \Sigma_{\mathbf{p}^{-}\in\mathbf{P}^{-}} exp(\mathbf{q}^{\text{prf}} \cdot \mathbf{p}^{-})} \tag{4.3}$$

where $\mathbf{p}^{+}$ represents a relevant passage embedding for the query, and $\mathbf{p}^{-}$ represents an irrelevant passage embedding (obtained from the negative sampling technique). During the training process, the ANCE-PRF model uses the passage embeddings from the original ANCE model. Therefore, the passage embeddings remain unchanged in the ANCE-PRF model: it is only the query embedding that changes into the PRF query embedding $\mathbf{q}^{\text{prf}}$.

With the PRF query embeddings $\mathbf{q}^{\text{prf}}$ from the trained ANCE-PRF query encoder $\text{RoBERTa}^{\text{prf}}$, the ANCE-PRF model performs another retrieval with the original ANCE index:

$$s^{\text{prf}}(q,p) = \mathbf{q}^{\text{prf}} \cdot \text{ANCE}^{\text{p}}(< s > p < /s >) \tag{4.4}$$

Where ANCE^{p} represents the original ANCE passage encoder and the passage embeddings are actually pre-computed and stored offline, so during retrieval time, only the query needs to be encoded.

Intuitively, ANCE-PRF should provide increases in search effectiveness because the newly trained ANCE-PRF query encoder learns to extract relevant information for the query from the PRF passages using the Transformer attention mechanism [205]. After training, the ANCE-PRF query encoder would then pay more attention to the relevant tokens in the PRF passages, while ignoring the non-relevant tokens from this signal. Although Yu et al. [234] do not report the query latency of ANCE-PRF, this should be approximately twice that of the original ANCE model.

4.2 Empirical Evaluation

4.2.1 Datasets

The datasets used in the original work of Yu et al. [234] are TERC DL 2019 [27], TREC DL 2020 [26], DL Hard [137], and MS MARCO Passage Ranking V1 [151]. These datasets are based on the same corpus provided by MS MARCO Passage Ranking V1, which has $\sim$ 8.8M passages in total. Note that for TREC DL 2019/2020, each query has multiple judgements on a relevance scale from 0 to 3, while MS MARCO Passage Ranking V1 only has an average of 1 judgement per query with binary relevance, either 0 or 1.

The original ANCE-PRF paper used the training split from MS MARCO Passage Ranking V1 for training ANCE-PRF, which includes $\sim$ 530K queries. The trained models were evaluated on TREC DL 2019 (43 judged queries), DL 2020 (54 judged queries), DL HARD, and MS MARCO Passage Ranking V1 Dev set (6,980 queries). For direct comparison with the ANCE-PRF model, we follow the same process except for evaluation on TREC DL HARD (the results on this dataset for other dense retrievers considered in this chapter are not publicly available). Statistical significance is assessed using a two-tailed paired t-test with Bonferroni correction.

4.2.2 Generalization to Other Dense Models

The original work by Yu et al. [234] only considered ANCE [223] as the initial dense retriever. To validate generalisability on other dense retrievers, we consider two retrievers that achieve higher effectiveness than ANCE, TCT ColBERT V2 HN+ [114] and DistilBERT KD TABS [73]. TCT ColBERT V2 HN+ uses a BERT-style encoder to encode queries and passages, while DistilBERT KD TABS uses a DistilBERT style encoder. These two dense retrievers are different from ANCE with respect to the training process, and the inference is slightly different from each other. We refer the reader to the original papers for further details. The output of these three dense retrievers are all embedding vectors that represent either the query or the passage based on the input. The index for all three models are pre-computed and stored offline.

TCT ColBERT V2 HN+

$$\mathbf{q}_{\text{TCT}}^{\text{prf}} = \text{TCT}^{\text{prf}}([CLS]\,[Q]\,q[SEP]p_1[SEP]...p_k[MASK]*512) \tag{4.5}$$

The scoring function for TCT ColBERT V2 HN+ is:

$$s_{\text{TCT-PRF}}(q,p) = \mathbf{q}_{\text{TCT}}^{\text{prf}} \cdot \text{TCT}^{\text{p}}([CLS]\,[D]\,p) \tag{4.6}$$

where TCT^{prf} represents the new PRF query encoder based on TCT ColBERT V2 HN+ query encoder. The input needs a `[CLS]` token as well as a `[Q]` in text as prepend to the actual query text, then the PRF passage texts are separated by `[SEP]` token, then a pad `[MASK]` token fills in the gap if the input is smaller than the max input size of the model, which is 512 for BERT-based models [37].

For retrieval, as in Equation 4.6, TCT^{p} represents the passage encoder, and the input passage text is prepended with `[CLS]` token and `[D]` in text.

DistilBERT KD TABS

For DistilBERT KD TABS, Equation 4.5 becomes:

$$\mathbf{q}_{\text{DBERT}}^{\text{prf}} = \text{DBERT}^{\text{prf}}([CLS]q[SEP]p_1[SEP]...p_k[SEP]) \tag{4.7}$$

And the scoring function for DistilBERT KD TABS is:

$$s_{\text{DBERT-PRF}}(q,p) = \mathbf{q}^{\text{prf}}_{\text{DBERT}} \cdot \text{DBERT}^{\text{p}}([CLS]p[SEP]) \tag{4.8}$$

Similar to TCT ColBERT V2 HN+, except the input is a standard BERT input with `[CLS]` token as prepend and `[SEP]` token as separators to separate the PRF passages for both PRF query encoding and retrieval.

4.2.3 Inferencing and Training of ANCE-PRF

During the reproduction of ANCE-PRF, we encountered several critical issues affecting both inference and training. In this section, we systematically detail these challenges and describe the methods we derived to address them.

Challenges in Reproducing Inference Procedures

To reproduce the ANCE-PRF results, we relied on the only publicly available resource from the original authors: a model checkpoint trained with PRF depth $k = 3$. As no official inference code was released, we implemented the ANCE-PRF inference pipeline using the open-source information retrieval toolkit Pyserini[1] [112], which natively supports the underlying ANCE [223] dense retriever. Our implementation extends this framework by introducing a second-stage retrieval step that uses the ANCE-PRF query encoder. Specifically, the first retrieval pass employs the standard ANCE query encoder, while the second pass re-encodes the query, which is now concatenated with feedback passages, using the trained ANCE-PRF query encoder. Importantly, both retrieval stages share the same pre-encoded passage index, ensuring that any performance differences are attributable solely to the query representation.

Challenges in Reproducing Training Procedures

Apart from the inference, the authors did not released the training code as well. To reproduce the ANCE-PRF training process, we utilise the open source dense retriever training toolkit Tevatron[2]. According to the original work from Yu et al. [234], all hyperparameters used in ANCE-PRF training are the same as ANCE training, and the ANCE-PRF query encoder is initialized from ANCE First Passage model[3] [223]. Although some of the parameters are still not clear (e.g. learning rate, optimizer), we tried our best to replicate the same model as ANCE-PRF with $k = 3$ by adjusting different training settings.

We also experimented with the other two dense retrievers, TCT ColBERT V2 HN+ [114] and DistilBERT KD TASB [73], with the same training process to investigate the generalisability of the ANCE-PRF model. Therefore, we adopted the same hyperparameters from these two models and trained with the same settings as ANCE-PRF.

[1] `https://github.com/castorini/pyserini`
[2] `https://github.com/texttron/tevatron`
[3] `https://github.com/microsoft/ANCE`

All models in our experiments were trained on two Tesla V100 SMX2 32GB GPUs. In the original work, the ANCE-PRF model is trained with per device batch size 4 and gradient accumulation step 8 for 450K steps, which is equivalent to per device batch size 32 for ∼56K steps; therefore, in our training, we use 10 epochs which is roughly ∼80K steps.

4.2.4 Evaluation Metrics

The official evaluation metric for MS MARCO Passage Ranking V1 dataset is MRR@10 [151], for TREC DL 2019 and 2020 are nDCG@10 and Recall@1000 [26, 27]. For Recall@1000 on TREC DL 2019 and 2020, the judgements are binarized at relevance point 2 according to the official guideline. Besides the official evaluation metrics, the authors in the original work [234] also use HOLE@10 as an additional evaluation metric to measure the unjudged fraction of top 10 retrieved passages [223], in the reflection of the coverage of the pooled labels on these dense retrieval systems. However, in our experiments, we kept the official evaluation metrics only, for the sake of comparison with other models and baselines. Statistical significance between model results was tested using two tails paired t-test.

4.2.5 Research Questions

In this chapter, we aim to address the following sub research questions under the main research question:

RQ1.2: What are the challenges and limitations of integrating PRF into neural models?
Along with the reproduction of the original method from Yu et al. [234]:

RQ1.2.1: Is it possible to replicate the inference results of ANCE-PRF given only a checkpoint of the trained model provided by the original authors?

RQ1.2.2: The training process is governed by a number of hyperparameters and choices, including learning rate, optimizer, and negative sampling technique settings. Given the insufficient details in the original study, it is reasonable to expect that researchers attempting to reproduce the ANCE-PRF method may set these parameters to values different from those in the original study. We are then interested to study:

What is the impact of ANCE-PRF training hyperparameters on the effectiveness of the method, and in particular if this is robust to different hyperparameter settings?

RQ1.2.3: The PRF strategy underlying ANCE-PRF can be adapted to other dense retrievers as observed by Yu et al. [234], but was not empirically validated. The original ANCE-PRF model was only trained with ANCE [223] as the initial dense retriever. We are then interested to investigate:

Do the improvements observed for ANCE-PRF generalise to other dense retrievers, such as the two more effective models, TCT ColBERT V2 HP+ [114] and DistilBERT KD TASB [73]?

Table 4.1: The reproduced results with the author provided $k = 3$ checkpoint for ANCE-PRF (ANCE-PRF 3) after inference. **Original** represents results with the PRF query contains both uppercase and lowercase letters. **Lowercase** indicates the results with PRF query converted to lowercase when tokenize. **ANCE-PRF 3** shows results in original paper. Best results are marked with **Bold**. Statistically significant improvements over ANCE baseline are marked with †.

Datasets	MS MARCO			TREC DL 2019		TREC DL 2020	
	MRR@10	nDCG@10	R@1000	nDCG@10	R@1000	nDCG@10	R@1000
ANCE [223]	0.3300	0.3880	0.9590	0.6480	0.7550	0.6460	0.7760
ANCE-PRF 3 [234]	**0.3440**†	0.4010†	0.9590	**0.6810**	0.7910†	**0.6950**†	**0.8150**
Original	0.3420	0.3990	**0.9600**	0.6780	**0.7920**†	0.6740	0.7940
Lowercase	**0.3440**†	**0.4020**†	**0.9600**	**0.6810**	0.7910†	**0.6950**†	**0.8150**

4.3 Results and Analysis

4.3.1 RQ1.2.1: Reproduce ANCE-PRF Inference

Our reproduction results of ANCE-PRF [234] is shown in Table 4.1. During the replication process, we found that the ANCE-PRF model is sensitive to the uppercase or lowercase of the letters. For the original queries used in all three datasets in this experiment, no uppercase letters existed, therefore this detail is omitted from the original paper. But from our replication experiments, uppercase letters exists in the collection corpus, and the token ids and their associated tokens embeddings are different with different case of the same word. Therefore, for PRF queries, after the concatenate the PRF passages with the original query text, the new PRF queries contain uppercase letters and lead to different tokens after tokenization, and resulting in different performance compares to what is reported in the original paper. On the other hand, if we set the tokenizer to do lowercase at inference time, then we can get the same results as the original paper. Hence, we successfully reproduced the ANCE-PRF model for inferencing by using the checkpoint provided by the authors.

To answer **RQ1.2.1**, we confirmed that it is possible to reproduce the same results with the model checkpoint; however, one key detail that was missing in the paper is the lowercase process to the PRF query. We make our ANCE-PRF inference implementation publicly available in Pyserini toolkit [4] so that others can easily reproduce the same results in the original paper with the author provided model checkpoint.

4.3.2 RQ1.2.2: Reproduce ANCE-PRF Training

Apart from the inferencing, which is fairly easy by using the checkpoint provided by the original authors, we would like to see if we can reproduce the model by following the training settings provided in the paper. However, some details were missing and we had to consult with the original authors to identify their exact settings. After clarifying the training parameters, we used the same setting to train

[4] `https://github.com/castorini/pyserini/blob/master/docs/experiments-ance-prf.md`

Table 4.2: The reproduced results with the trained ANCE-PRF 3 checkpoint after inference. **ANCE-PRF 3** shows the results from the original paper. **ANCE** represents the results from the original ANCE model. **Reproduced (AdamW)** is the results from our reproduced ANCE-PRF model with AdamW optimizer, **Reproduced (LAMB)** is the results from our reproduced ANCE-PRF model with LAMB optimizer. Best results are marked with **Bold**. Statistically significant improvements over ANCE baseline are marked with †.

Datasets	MS MARCO			TREC DL 2019		TREC DL 2020	
	MRR@10	nDCG@10	R@1000	nDCG@10	R@1000	nDCG@10	R@1000
ANCE [223]	0.3300	0.3880	0.9590	0.6480	0.7550	0.6460	0.7760
ANCE-PRF 3 [234]	0.3440†	0.4010†	0.9590	**0.6810**	0.7910†	0.6950†	**0.8150**
Reproduced (AdamW)	0.3390	0.3930	0.9590	0.6670	0.7750	0.6760	0.7970
Reproduced (LAMB)	**0.3470**†	**0.4050**†	**0.9630**	0.6720	**0.7940**†	**0.7010**†	0.8140

our own ANCE-PRF model, the results are shown in Table 4.2. To obtain the ANCE results, we used the checkpoints provided by the original authors Xiong et al. [223].

We obtained results that are close to those reported and with similar trends. The minor differences between the two results can be potentially explained by random neuron drop out during training and the random seed while sampling the hard negatives from the initially retrieved ANCE results.

In the original study, the authors reported that they were using all hyper-parameters from ANCE [223] training, and all models were trained on two RTX 2080 Ti GPUs with per-GPU batch size 4 and gradient accumulation step 8 for 450K steps. However, some parameters are still unclear in ANCE training. We trained the ANCE-PRF model with two Tesla-V100 SMX2 32GB GPUs with per-GPU batch size 32, learning rate $1e-5$, no in batch negatives or cross batch negatives, and no gradient accumulation steps for 10 epoches. The reason why we choose to remove the gradient accumulation step setting is because we are using GPUs with larger memory. In the original settings, 450K steps with gradient accumulation step 8 and per-GPU batch size 4 is the same as 56,250 steps with per-GPU batch size 32. Therefore, in our training process, we use 10 training epoches, which is equivalent to 83,240 steps in total, and it is already more than the steps used in the original settings.

The optimizer in the training process for the ANCE-PRF model reported in the original study is the LAMB optimizer, which we did not notice at first, instead we used the AdamW optimizer which leads to slightly lower effectiveness in the replication as shown in Table 4.2.

A common practice for training new models based on another model is to re-initialise the linear head layer and train from scratch while keep the model body, which is exactly what we have done at first, but it appears that ANCE-PRF model is trained with everything inherited from the ANCE model, including the embedding head and normalisation (linear head layer), so without keeping the linear head layer from ANCE, our trained ANCE-PRF is significantly worse than the ANCE-PRF model with inherited linear head layer, as shown in Table 4.3.

With other models such as RocketQA [168], uniCOIL [111], in-batch negatives help the model to learn and achieves better performance. However, in our experiments, in-batch negatives does not help to improve the model performance, as shown in Table 4.3, the difference between using in-batch negatives and without using in-batch negatives are not significant.

Table 4.3: **Initial** represents the results by re-initialising the linear head layer. **Inherit** represents the results by inheriting the linear head layer from ANCE. **In Batch** represents the results by using in batch negatives. **No In Batch** represents the results by not using in batch negatives. **1e-6** represents the results by using 1e-6 as the learning rate. **1e-5** represents the results by using 1e-5 as learning rate. Best results are marked with **Bold**. Statistical significance is marked with † ($p < 0.05$, two tails paired t-test)

Dataset	MS MARCO	TREC DL 2019		TREC DL 2020	
	MRR@10	nDCG@10	R@1000	nDCG@10	R@1000
Initial	0.3130	0.6310	0.7100	0.6270	0.7310
Inherit	**0.3350**†	**0.6800**†	**0.7980**†	**0.7030**†	**0.8160**†
In Batch	**0.3470**	0.6720	**0.7970**	**0.7030**	0.8140
No In Batch	**0.3470**	**0.6730**	0.7940	0.7010	**0.8160**
1e-6	0.3350	**0.6800**	**0.7980**	**0.7050**	**0.8170**
1e-5	**0.3470**†	0.6730	0.7940	0.7010	0.8160

Table 4.4: We report results obtained by training two stronger dense retrievers using the same training procedure as ANCE-PRF. For a direct comparison, all models were trained with a PRF depth of 3. Results that are statistically significant ($p < 0.05$) compared to their corresponding base models without PRF are marked with †. Best results are marked with **Bold**.

Datasets	MS MARCO			TREC DL 2019		TREC DL 2020	
	MRR@10	nDCG@10	R@1000	nDCG@10	R@1000	nDCG@10	R@1000
ANCE [223]	0.3300	0.3880	**0.9590**	0.6480	0.7550	0.6460	0.7760
ANCE-PRF 3 [234]	**0.3440**	**0.4010**	**0.9590**	**0.6810**	**0.7910**	**0.6950**	**0.8150**
TCT ColBERT V2 HN+	**0.3590**	**0.4200**	0.9700	0.7200	0.8260	0.6880	**0.8430**
TCT ColBERT V2 HN+ PRF 3	0.3570	0.4180	**0.9710**†	**0.7410**	**0.8520**†	**0.7120**†	0.8400
DistilBERT KD TASB	0.3440	0.4070	**0.9770**	0.7210	0.8410	0.6850	**0.8730**
DistilBERT KD TASB PRF 3	**0.3480**	**0.4110**†	0.9740	**0.7360**	**0.8570**†	**0.6980**†	0.8660

Learning rate also plays an important part in the training process, we have experimented with two different learning rates, 1e-5 and 1e-6, the results are shown in Table 4.3. By using a larger learning rate, it tends to improve the MRR@10 for MS MARCO dataset, and with a smaller learning rate, it tends to improve nDCG@10 and R@1000 in TREC DL 2019 and TREC DL 2020. However, only the MRR@10 difference is significant.

In addition to the settings above, we also experimented with different number of hard negatives from the ANCE initially retrieved results. More specifically, we tried to sample 2, 8, and 21 negative samples from the top 200 or top 1000 results, the outcome meets our expectation, with 21 negative samples from top 200 giving the best performance.

To answer **RQ1.2.2**, some hyperparameters, such as learning rate, number of negatives, and optimizer, are crucial for reproducing the model checkpoint.

Table 4.5: Impact of inference-time PRF depth on model generalizability. We evaluate an ANCE-PRF model trained with a fixed depth of $k = 3$ while varying the PRF depth during inference ($k \in \{1, 3, 5\}$). Results that are statistically significant ($p < 0.05$) compared to the ANCE baseline are marked with †. Best results are marked with **Bold**.

Datasets	MS MARCO			TREC DL 2019		TREC DL 2020	
	MRR@10	nDCG@10	R@1000	nDCG@10	R@1000	nDCG@10	R@1000
ANCE [223]	0.3300	0.3880	0.9590	0.6480	0.7550	0.6460	0.7760
ANCE-PRF 3 (Inf Depth 1)	0.3320	0.3900	0.9590	0.6490	0.7550	0.6460	0.7760
ANCE-PRF 3 (Inf Depth 3)	**0.3470**†	**0.4050**†	**0.9630**	**0.6720**	**0.7940**†	**0.7010**†	**0.8140**
ANCE-PRF 3 (Inf Depth 5)	0.3270	0.3880	0.9570	0.6400	0.7510	0.6470	0.7790

4.3.3 RQ1.2.3: Generalisability of ANCE-PRF Beyond ANCE

After replicating the ANCE-PRF model, we applied the same PRF strategy to other dense retrievers to assess its effectiveness gains. However, this improvement is of a smaller magnitude than that observed for ANCE, which can be observed from Table 4.4. This may be due to (1) The best hyperparameter settings for ANCE-PRF may not be adequate to generalise to other dense retrievers, and different settings may lead other dense retrievers to obtain larger improvements; this speaks to the limited robustness of ANCE-PRF's training strategy. (2) The dense retrievers we consider, TCT ColBERT V2 HN+[114] and DistilBERT KD TASB [73] are more effective than ANCE. The limited improvement then may be due to the fact that it is easier to improve a weaker model (ANCE) than it is to improve a more effective one (TCT ColBERT V2 HN+ and DistilBERT KD TASB).

To answer **RQ1.2.3**, we find that applying the same training strategy as ANCE-PRF to other more effective dense retrievers provides little improvement. Hence, the ANCE-PRF method may not generalize to other dense retrievers or requires specific expensive and time-consuming hyperparameter tuning.

To further assess the generalisability of the ANCE-PRF training strategy, we evaluated a trained ANCE-PRF model (trained with PRF depth $k = 3$) by applying it at different PRF depths during inference, as shown in Table 4.5. The results consistently showed either significantly degraded or unchanged effectiveness compared to the original ANCE model when the inference depth deviated from the training configuration. These findings indicate that the ANCE-PRF query encoder is highly sensitive to the depth at which it was trained, and its representations do not transfer effectively across different PRF depths. Consequently, a separate ANCE-PRF query encoder must be trained for each desired PRF depth, limiting the flexibility and practicality of this approach. This observation highlights a core limitation of ANCE-PRF and motivates the need for alternative methods that are agnostic to depth and do not require retraining.

Finally, consistent with the thesis's focus on the trade-offs between effectiveness and efficiency, we analyze the computational cost of the ANCE-PRF. We measured query latency on a single NVIDIA A100 GPU using the TREC DL 2019 query set (43 queries). While the baseline ANCE retriever operates with an average latency of approximately 61ms per query, the full ANCE-PRF pipeline with $k = 3$ increases this to approximately 249ms, $\approx$ 4x increase. Decomposing this overhead reveals that

the PRF encoding stage (with $k = 3$) accounts for the majority of the cost ($\approx$ 124ms), followed by the second round of retrieval ($\approx$ 64ms). As feedback depth increases, latency rises until the input reaches the length limit (approximately 5-7 passages). Beyond this point, additional passages are truncated, causing latency to plateau (see Figure 8.3 in Chapter 8). This substantial increase in latency highlights the computational expense required to process concatenated query-passage sequences within the feedback encoder, highlighting a critical limitation of learned PRF methods for latency-sensitive applications.

4.4 Summary

In this chapter, we investigated the ANCE-PRF model proposed by Yu et al. [234], a seminal approach that integrates pseudo-relevance feedback (PRF) directly into the dense retriever's query encoder without modifying the passage encoder or the underlying passage index. This design enables the model to encode PRF signals within a transformer-based query representation, aiming to improve retrieval effectiveness while preserving the efficiency advantages of dense retrieval.

Our analysis was guided by the overarching research question **RQ1.2: What are the challenges and limitations of integrating PRF into neural models?** To this end, we examined three sub-questions related to the reproducibility and generalisability of the ANCE-PRF method.

RQ1.2.1 explored the feasibility of reproducing inference results using the original ANCE-PRF model checkpoint. We identified a previously undocumented preprocessing constraint: the model is trained as an uncased encoder, and thus, inconsistencies in letter casing at inference time significantly impacts performance. This highlights the importance of precise documentation and preprocessing alignment in neural IR reproducibility.

RQ1.2.2 examined the reproducibility of the ANCE-PRF training pipeline. Our initial attempts, based on details from the original paper and common practices (e.g., reinitialising the linear head layer), yielded suboptimal results. After consulting the authors and adhering strictly to their training setups, including retaining the original ANCE linear head and omitting in-batch negatives, we successfully reproduced the model with comparable performance. These findings reveal a key limitation: ANCE-PRF's effectiveness is highly sensitive to training setup and hyperparameters, limiting its robustness and ease of adaptation.

RQ1.2.3 evaluated the generalisability of the ANCE-PRF training strategy to more recent and effective dense retrievers, namely TCT-ColBERT V2 HN+[114] and DistilBERT KD TASB[73]. While marginal improvements were observed, the performance gains were consistently smaller than those obtained with ANCE. This outcome indicates that the incremental benefits of the ANCE-PRF training paradigm appear to diminish when applied to stronger dense retrieval backbones, in contrast to the more substantial gains observed with the ANCE baseline. Additionally, the need to retrain a separate ANCE-PRF encoder for each desired PRF depth further reduces its practicality for deployment. Furthermore, processing concatenated query-passage sequences within the feedback encoder incurs a 4x increase in latency compared to the ANCE retriever, further reducing its practicality.

In summary, our reproduction and analysis of ANCE-PRF reveal several challenges associated with integrating PRF into dense neural models: sensitivity to preprocessing, dependence on non-trivial training configurations, lack of generalisability, incur high latency, and inflexibility with respect to PRF depth. These findings re-emphasize the limitations and challenges identified along with Text-based PRF in the previous chapter and motivate the need for alternative strategies that are model-agnostic, training-free, and scalable.

In the next chapter, we propose our Vector-based PRF methods. These approaches operate directly in the embedding space, offering efficient lightweight mechanisms for incorporating PRF signals without retraining or modifying the original retrieval architecture. In doing so, we aim to address the key limitations observed in both Text-based PRF and the ANCE-PRF framework.

The code and implementation details for all ANCE-PRF reproduction experiments are made publicly available at: `https://github.com/ielab/APR` and `https://github.com/castorini/pyserini/blob/master/docs/experiments-ance-prf.md`

Chapter 5

Vector-Based Pseudo-Relevance Feedback For Dense Retrieval

In the previous chapters, we explored different strategies for integrating PRF into neural information retrieval models. Specifically, we examined Text-based PRF methods on top of neural rerankers, and ANCE-PRF, which incorporates a learned PRF-aware query encoder with dense retrieval frameworks. However, both revealed challenges and limitations, including generalisability, deployability, efficiency, and limited improvement on effectiveness even with learned models. In this chapter, to address the challenges and limitations, we also adopt the dense retriever models to generate embeddings to represent text [72, 73, 114, 115, 173, 223, 238]. As discussed in Chapter 4, for dense retrievers, query latency is reduced to the time of generating the query embeddings because the passages embeddings are pre-generated. In the context of PRF, we further utilise these pre-generated passages embeddings to efficiently integrate the relevance signals while eliminating the input size limit of deep language models, which we refer to as *Vector-based* PRF (VPRF) approach. Each feedback passage is pre-generated as embeddings (vectors) in this approach, and VPRF directly manipulates the query embedding with feedback passage embeddings in the vector space to refine the query representation. Furthermore, we adopt two different vector fusion methods (Average and Rocchio) to integrate the feedback vectors into the query vectors. The Rocchio method is inspired by the classic Rocchio [179] method, our Rocchio approach has two parameters: the query vector and the feedback passage vector weights. We empirically investigate the influence of query and feedback passages through weighting within the Rocchio PRF approach.

We are aiming to address the research questions:

RQ1.3: Does Vector-based Pseudo-Relevance Feedback provides better query representations?

RQ1.4: What are the trade-offs in Pseudo-Relevance Feedback integration strategies?

For this Vector-based PRF approach, we find that our models improve the respective baselines (seven dense retrievers) across all evaluation metrics and all datasets for the retrieval task; the proposed approach also outperforms the strong BM25+BERT ranker across several metrics. This result suggests

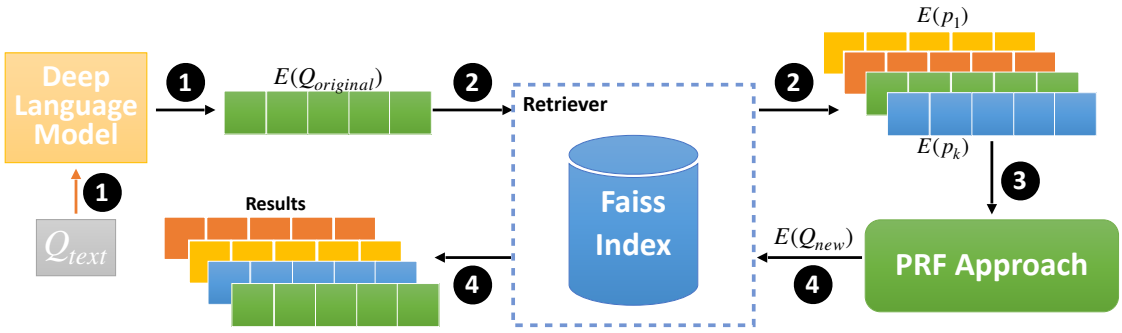


Figure 5.1: The proposed architecture for integrating Vector-based Pseudo-Relevance Feedback with Deep Language Model dense retrievers for the retrieval task.

that encoding the PRF feedback passages into embedding vectors better models the relevance signals exploited by the PRF mechanism. Unlike Text-based PRF, the passage vectors are pre-generated and indexed, so the inference steps on passages are not required at retrieval or rerank time. This makes Vector-based PRF very efficient: it takes only 1/20th of the time of the BM25+BERT reranker and only about double the time of the simple bag-of-words BM25. In addition, since our proposed approach works directly with the vector embeddings of queries and passages without further training or fine-tuning of the backbone models, they can be applied on top of any choice of dense retriever. For the reranking task, we find that our models outperform the BM25 and BM25+RM3 baselines across all metrics and datasets, while they have only mixed improvements over the strong BM25+BERT reranker. Overall, the Vector-based PRF approach for retrieval tends to improve deep metrics, while for reranking, they tend to improve shallow metrics.

5.1 Vector-Based Pseudo-Relevance Feedback

Figure 5.1 illustrates the proposed architecture for integrating Vector-based PRF signals into dense retrieval with deep language models. The top portion shows the standard dense retrieval pipeline (Steps ❶ – ❷), while the bottom portion depicts the extended Vector-based PRF process (Steps ❸ – ❹).

A deep language model is used to generate embeddings for all passages offline, which are stored in a FAISS index [41]. At inference time, the same model encodes the query into a dense embedding (❶). This embedding is then used by the dense retriever to search the FAISS index and return an initial ranked list (❷).

From this list, the top-k passage embeddings are selected as pseudo-relevance feedback signals (❸). These are combined with the original query embedding through vector operations (e.g., Rocchio fusion), yielding an updated query representation. This updated query is then used to retrieve a final ranked list from the same index (❹).

Using 7 existing, efficient first stage dense retrievers (Details in Section 5.3.3), we employ two Vector-based PRF methods for the retrieval task, and we describe the two proposed Vector-based PRF approaches in the next subsections.

5.1.1 Vector-Based PRF with Average

A new query embedding is generated by averaging the original query embedding and the top-k feedback passage embeddings. The intuition is to treat the original query at par of the signal from the top-k feedback passages (i.e., the query weights as much as each passage). The new query embedding is computed as follows:

$$\mathbf{q}_{new} = Avg(\mathbf{q}_{original}, \mathbf{p}_1, ..., \mathbf{p}_k) \tag{5.1}$$

where $\mathbf{q}$ and $\mathbf{p}$ represent the embeddings of the query and the feedback passage, respectively. $\mathbf{q}_{new}$ represents the newly formulated query embeddings. We do not generate an actual text query in the Vector-based PRF approaches: only the embedding of the new query is generated. $\mathbf{q}_{original}$ represents the original query, $\mathbf{p}_1, ..., \mathbf{p}_k$ represent the top-k passages retrieved by the first stage ranker. In the remainder of the paper we refer to this method as Vector Average, represented by V-A for brevity.

5.1.2 Vector-Based PRF with Rocchio

This method is inspired by the original Rocchio method for relevance feedback [179] but adapted to deep language models. The intuition is to transform the original query embedding towards the average of the top-k feedback passage embeddings by assigning different weights to query and (the combination of) feedback passages, thus controlling the contribution of each component toward the final score. Unlike in the original version of Rocchio, in this work we do not model the PRF with non-relevant passages: hence the negative portion of Rocchio is omitted. We note that this could be extended by identifying which passages in the initial ranked list could represent a negative relevance signal (e.g., the bottom passages) – however we leave this for future consideration.

Thus, our Rocchio PRF approach consists of interpolating the query embedding and the average PRF embedding:

$$\mathbf{q}_{new} = \alpha * \mathbf{q}_{original} + \beta * Avg(\mathbf{p}_1, ..., \mathbf{p}_k) \tag{5.2}$$

where α controls the weight assigned to the original query embedding and β the weight assigned to the PRF signal. In the remainder of the paper we refer to this method as Rocchio, represented by $\mathcal{RC}_{\alpha}$ and $\mathcal{RC}_{\alpha,\beta}$ for brevity.

5.2 Hybrid Pseudo-Relevance Feedback

Text-based PRF is a computationally expensive approach for the reranking task as we have investigated in Chapter 3, where the BERT inference step is executed twice: one before the PRF, one after the PRF. On the other hand, Vector-based PRF is an efficient approach for the retrieval task because of the high

efficiency of the dense retriever models. In this chapter, we further investigate a hybrid approach where the architecture of Vector-based PRF in Figure 5.1 is adapted to the reranking task. The main difference is that the initial ranking of passages is obtained from an inverted-index (Text-based) multi-stage pipeline such as BM25+BERT (as in Figure 3.1). In particular, the initial retrieval results obtained through steps ❶ and ❷ in Figure 5.1 are replaced by steps ❶, ❷, ❸, and ❹ in Figure 3.1. The ranked list of passages produced by the BERT reranker is mapped to embeddings using the FAISS index before applying the Vector-based PRF methods.

5.3 Empirical Evaluation

To guide our empirical investigation of Vector-based PRF approaches, we pose the following research questions:

RQ1.3.1: What is the impact of PRF depth on the effectiveness of Vector-based PRF with dense retrievers?

RQ1.3.2: What is the impact of dense representation on the effectiveness of Vector-based PRF with dense retrievers?

RQ1.3.3: What is the impact of vector fusion method on the effectiveness of Vector-based PRF with dense retrievers?

RQ1.3.4: What is the overall effectiveness of Vector-based PRF models on the task of retrieval?

RQ1.3.5: What is the impact of PRF on the efficiency with neural models?

RQ1.4: What are the trade-offs when integrating Text-based and Vector-based PRF into neural models?

The implementation of Vector-based PRF and the baselines evaluated in this chapter is available at: `https://github.com/ielab/Neural-Relevance-Feedback-Public/tree/master/Vector_Based_PRF`.

5.3.1 Datasets and Evaluation Metrics

Our experiments use the same datasets as Text-based PRF experiments in Section 3.2.1 from Chapter 3: TREC Deep Learning Track Passage Retrieval Task 2019 [27] (DL 2019) and 2020 [26] (DL 2020), DL HARD [137], the TREC Conversational Assistance Track 2019 [35] (CAsT 2019), and the Web Answer Passages (WebAP) [84].

For the retrieval task with Vector-based PRF, we report MAP, nDCG@{1, 3, 10}, Reciprocal Rank (RR), and Recall@1000. For the reranking task involving Vector-based and hybrid PRF, we include Success@1, but Recall@1000 is excluded. Recall is included in retrieval evaluation to help distinguish whether gains in metrics like MAP result from retrieving more relevant passages or from

better ranking of an existing set. All evaluations follow the official TREC DL 2019 and 2020 guidelines, and statistical significance is assessed using a two-tailed paired t-test with Bonferroni correction.

5.3.2 Baselines

We consider the following baselines (including baselines from Text-based PRF (Section 3.2.2) for further comparison between Text-based and Vector-based PRF approaches):

- `BM25`: traditional first stage retriever, implemented using the Anserini toolkit [229] with its default settings ($k_1 = 0.9$, $b = 0.4$ for BM25).
- `BM25+RM3`: RM3 pseudo relevance feedback method [1] on top of BM25, as implemented in Anserini. We use this approach as a representative bag-of-words PRF method, since previous research has found alternative bag-of-words PRF approaches achieve similar effectiveness [146]. We note that BM25+RM3 is a standard baseline for MS MARCO and TREC DL. We use default settings with 10 feedback terms, 10 feedback documents, original query weight 0.5.
- `RepBERT (R)`: first stage dense retriever [238]. We use the implementation made available by the authors.
- `ANCE (A)`: first stage dense retriever [223]. We use the scripts provided by the authors for both data pre-processing and model implementation.
- `TCT-ColBERT V1/V2 HN+, DistilBERT KD/Balanced, and SBERT`: first stage dense retrievers [72, 73, 114, 115, 173] employed to evaluate the generalisability of our method. We use the implementations provided in the pyserini toolkit [112].
- `RepBERT+BERT (R+B)`: first stage dense retriever with an additional BERT reranker to rerank the initial results provided by RepBERT.
- `ANCE+BERT (A+B)`: first stage dense retriever with an additional BERT reranker to rerank the initial results provided by ANCE.
- `BM25+BERT (BB)`: A common two-stage reranker pipeline, first proposed by Nogueira and Cho [155], where the initial stage is BM25, and BERT is used to rerank the results from BM25. BERT is fine-tuned on MS MARCO Passage Retrieval Dataset [151]. In all of our experiments, we use the 12 layer uncased BERT-Base provided by Nogueira and Cho [155], unless stated otherwise, and we simply refer to it as BERT. In Section 5.4.5 we also use BERT-Large for the efficiency analysis.

5.3.3 Applying Vector-Based PRF to Retrievers

We choose RepBERT [238], ANCE [223], TCT-ColBERT V1 [114], TCT-ColBERT V2 HN+ [115], DistilBERT KD [72], DistilBERT Balanced [73], and SBERT [173] as representative first stage dense

retrievers because they achieve state-of-the-art effectiveness in previous works on MS MARCO. We note that a host of alternative first stage dense retrievers have been recently proposed, including stronger ones like RocketQA [168] and RocketQAv2 [175], but most of these retrievers consider more complex training procedures than those selected in this chapter. We further note that the implementation of the best first stage dense retriever when we proposed Vector-based PRF, RocketQAv2, has only just been made available and is based on PaddlePaddle [129] toolkit, thus uses a setup that differs from ours and is not selected for simplicity. We expect that findings that apply for the dense retrievers we chose are likely to translate to other dense retrievers, like RocketQA and RocketQAv2.

For the dense retrievers, we utilise the FAISS toolkit [79] to build the index and perform retrieval. We develop our PRF approaches on top of these dense retrievers. To be consistent with the original dense retriever models, we truncate the query tokens and passage tokens according to the original settings in their papers. For simplicity, we mainly investigate our proposed Vector-based PRF models on top of ANCE and RepBERT; the rest of the models are only shown in Tables 5.6 and 5.7 for validation purposes as well as a demonstration of the generalisability of our proposed models. Therefore, in the following sections, Vector-based PRF with RepBERT as base model is represented by R+PRF-R, and with ANCE as base model is represented by A+PRF-A for the retrieval task.

5.3.4 Applying Vector-Based PRF to Rerankers

We further consider the vector representations (embeddings) generated by RepBERT and ANCE to apply PRF as a second stage ranker, represented as BB+PRF-R and BB+PRF-A, where BB represents BM25+BERT, R represents RepBERT, and A represents ANCE. To achieve this, the top-k passages IDs from BERT are mapped to their vector representations before estimating the final scores.

5.3.5 Hyperparameter Settings for Vector-based PRF

For our Vector-based PRF methods (Average and Rocchio), we perform a grid search on the validation set (MS MARCO Dev set) to select the optimal parameters. For Rocchio with varying feedback passage weight, we fix $\alpha = 1.0$ and tune $\beta \in [0.1, 1.0]$. For Rocchio with varying both query and feedback passage weights, we tune both α and β in the range $[0.1, 1.0]$ with a step size of 0.1. We use top $k = \{1, 3, 5, 10\}$ passages as pseudo-relevance feedback passages for all VPRF methods in our experiments.

5.3.6 Efficiency Experiments

To evaluate the runtime of each method, we conduct our experiments on consistent hardware configurations. For BM25 and BM25 + RM3, we use a Unix-based server with an Intel(R) Xeon(R) Gold 6132 CPU @ 2.60GHz. For dense retrievers and our proposed approaches, experiments are conducted on a Unix-based server equipped with a single Tesla V100 SMX2 GPU with 32GB memory. To ensure a

fair comparison, these hardware settings are identical to those used in the Text-based PRF experiments reported in Section 3.2.4 of Chapter 3.

5.4 Results

In this section, we also include results from Text-based PRF approaches (from Section 3.3 in Chapter 3) where necessary to facilitate a more comprehensive comparison with our Vector-based PRF methods. While the primary focus remains on evaluating and analyzing the effectiveness of Vector-based PRF across different configurations and tasks, incorporating Text-based PRF results offers valuable context for understanding relative performance trends. Specifically, this allows us to compare the two approaches in terms of efficiency, robustness to query drift, and effectiveness across ranking metrics.

Unless otherwise stated, we report the test set performance of the configuration that achieved the highest nDCG@10 on the validation set. This protocol ensures that our conclusions regarding "superiority" are based on a consistent selection criterion (effectiveness on held-out data) rather than post-hoc selection of the best test result.

5.4.1 Impact of PRF Depth on the Effectiveness of Vector-Based PRF with Dense Retrievers

RQ1.3.1 examines how varying the depth of PRF influences effectiveness. To address this question, we vary the number of top-k feedback passages used in the Vector-based PRF process and evaluate performance across multiple metrics.

Retrieval With Different PRF Depths For Vector-Based PRF

The results for Vector-based PRF methods, specifically, R+PRF-R and A+PRF-A, are presented in Figures 5.2 – 5.6. For deep evaluation metrics such as MAP, nDCG@10, and Recall@1000, increasing PRF depth consistently leads to substantial effectiveness gains over the baseline dense retrievers across all datasets, with the exception of TREC DL HARD (Figure 5.6), where the improvements are marginal. These results suggest that incorporating more feedback passages can help refine the query representation and retrieve a broader range of relevant content, particularly in well-annotated datasets. However, for shallow metrics such as Reciprocal Rank (RR), increased PRF depth generally results in diminished performance, indicating a degradation in early precision. This suggests that deeper PRF signals may dilute the original query intent, potentially due to the inclusion of less relevant or noisy feedback passages. An exception is observed with A+PRF-A at a PRF depth of 10, which yields comparable or slightly better RR values than the baseline, suggesting that ANCE-based representations may be more robust to increased PRF depth under certain conditions. This reinforces the importance of carefully selecting PRF depth to balance deep relevance coverage with early precision.

The effect of PRF depth on nDCG@1, 3 varies across datasets, aligning with established findings in selective query expansion which posit that PRF effectiveness is bounded by the quality of the

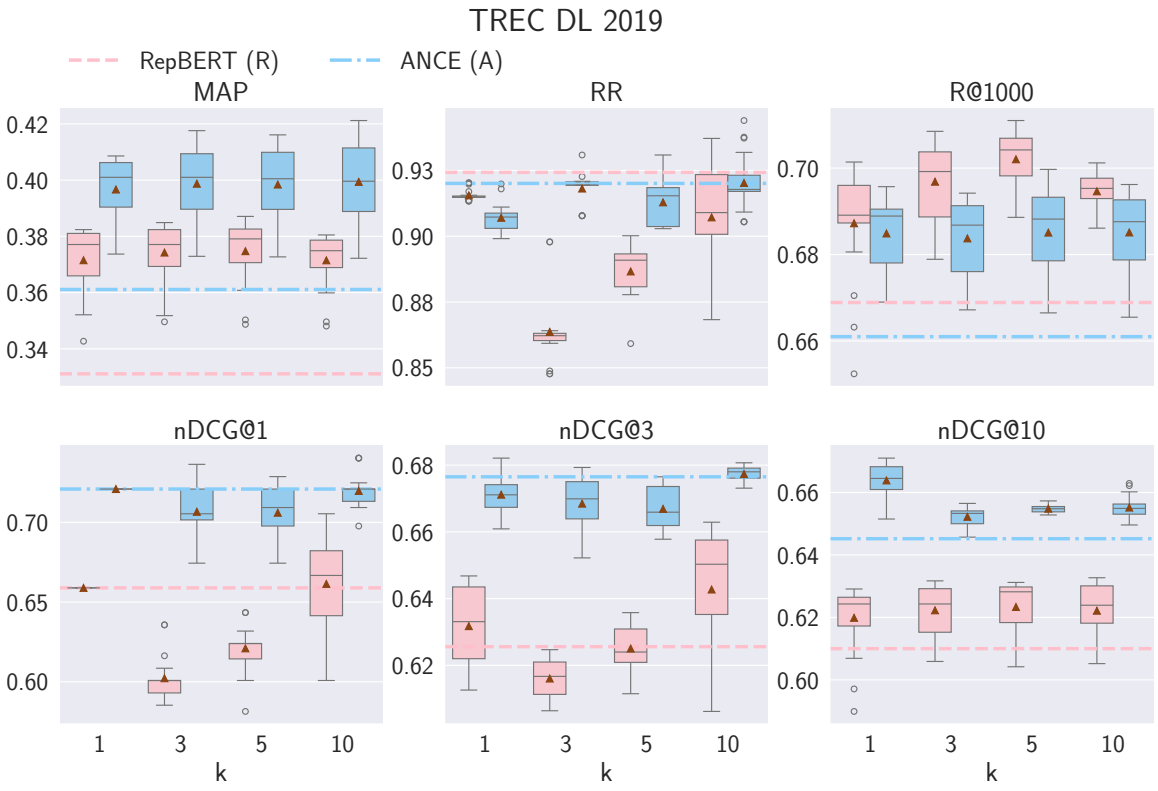


Figure 5.2: Impact of PRF depth on the effectiveness (y-axis) of RepBERT+PRF-RepBERT(R+PRF-R) and ANCE+PRF-ANCE(A+PRF-A) for the task of retrieval on TREC DL 2019, *k* represents different PRF depths. Baseline RepBERT(R) is marked with a dashed red line, ANCE(A) is marked with a dash-dot blue line.

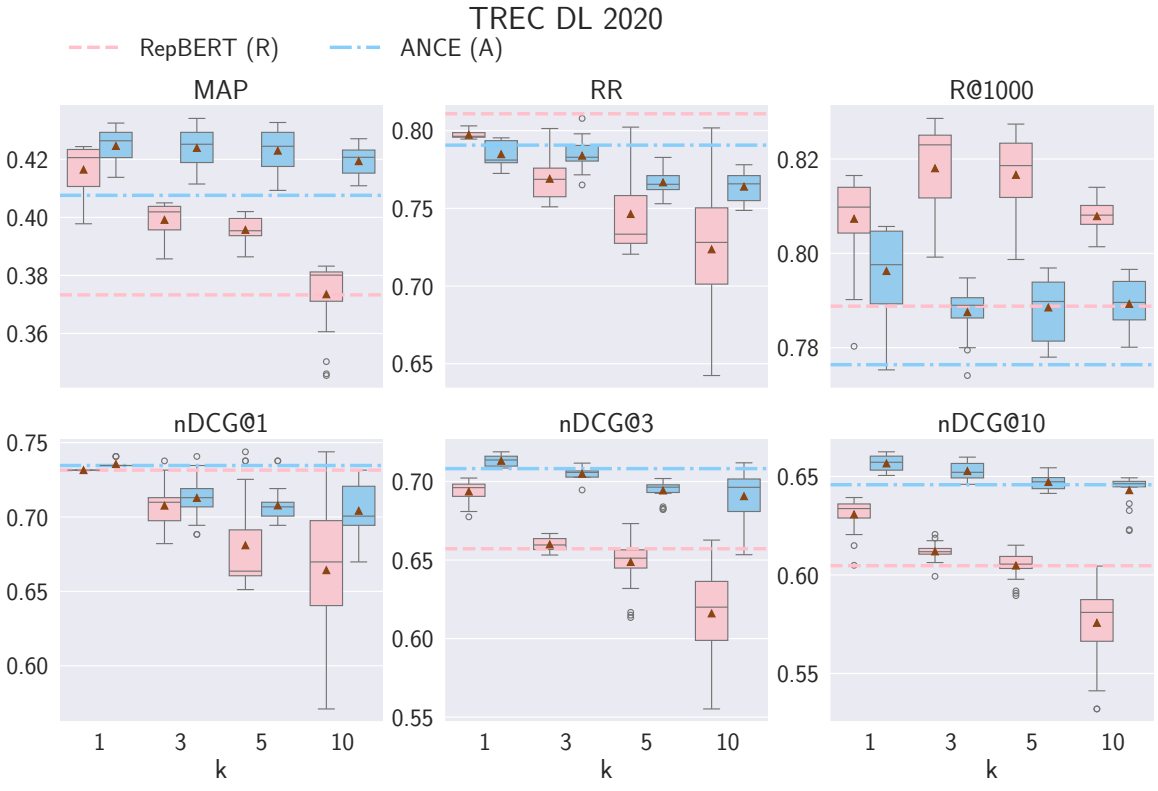


Figure 5.3: Impact of PRF depth on the effectiveness (y-axis) of RepBERT+PRF-RepBERT(R+PRF-R) and ANCE+PRF-ANCE(A+PRF-A) for the task of retrieval on TREC DL 2020, *k* represents different PRF depths. Baseline RepBERT(R) is marked with a dashed red line, ANCE(A) is marked with a dash-dot blue line.

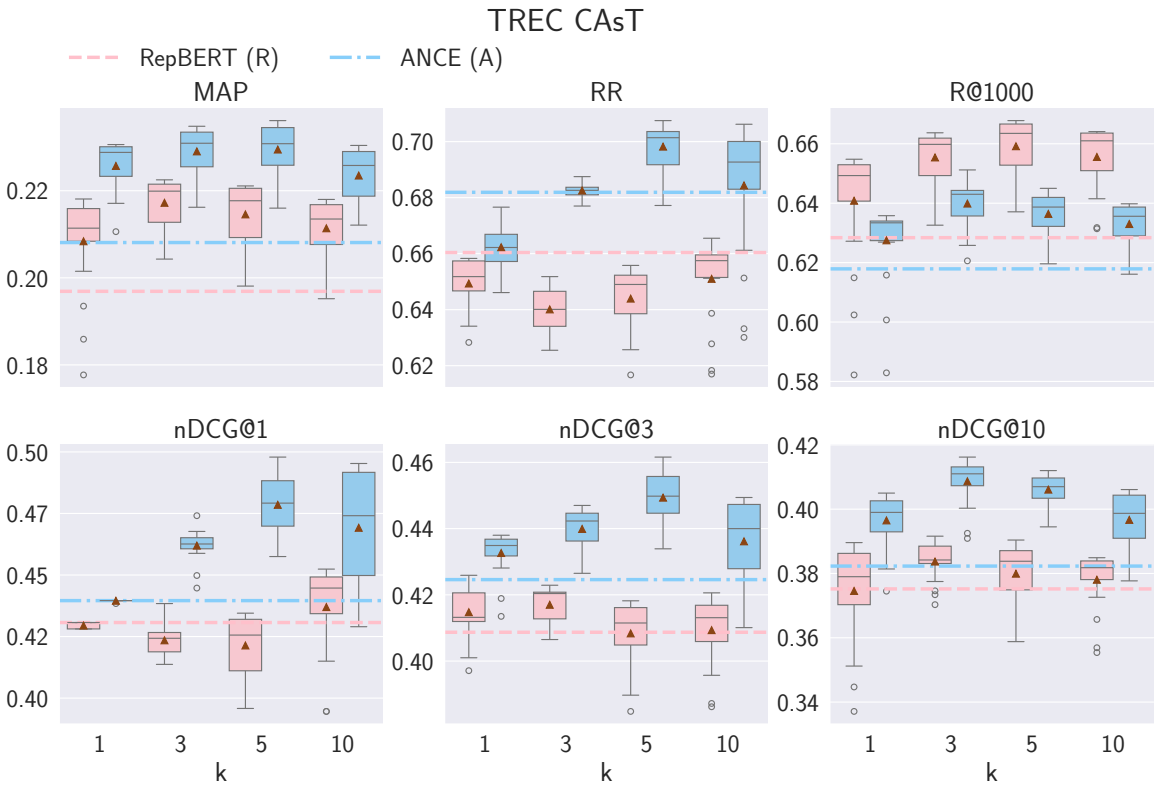


Figure 5.4: Impact of PRF depth on the effectiveness (y-axis) of RepBERT+PRF-RepBERT(R+PRF-R) and ANCE+PRF-ANCE(A+PRF-A) for the task of retrieval on TREC CAsT, *k* represents different PRF depths. Baseline RepBERT(R) is marked with a dashed red line, ANCE(A) is marked with a dash-dot blue line.

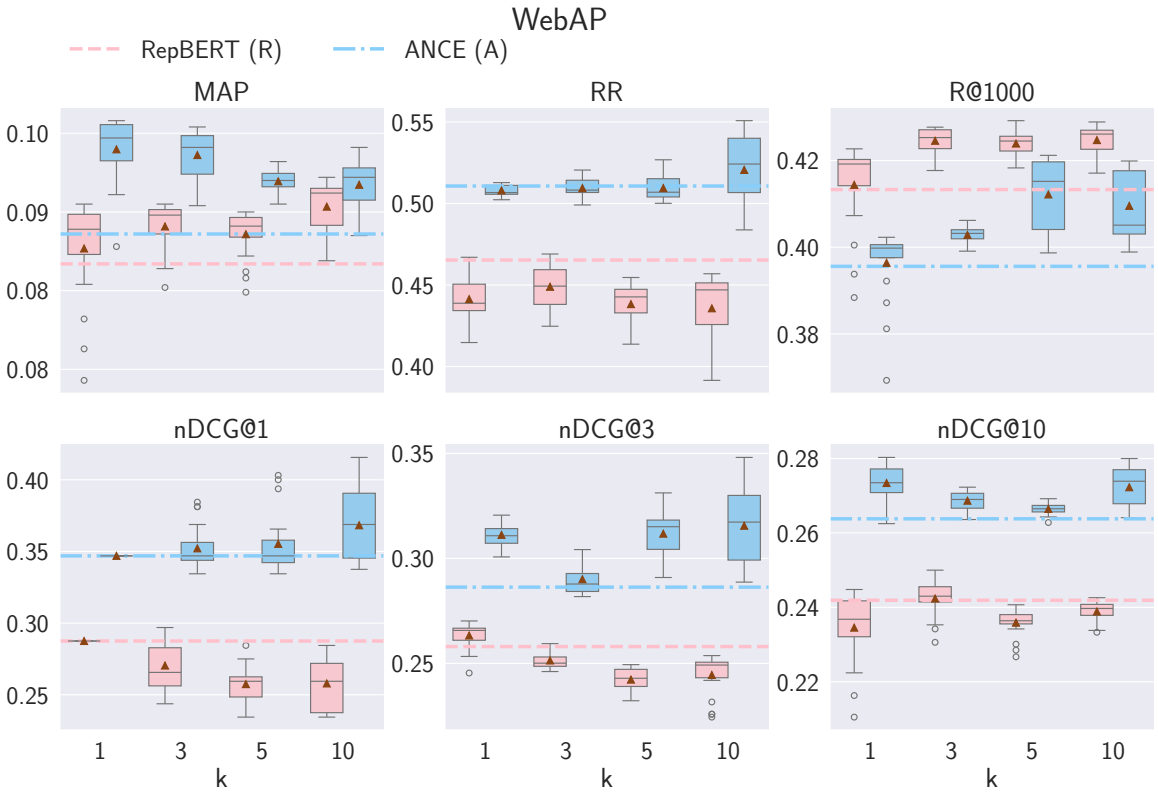


Figure 5.5: Impact of PRF depth on the effectiveness (y-axis) of RepBERT+PRF-RepBERT(R+PRF-R) and ANCE+PRF-ANCE(A+PRF-A) for the task of retrieval on WebAP, *k* represents different PRF depths. Baseline RepBERT(R) is marked with a dashed red line, ANCE(A) is marked with a dash-dot blue line.

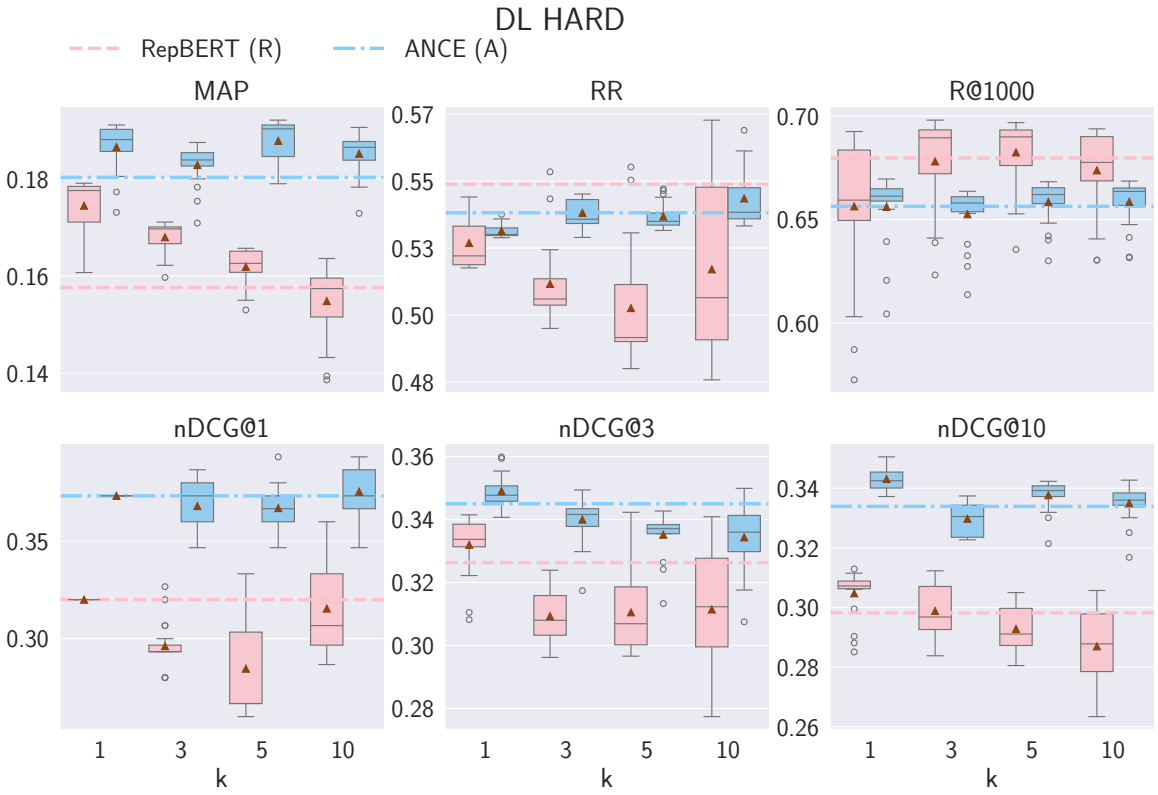


Figure 5.6: Impact of PRF depth on the effectiveness (y-axis) of RepBERT+PRF-RepBERT(R+PRF-R) and ANCE+PRF-ANCE(A+PRF-A) for the task of retrieval on DL HARD, *k* represents different PRF depths. Baseline RepBERT(R) is marked with a dashed red line, ANCE(A) is marked with a dash-dot blue line.

initial retrieval [18, 28]. On TREC DL 2019 (Figure 5.2) and 2020 (Figure 5.3), shallow PRF (depth 1) performs comparably to the baseline, indicating that minimal feedback may be sufficient to maintain early precision in these settings. In contrast, on TREC CAsT (Figure 5.4) and WebAP (Figure 5.5), deeper PRF yields noticeable improvements for A+PRF-A, suggesting that ANCE-based representations benefit from richer feedback signals in datasets with more diverse or conversational queries. For R+PRF-R, however, the best results in these datasets are generally achieved with PRF of depth 1, indicating a higher sensitivity to feedback noise, a known vulnerability of feedback mechanisms when the initial ranking lacks robustness [64]. This is most evident on DL HARD (Figure 5.6), a challenging retrieval setting where increasing PRF depth fails to contribute meaningfully to early precision. This obeservation suggests that when the initial signal is poor, increasing feedback depth rather amplifies noise than resolving ambiguity, confirming that dense retrieval models are not immune to the classic robustness trade-offs of PRF.

Reranking With Different PRF Depths For Vector-Based PRF and Comparison With Text-Based PRF

Furthermore, we report results for a second type of *hybrid* PRF models that combine dense retrievers with a BERT reranker (R+PRF+B and A+PRF+B), as shown in Figures 5.7 – 5.11. These models generally demonstrate either a degradation or only marginal improvement in RR across all datasets and

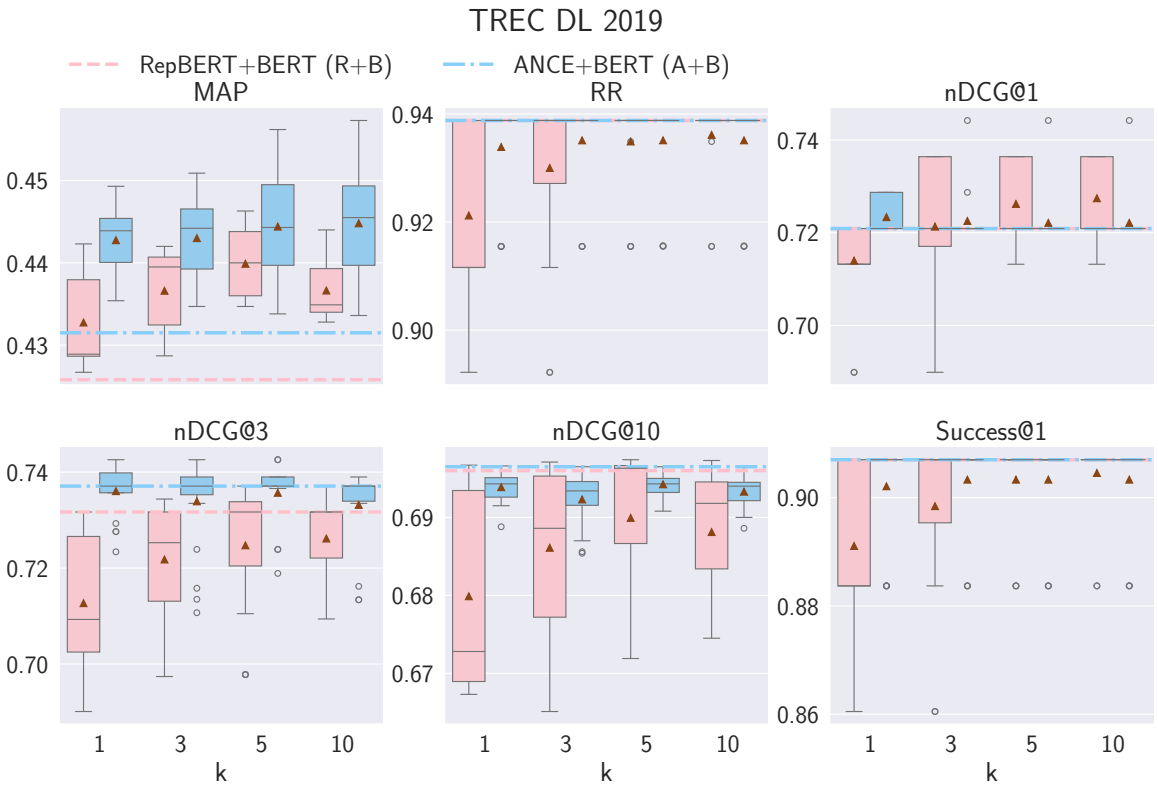


Figure 5.7: Impact of PRF depth on the effectiveness (y-axis) of ANCE+PRF+BERT(A+PRF+B) and RepBERT+PRF+BERT(R+PRF+B) for the task of reranking on TREC DL 2019, k represents the different PRF depths. Baseline ANCE+BERT(A+B) and RepBERT+BERT(R+B) are marked with a dash-dot blue line and a dashed red line respectively.

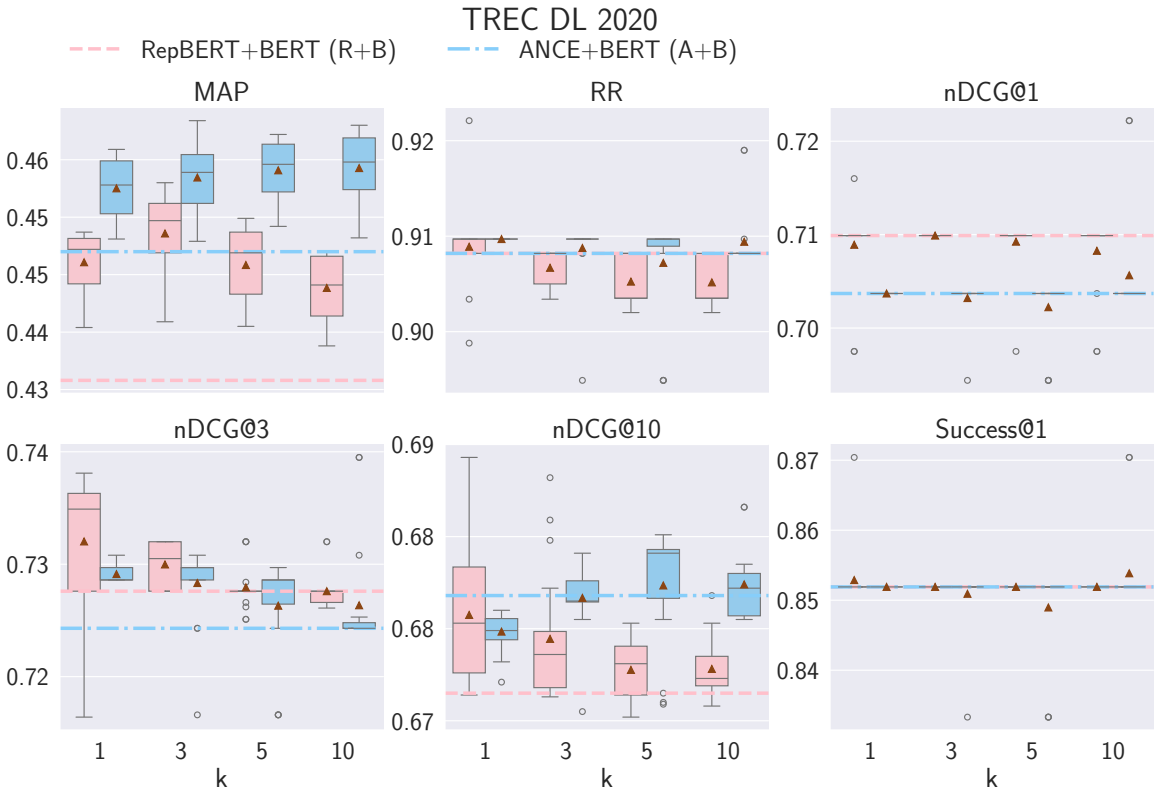


Figure 5.8: Impact of PRF depth on the effectiveness (y-axis) of ANCE+PRF+BERT(A+PRF+B) and RepBERT+PRF+BERT(R+PRF+B) for the task of reranking on TREC DL 2020, k represents the different PRF depths. Baseline ANCE+BERT(A+B) and RepBERT+BERT(R+B) are marked with a dash-dot blue line and a dashed red line respectively.

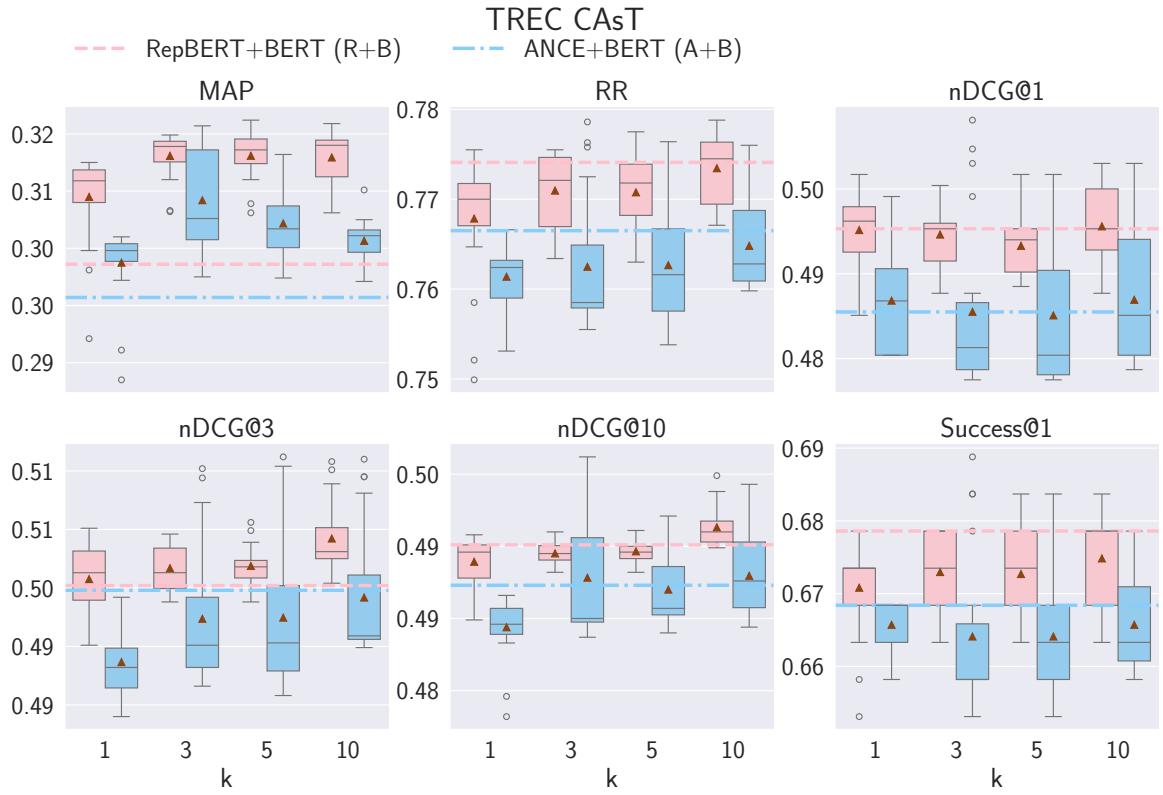


Figure 5.9: Impact of PRF depth on the effectiveness (y-axis) of ANCE+PRF+BERT(A+PRF+B) and RepBERT+PRF+BERT(R+PRF+B) for the task of reranking on TREC CAsT, *k* represents the different PRF depths. Baseline ANCE+BERT(A+B) and RepBERT+BERT(R+B) are marked with a dash-dot blue line and a dashed red line respectively.

PRF depths. This reinforces the principle that downstream reranking cannot fully compensate for poor upstream candidate generation; if the initial PRF-enhanced retrieval fails to surface relevant candidates, the reranker's potential is capped. In terms of MAP, however, all PRF variants show consistent improvements at PRF depths 3 – 5, highlighting that moderate-depth feedback contributes more meaningfully to overall relevance ranking. On TREC DL 2019 (Figure 5.7) and 2020 (Figure 5.8), ANCE-based variants slightly outperform RepBERT-based models, suggesting better underlying contextualization from ANCE-derived feedback to the BERT reranker. In contrast, on TREC CAsT (Figure 5.9), RepBERT-based models show a slight edge, possibly due to alignment between the conversational nature of the queries and the characteristics of the RepBERT model. For other datasets, including WebAP and DL HARD, both variants yield comparable performance across different PRF depths, with no substantial divergence. Across all metrics, the highest effectiveness is generally achieved at PRF depths 3 – 5, re-highlighting prior observations that moderate feedback depth achieves a balance between informative signals and noises. Nevertheless, the overall improvements remain modest, indicating diminishing returns from reranking in hybrid pipelines when the initial retrieval candidates are not significantly enhanced.

We also extend the analysis to the alternative hybrid configuration (BB+PRF-R and BB+PRF-A), presented in Figures 5.12 – 5.16. On TREC DL 2019 (Figure 5.12), increasing PRF depth offers substantial gains in RR and nDCG@1, with marginal improvements for nDCG across depths 3 to 10. These results indicate that in high-quality retrieval environments, hybrid reranking effectively leverages

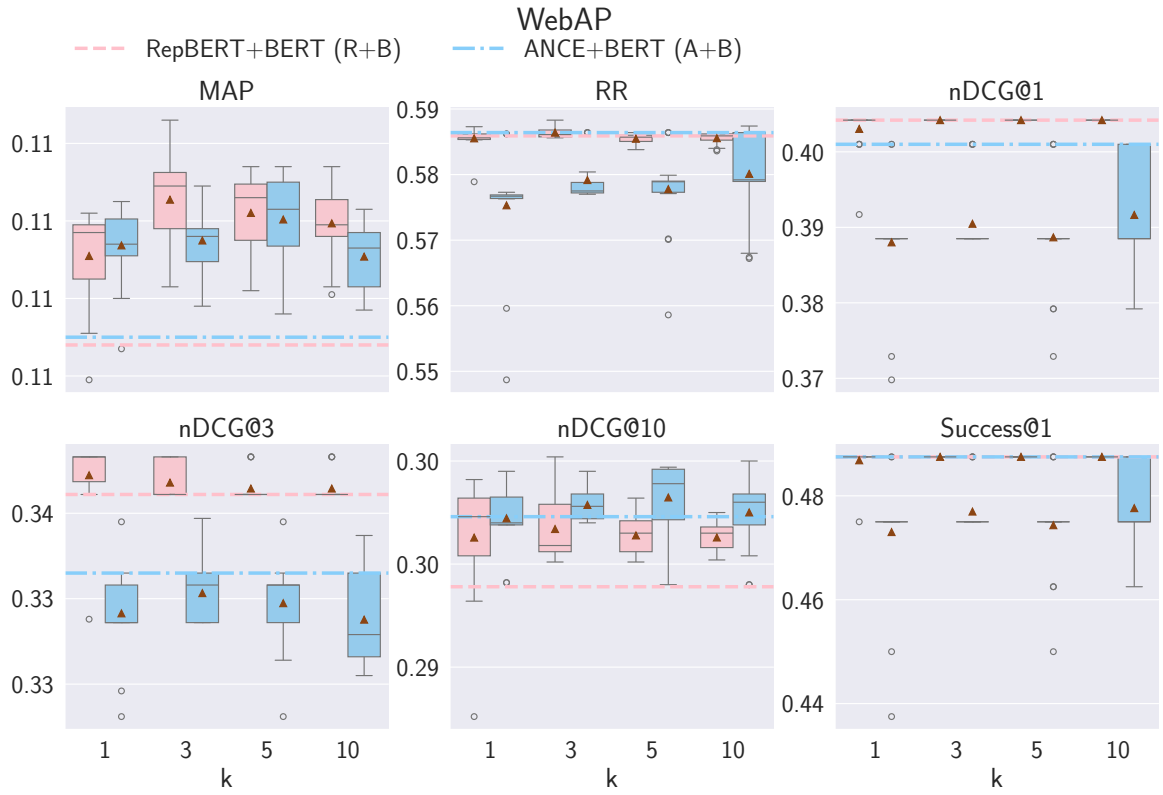


Figure 5.10: Impact of PRF depth on the effectiveness (y-axis) of ANCE+PRF+BERT(A+PRF+B) and RepBERT+PRF+BERT(R+PRF+B) for the task of reranking on WebAP, *k* represents the different PRF depths. Baseline ANCE+BERT(A+B) and RepBERT+BERT(R+B) are marked with a dash-dot blue line and a dashed red line respectively.

deeper feedback to optimize early precision. Similarly, for TREC DL 2020 and TREC CAsT, deeper PRF depths lead to modest but consistent improvements in nDCG@1, 3. However, a stark contrast is observed on WebAP and DL HARD, where increased PRF depth frequently triggers a decline in overall effectiveness. This degradation is consistent with the breakdown of the pseudo-relevance assumption in difficult retrieval scenarios: as the quality of the initial ranking drops, the feedback mechanism propagates noise rather than relevance signals. Therefore, the utility of hybrid reranking proves highly sensitive to dataset difficulty, reaffirming that without explicit quality estimation or selective expansion strategies, aggressive feedback incorporation can induce and amplify query drift.

Summary For RQ1.3.1

To summarize, Vector-based PRF demonstrates consistent and scalable performance characteristics, particularly in its ability to enhance both retrieval and hybrid reranking tasks. Increasing PRF depth typically improves effectiveness for hybrid models, especially on deeper metrics such as Recall@1000, nDCG@10, and MAP, while also offering moderate gains in shallow reranking metrics like RR and nDCG@{1, 3}. When used alone for retrieval, the optimal performance often achieved at moderate depths (e.g., 3 – 5). When combined with a BERT reranker, Vector-based PRF yields modest but consistent MAP improvements across datasets, though benefits on other metrics are generally limited

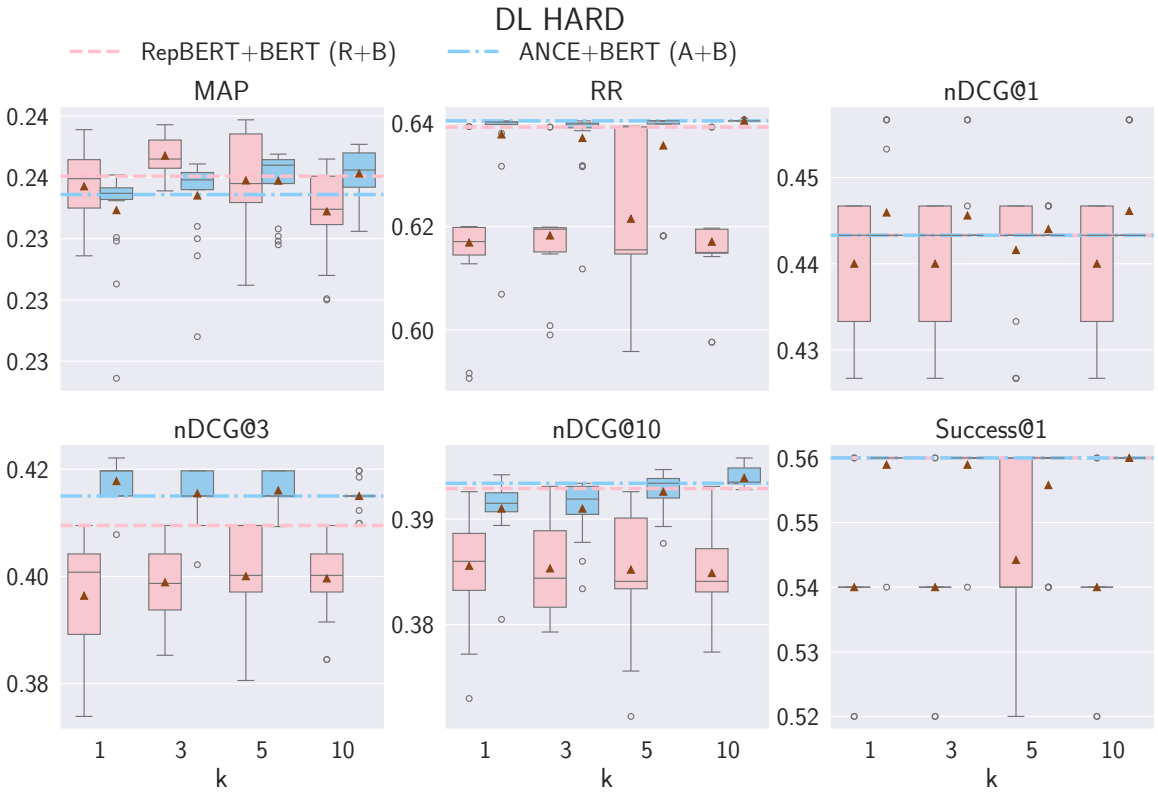


Figure 5.11: Impact of PRF depth on the effectiveness (y-axis) of ANCE+PRF+BERT(A+PRF+B) and RepBERT+PRF+BERT(R+PRF+B) for the task of reranking on DL HARD, k represents the different PRF depths. Baseline ANCE+BERT(A+B) and RepBERT+BERT(R+B) are marked with a dash-dot blue line and a dashed red line respectively.

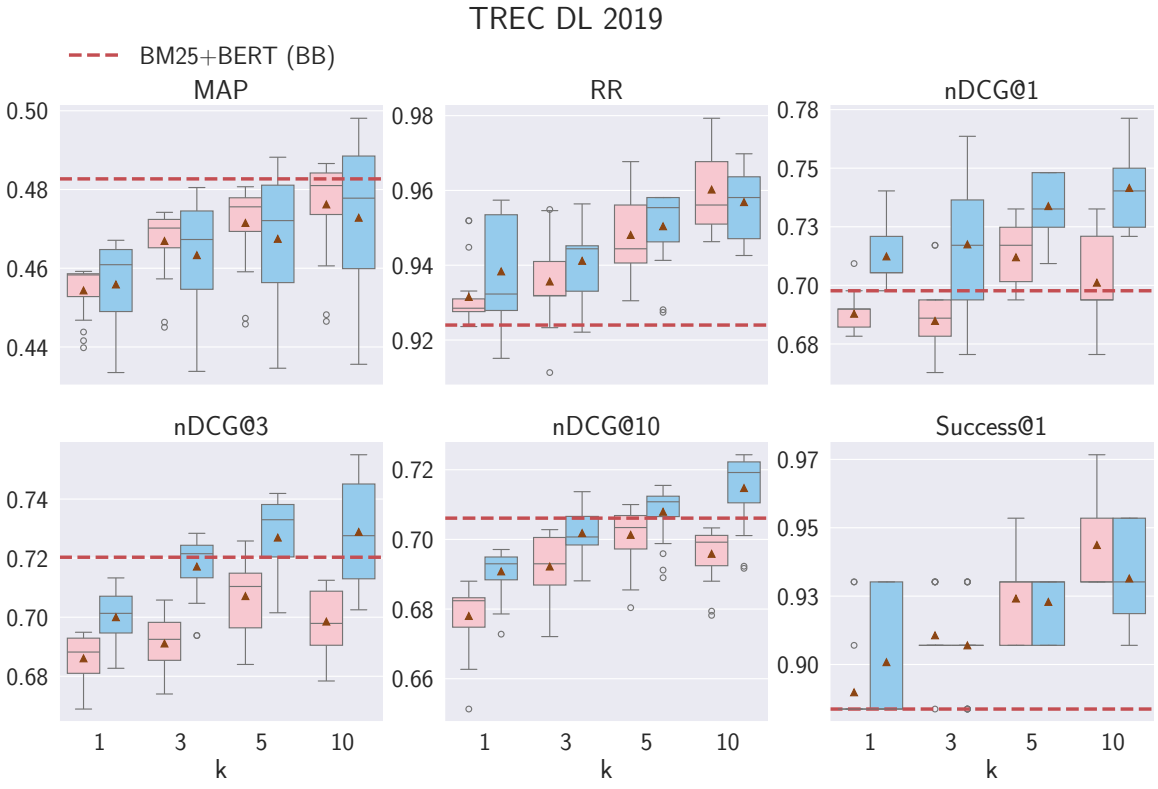


Figure 5.12: Impact of PRF depth on the effectiveness (y-axis) of BM25+BERT+PRF-RepBERT (BB+PRF-R) and BM25+BERT+PRF-ANCE (BB+PRF-A) for the task of reranking on TREC DL 2019, k represents the different PRF depths. Baseline BM25+BERT(BB) is marked with a dashed red line.

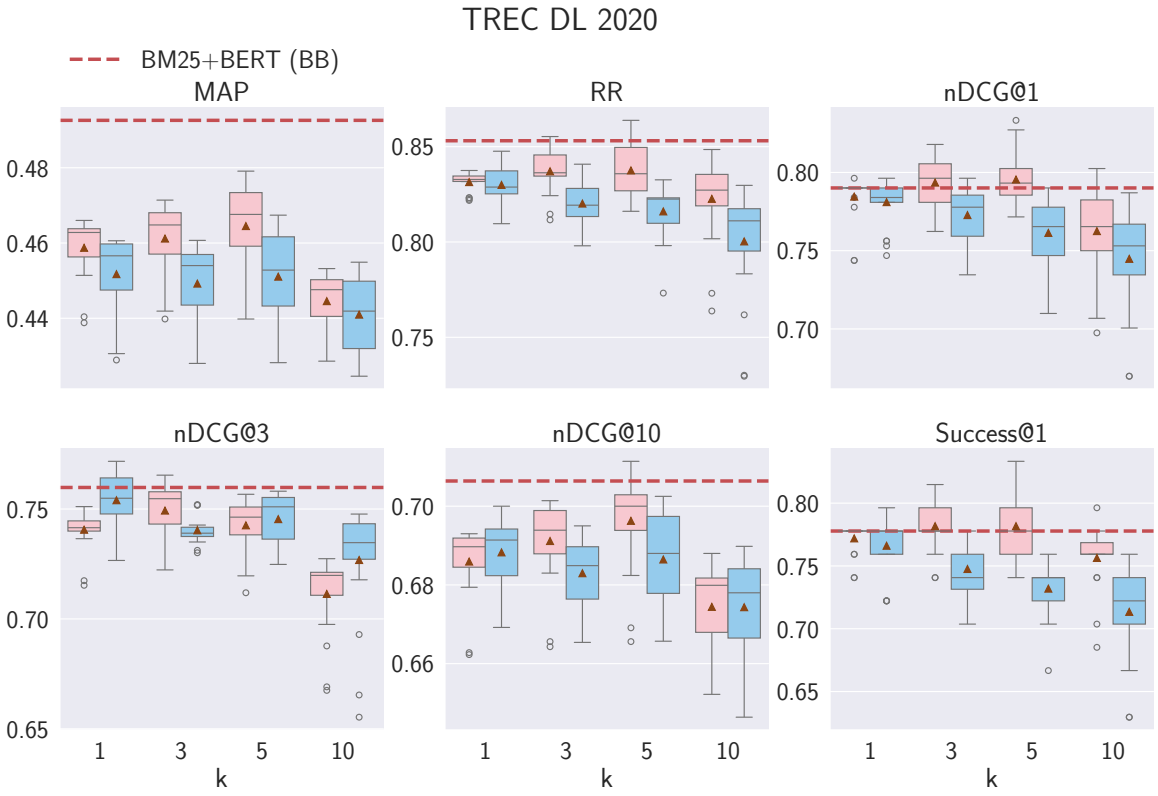


Figure 5.13: Impact of PRF depth on the effectiveness (y-axis) of BM25+BERT+PRF-RepBERT (BB+PRF-R) and BM25+BERT+PRF-ANCE (BB+PRF-A) for the task of reranking on TREC DL 2020, *k* represents the different PRF depths. Baseline BM25+BERT(BB) is marked with a dashed red line.

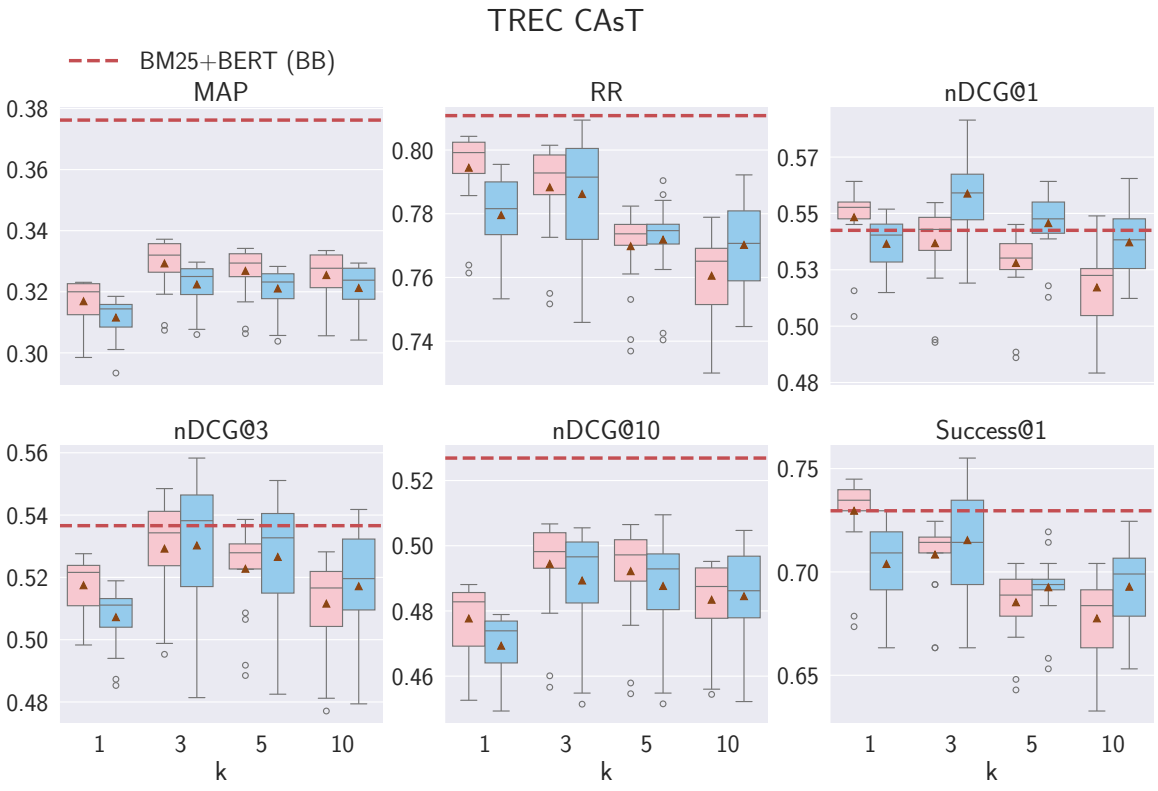


Figure 5.14: Impact of PRF depth on the effectiveness (y-axis) of BM25+BERT+PRF-RepBERT (BB+PRF-R) and BM25+BERT+PRF-ANCE (BB+PRF-A) for the task of reranking on TREC CAsT, *k* represents the different PRF depths. Baseline BM25+BERT(BB) is marked with a dashed red line.

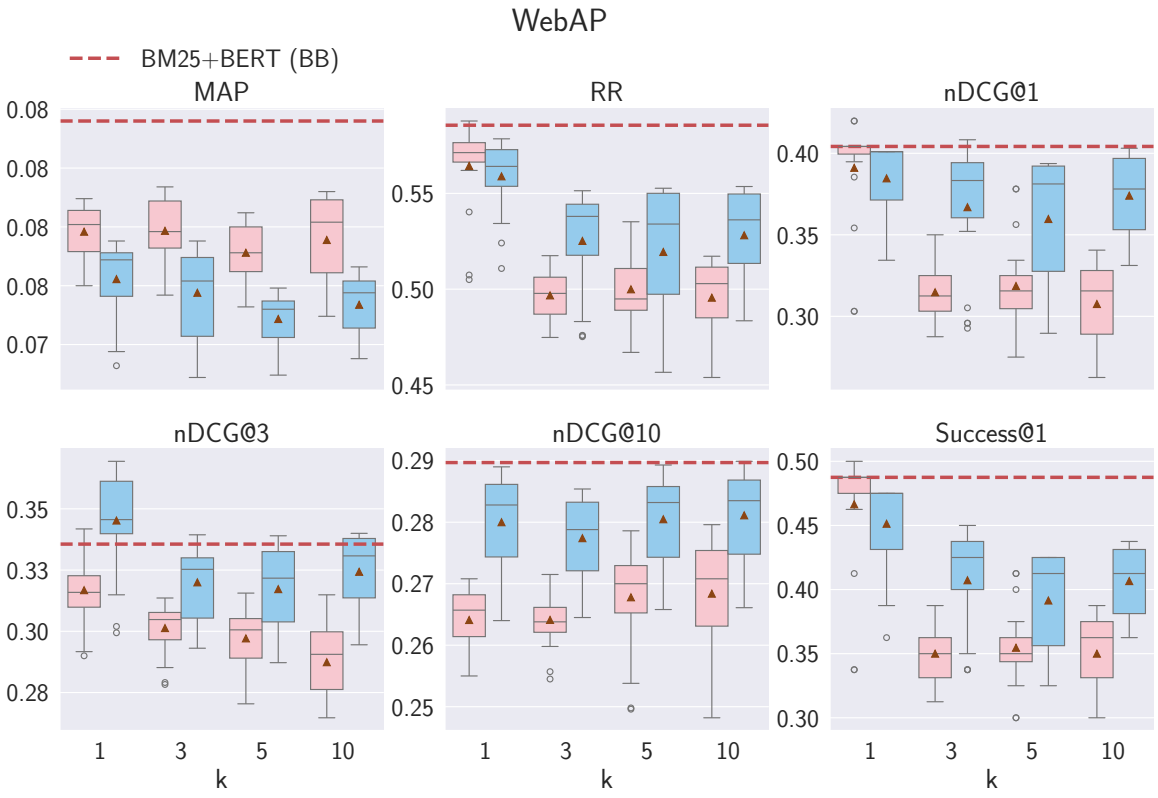


Figure 5.15: Impact of PRF depth on the effectiveness (y-axis) of BM25+BERT+PRF-RepBERT (BB+PRF-R) and BM25+BERT+PRF-ANCE (BB+PRF-A) for the task of reranking on WebAP, k represents the different PRF depths. Baseline BM25+BERT(BB) is marked with a dashed red line.

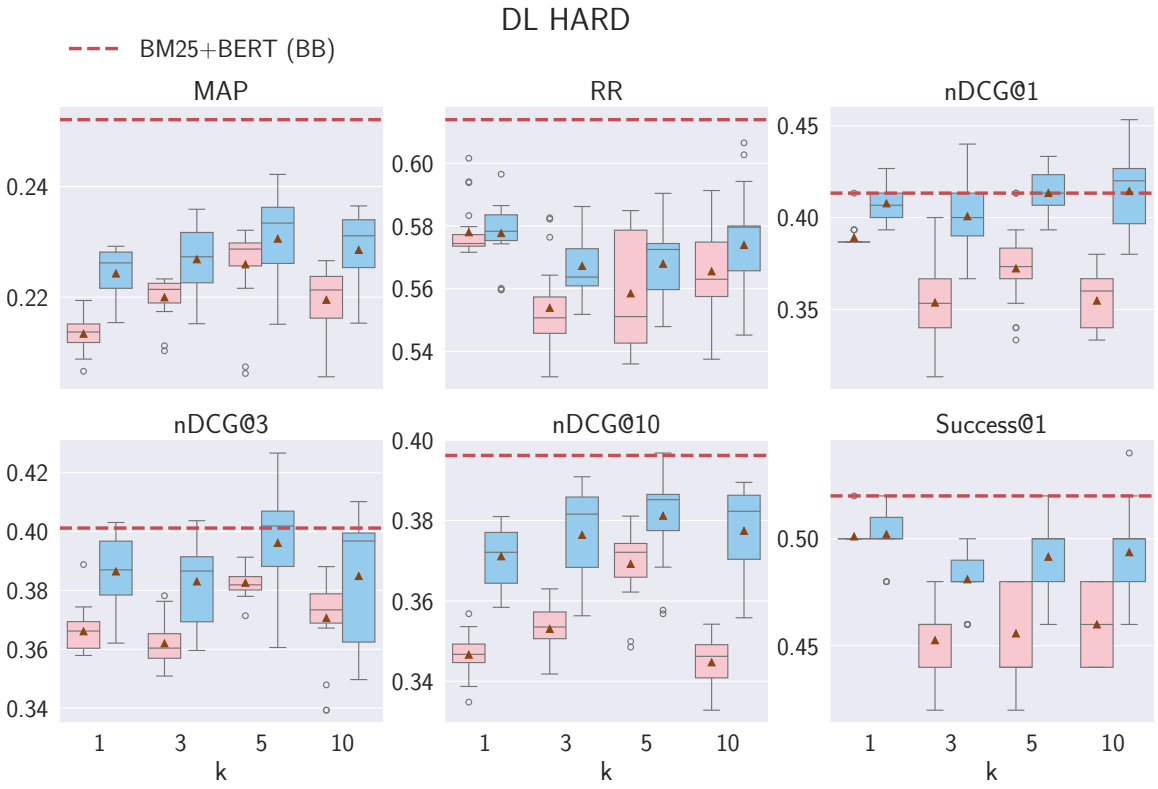


Figure 5.16: Impact of PRF depth on the effectiveness (y-axis) of BM25+BERT+PRF-RepBERT (BB+PRF-R) and BM25+BERT+PRF-ANCE (BB+PRF-A) for the task of reranking on DL HARD, k represents the different PRF depths. Baseline BM25+BERT(BB) is marked with a dashed red line.

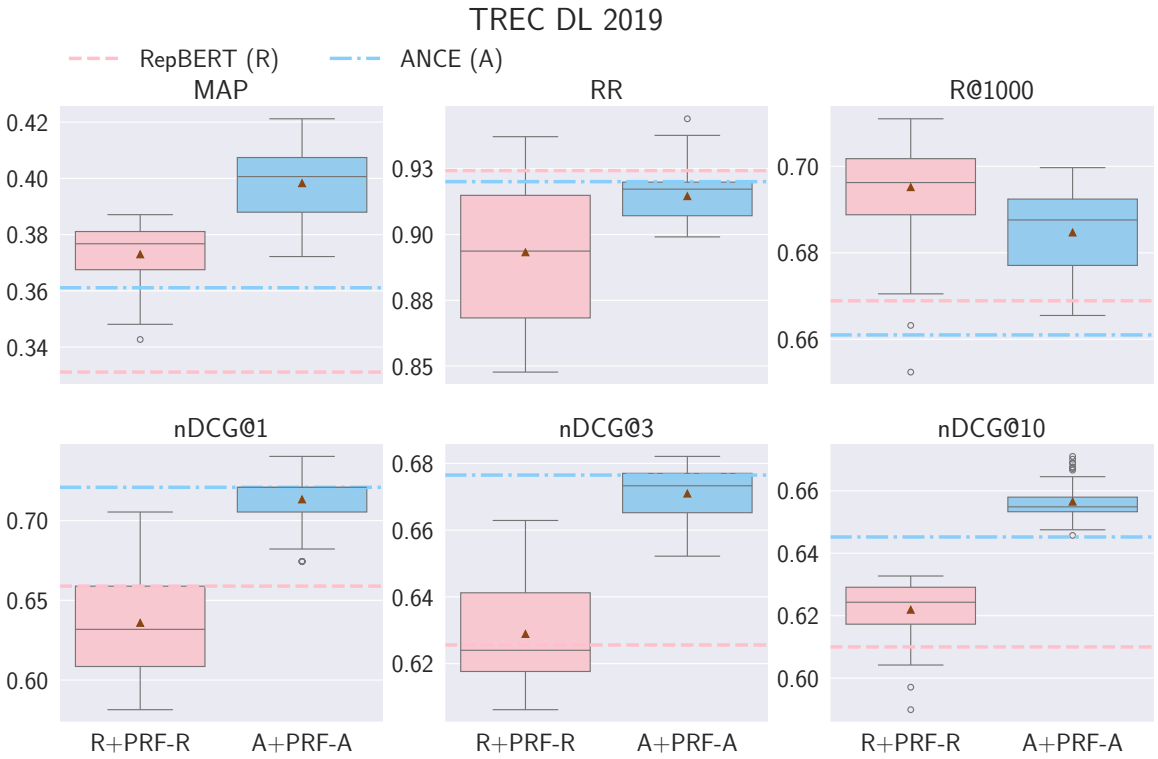


Figure 5.17: Vector-based PRF retrieval effectiveness (y-axis) by using different dense retrieval models RepBERT(R+PRF-R) and ANCE(A+PRF-A) on TREC DL 2019. Baseline RepBERT(R) is marked with dashed red line, ANCE(A) is marked with dash-dot blue line.

or dataset-dependent. These findings reinforce the robustness and generalizability of Vector-based PRF as an effective enhancement.

For comparison, Text-based PRF, as discussed in Chapter 3, exhibits more volatile behavior. While it can occasionally improve re-ranking performance, its effectiveness tends to decline as PRF depth increases, likely due to query drift and the inclusion of noisy context (see Section 12.1 in Chapter 12 for detailed case study). Unlike Vector-based PRF, which operates in a stable embedding space, Text-based approaches directly manipulate query text, making them more sensitive to feedback quality and less efficient in practice.

5.4.2 Impact of Dense Representation on the Effectiveness of Vector-Based PRF with Dense Retrievers

RQ1.3.2 investigates the impact of underlying text representations on the effectiveness of Vector-based PRF. To answer this, we evaluate the performance of PRF using different base encoders-specifically, ANCE and RepBERT-across multiple retrieval and reranking metrics. For reference, we also include results (Figures 3.7 – 3.11) from various text handling methods used in the Text-based PRF approaches discussed in Chapter 3.

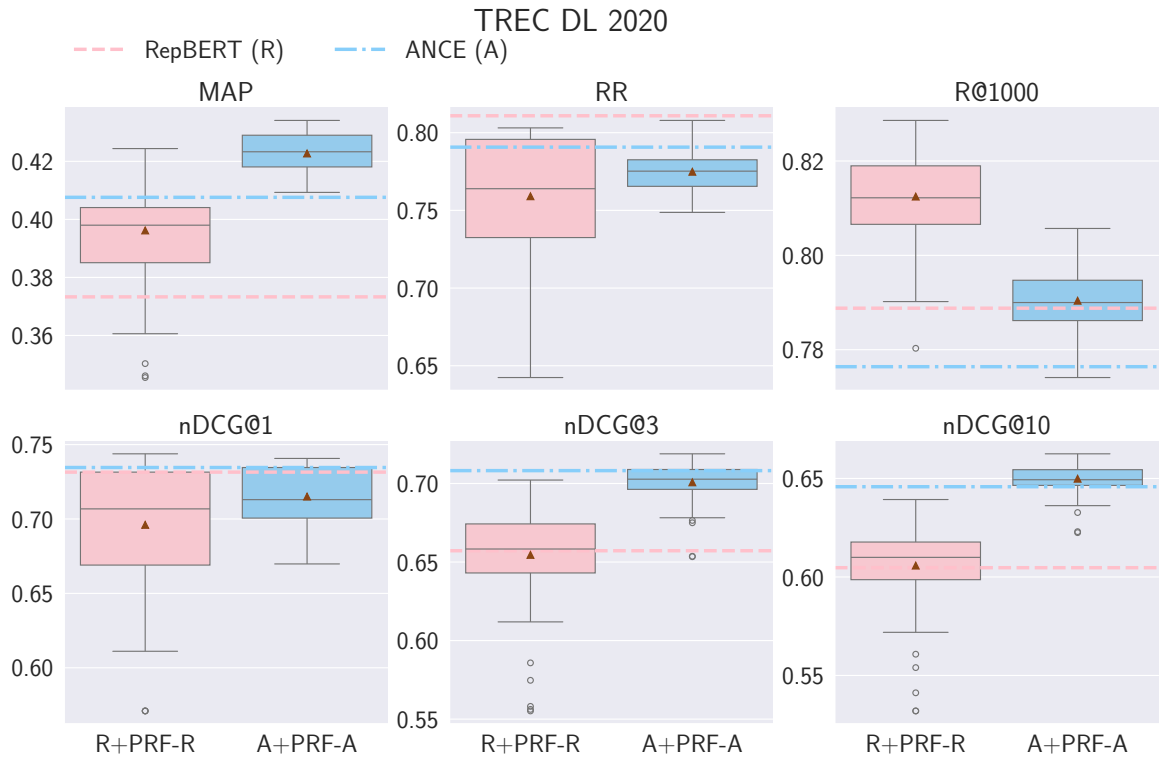


Figure 5.18: Vector-based PRF retrieval effectiveness (y-axis) by using different dense retrieval models RepBERT(R+PRF-R) and ANCE(A+PRF-A) on TREC DL 2020. Baseline RepBERT(R) is marked with dashed red line, ANCE(A) is marked with dash-dot blue line.

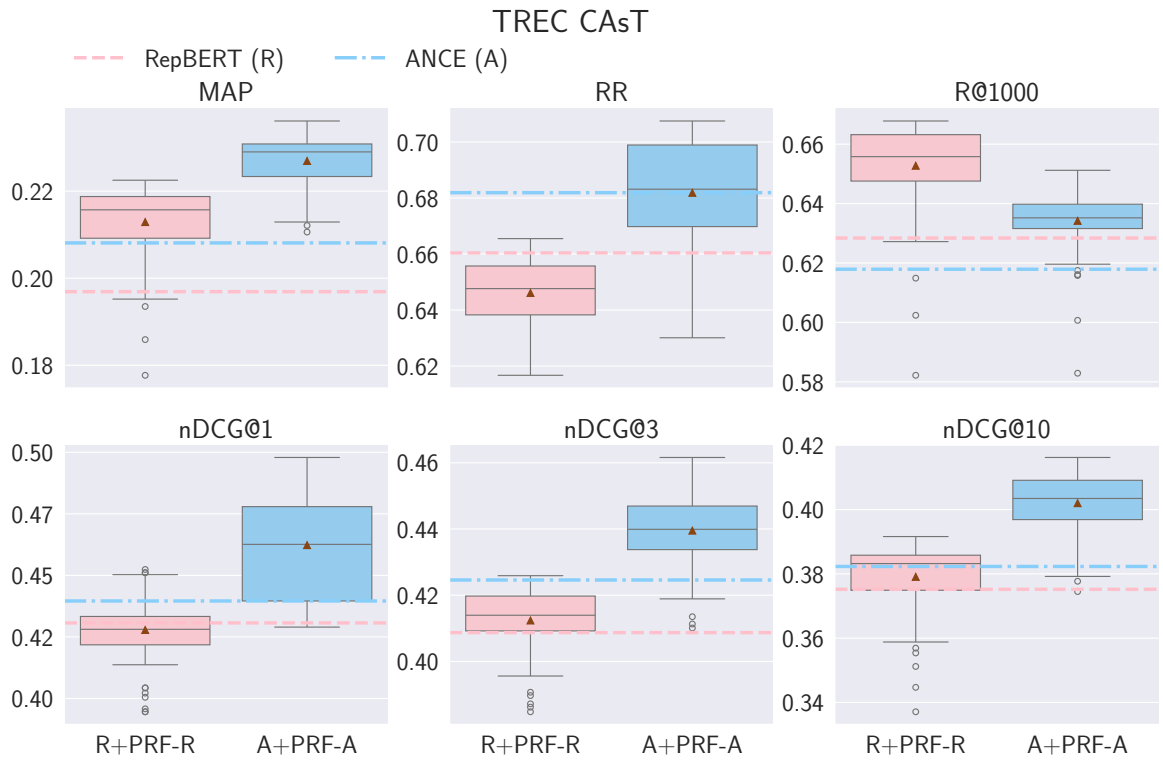


Figure 5.19: Vector-based PRF retrieval effectiveness (y-axis) by using different dense retrieval models RepBERT(R+PRF-R) and ANCE(A+PRF-A) on TREC CAsT. Baseline RepBERT(R) is marked with dashed red line, ANCE(A) is marked with dash-dot blue line.

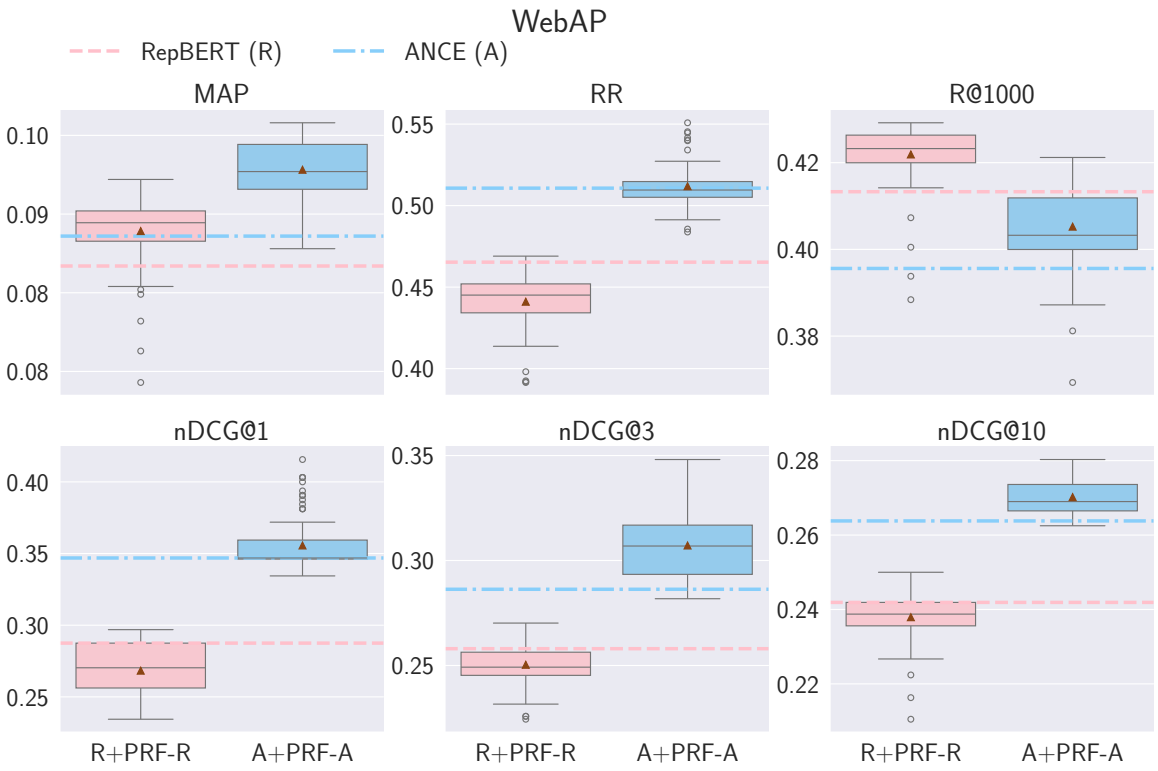


Figure 5.20: Vector-based PRF retrieval effectiveness (y-axis) by using different dense retrieval models RepBERT(R+PRF-R) and ANCE(A+PRF-A) on TREC WebAP. Baseline RepBERT(R) is marked with dashed red line, ANCE(A) is marked with dash-dot blue line.

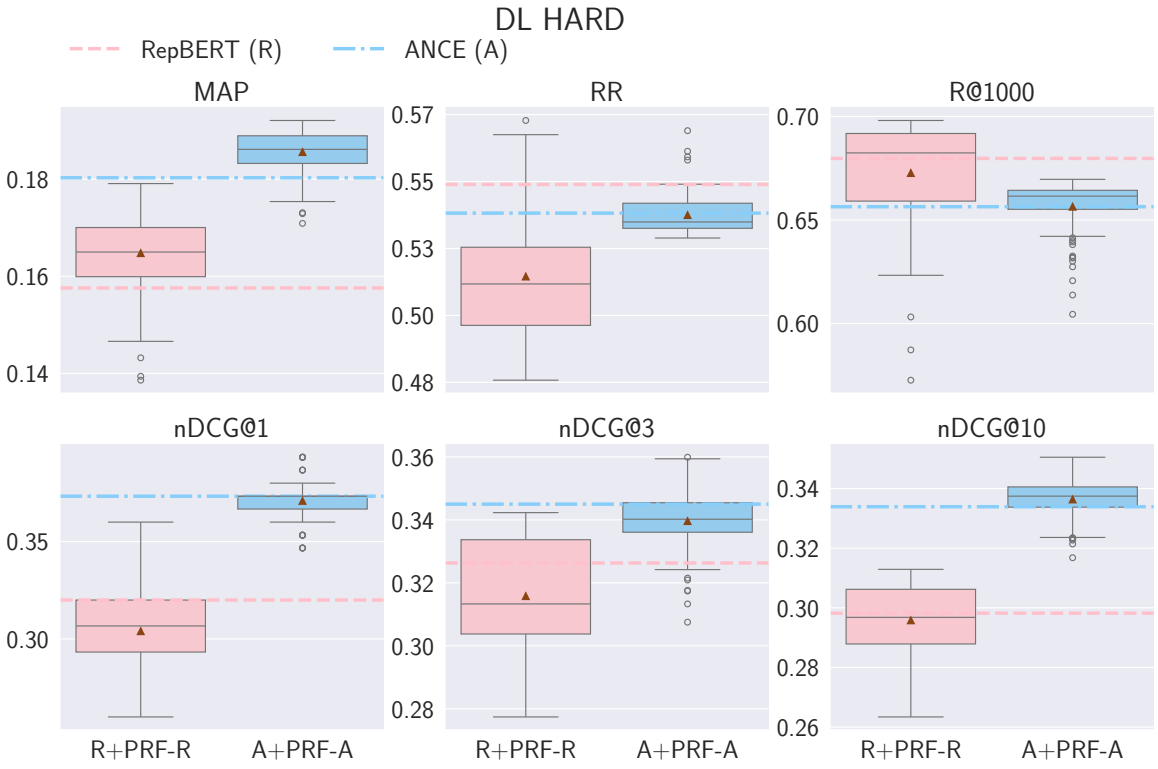


Figure 5.21: Vector-based PRF retrieval effectiveness (y-axis) by using different dense retrieval models RepBERT(R+PRF-R) and ANCE(A+PRF-A) on DL HARD. Baseline RepBERT(R) is marked with dashed red line, ANCE(A) is marked with dash-dot blue line.

Retrieval With Different Representations For Vector-Based PRF

Results addressing **RQ1.3.2**, the impact of dense representations on Vector-based PRF effectiveness, are presented in Figure 5.17 – 5.21. Across datasets, both A+PRF-A and R+PRF-R consistently enhance deep retrieval metrics such as MAP, Recall@1000, and nDCG@10 relative to their respective dense retriever baselines. For example, on TREC DL 2019 (Figure 5.17) and 2020 (Figure 5.18), both models offer substantial improvements over MAP and R@1000, while nDCG@10 improves marginally for A+PRF-A and remains comparable for R+PRF-R. These results suggest that both ANCE and RepBERT representations benefit from Vector-based PRF, particularly in enhancing MAP and recall. However, ANCE-based PRF shows more robust and consistent improvements across metrics, indicating that the underlying encoder plays a key role in the quality of the feedback-enhanced representation. The gap between A+PRF-A and R+PRF-R is more noticeable in early precision metrics, where A+PRF-A generally performs better, likely due to ANCE's stronger baseline performance and better feedback signal quality. This difference becomes particularly significant in more challenging datasets like DL HARD (Figure 5.21), where improvements from both methods are limited, but A+PRF-A still shows relatively stronger performance. Overall, these findings highlight that while Vector-based PRF is broadly effective, its impact is amplified when paired with stronger dense retrievers such as ANCE.

In contrast, gains on shallow metrics are less observable. Neither method shows consistent improvements in Reciprocal Rank (RR) or nDCG@1 across datasets, suggesting that Vector-based PRF has limited impact on early precision. On TREC CAsT 2019 (Figure 5.19), A+PRF-A achieves substantial gains in nDCG@1, while R+PRF-R also performs competitively, indicating that when the feedback signal is strong and well-aligned with the query intent, as may be the case in CAsT's conversational setting, Vector-based PRF can improve top-ranking results. Both models show strong improvements in nDCG@3 and nDCG@10, particularly on CAsT (Figure 5.19) and WebAP (Figure 5.20), highlighting their effectiveness at improving mid- to deep-rank precision. However, R+PRF-R degrades nDCG@1,3 on WebAP, suggesting that its feedback signal may be noisier or less reliable in datasets with longer passages, where RepBERT underperforms. For DL HARD (Figure 5.21), improvements are observed for both models showing notable gains in MAP, R@1000, and nDCG@10, indicating some effectiveness in not only retrieving a broader range of relevant documents but also moving relevant documents to higher ranks. However, most other metrics on DL HARD remain unaffected or exhibit only marginal variations, reflecting the dataset's difficulty and the limited utility of noisy feedback passages. These patterns highlight that while Vector-based PRF generally boosts deep metrics, its ability to enhance early precision is conditional and model-dependent.

These results highlight the effectiveness of Vector-based PRF in improving retrieval performance, particularly when using strong dense representations. The improvements are most reliable on metrics like MAP and R@1000, while early precision metrics remain challenging to improve consistently.

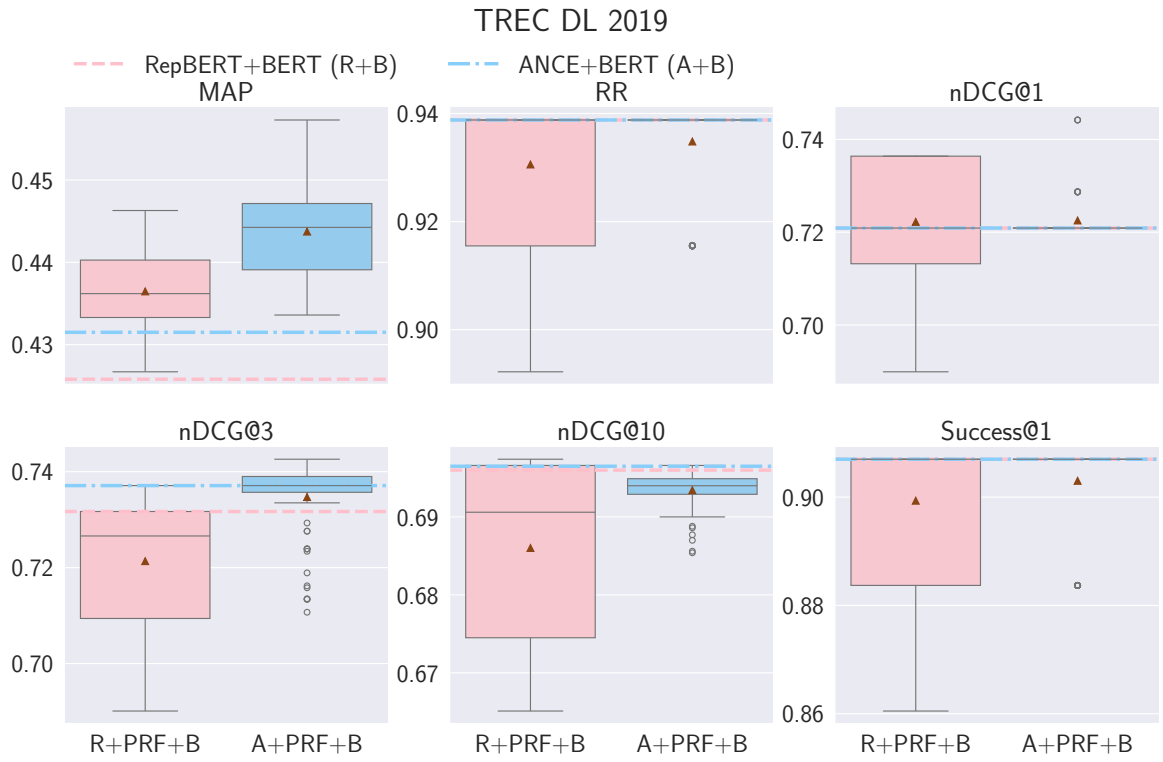


Figure 5.22: Impact of dense representations on the effectiveness (y-axis) of PRF approaches for the task of reranking on TREC DL 2019, where R+PRF+B and A+PRF+B represents RepBERT+PRF+BERT and ANCE+PRF+BERT, respectively. Baseline ANCE+BERT(A+B) and RepBERT+BERT(R+B) are marked with a dash-dot blue line and a dashed red line respectively.

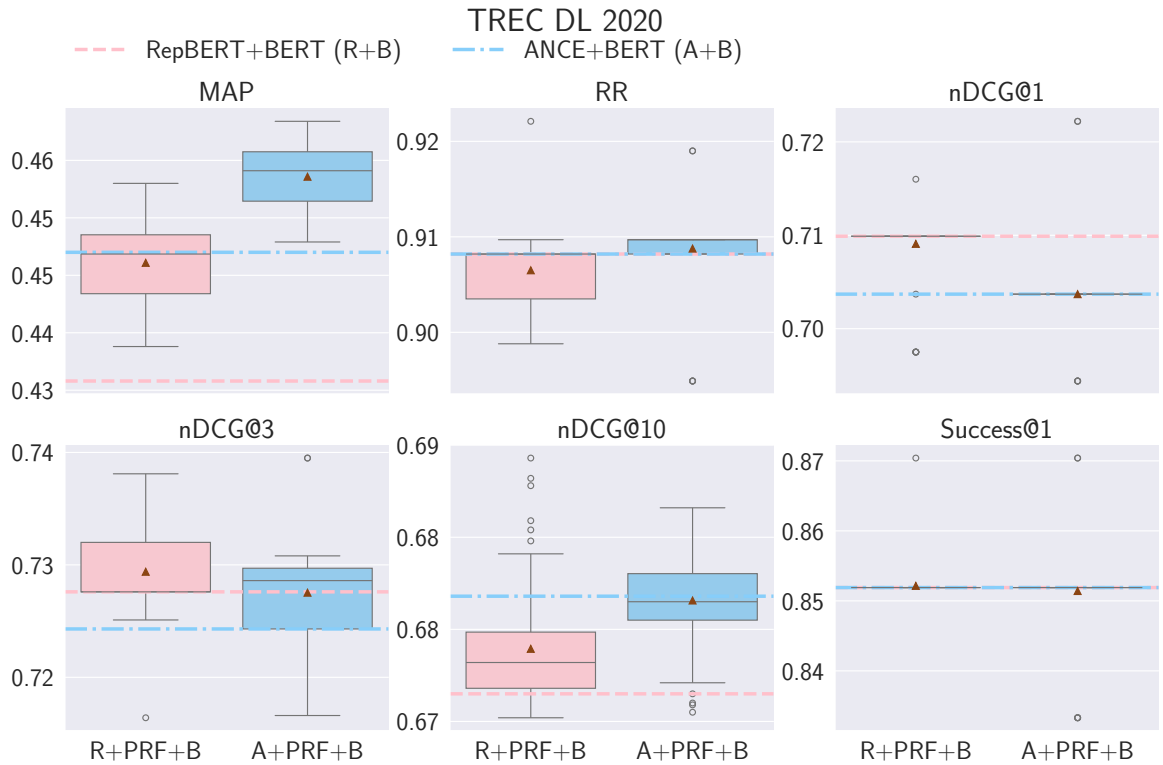


Figure 5.23: Impact of dense representations on the effectiveness (y-axis) of PRF approaches for the task of reranking on TREC DL 2020, where R+PRF+B and A+PRF+B represents RepBERT+PRF+BERT and ANCE+PRF+BERT, respectively. Baseline ANCE+BERT(A+B) and RepBERT+BERT(R+B) are marked with a dash-dot blue line and a dashed red line respectively.

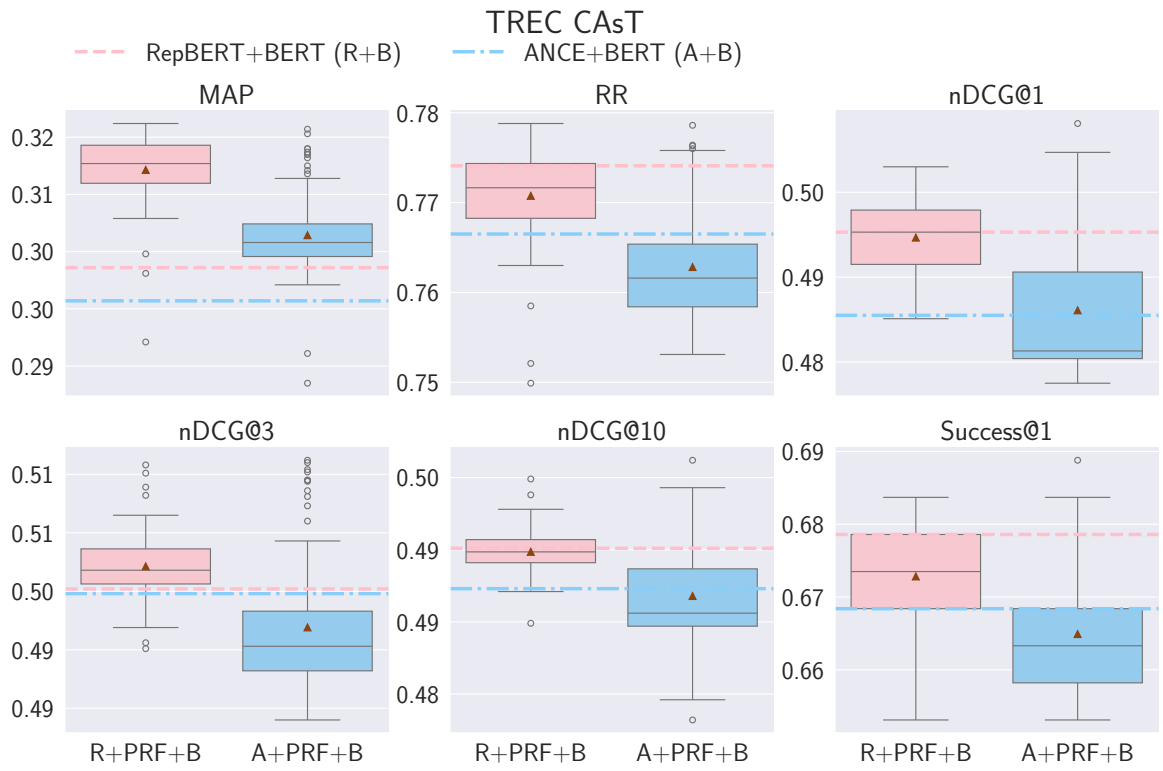


Figure 5.24: Impact of dense representations on the effectiveness (y-axis) of PRF approaches for the task of reranking on TREC CAsT, where R+PRF+B and A+PRF+B represents RepBERT+PRF+BERT and ANCE+PRF+BERT, respectively. Baseline ANCE+BERT(A+B) and RepBERT+BERT(R+B) are marked with a dash-dot blue line and a dashed red line respectively.

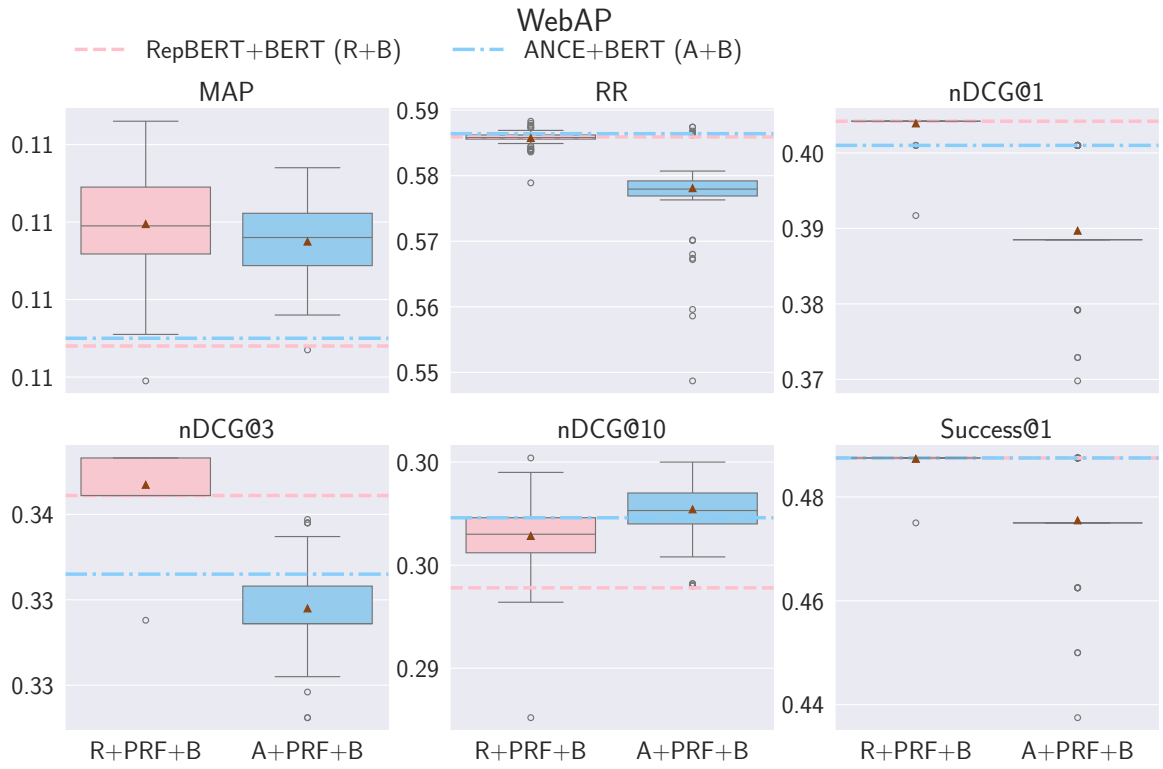


Figure 5.25: Impact of dense representations on the effectiveness (y-axis) of PRF approaches for the task of reranking on WebAP, where R+PRF+B and A+PRF+B represents RepBERT+PRF+BERT and ANCE+PRF+BERT, respectively. Baseline ANCE+BERT(A+B) and RepBERT+BERT(R+B) are marked with a dash-dot blue line and a dashed red line respectively.

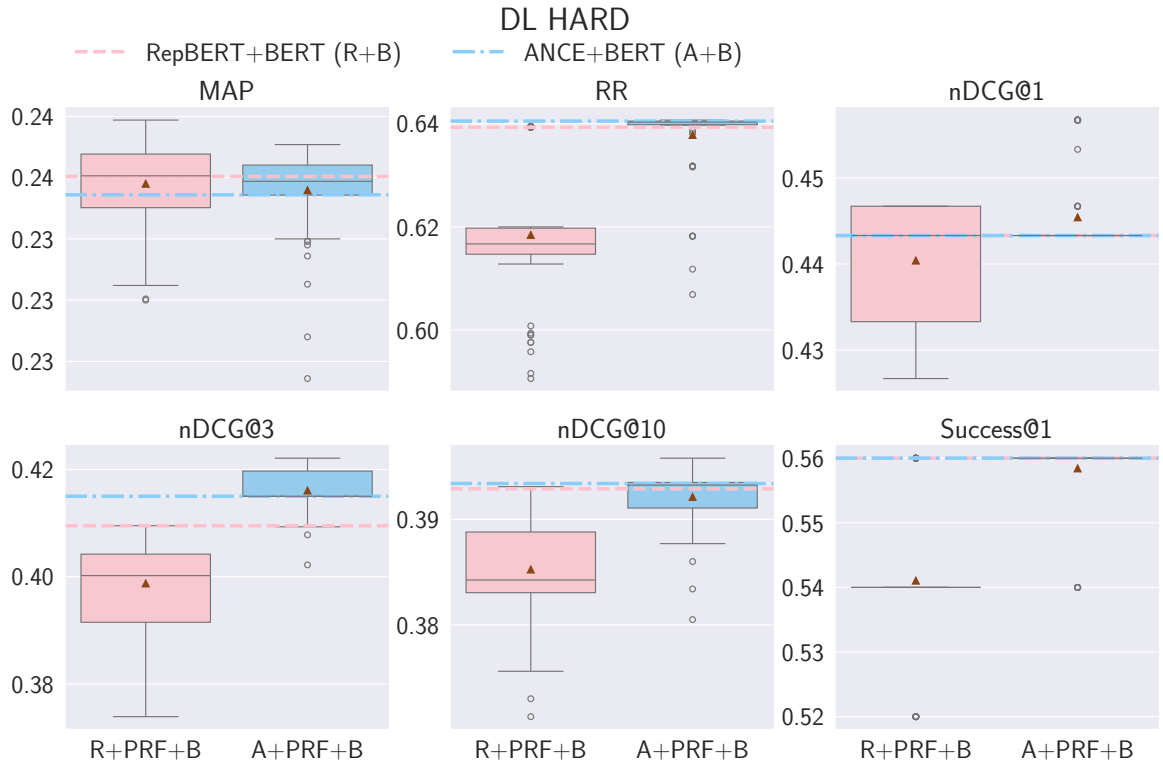


Figure 5.26: Impact of dense representations on the effectiveness (y-axis) of PRF approaches for the task of reranking on DL HARD, where R+PRF+B and A+PRF+B represents RepBERT+PRF+BERT and ANCE+PRF+BERT, respectively. Baseline ANCE+BERT(A+B) and RepBERT+BERT(R+B) are marked with a dash-dot blue line and a dashed red line respectively.

Reranking With Different Representations For Vector-Based PRF and Comparison With Text-Based PRF

To further investigate **RQ1.3.2**, we evaluate the impact of dense representations on the effectiveness of Vector-based PRF within *hybrid* reranking frameworks. Results are presented in Figures 5.22 – 5.26. We focus on two dense retrievers, RepBERT and ANCE, combined with BERT reranking. Across datasets, Vector-based PRF improves MAP consistently, though the magnitude of improvement varies depending on the dataset and underlying representation. For instance, on TREC DL 2019 (Figure 5.22) and 2020 (Figure 5.23), ANCE-based models outperform RepBERT-based ones in MAP, RR, and nDCG@10, confirming ANCE's stronger baseline and better alignment with the BERT reranker in these settings. The consistent superiority of ANCE-based variants suggests that stronger base encoders provide more reliable feedback signals, which in turn benefit reranking. On CAsT 2019 (Figure 5.24) and WebAP (Figure 5.25), however, RepBERT-based models offer higher gains on most metrics, particularly MAP, indicating that RepBERT is more effective in capturing conversational or longer-form passage characteristics. This reversal in relative performance highlights that the advantage of a given representation is context-dependent. On DL HARD (Figure 5.26), both variants produce only marginal MAP improvements, with ANCE-based ones also slightly improve nDCG@3, but fail to improve other metrics (either on par or severe degradation), reflecting the challenging nature of the dataset and suggesting that noisy or ambiguous feedback signals cannot be easily mitigated by either representation. Overall, these findings emphasize that the choice of dense representation in

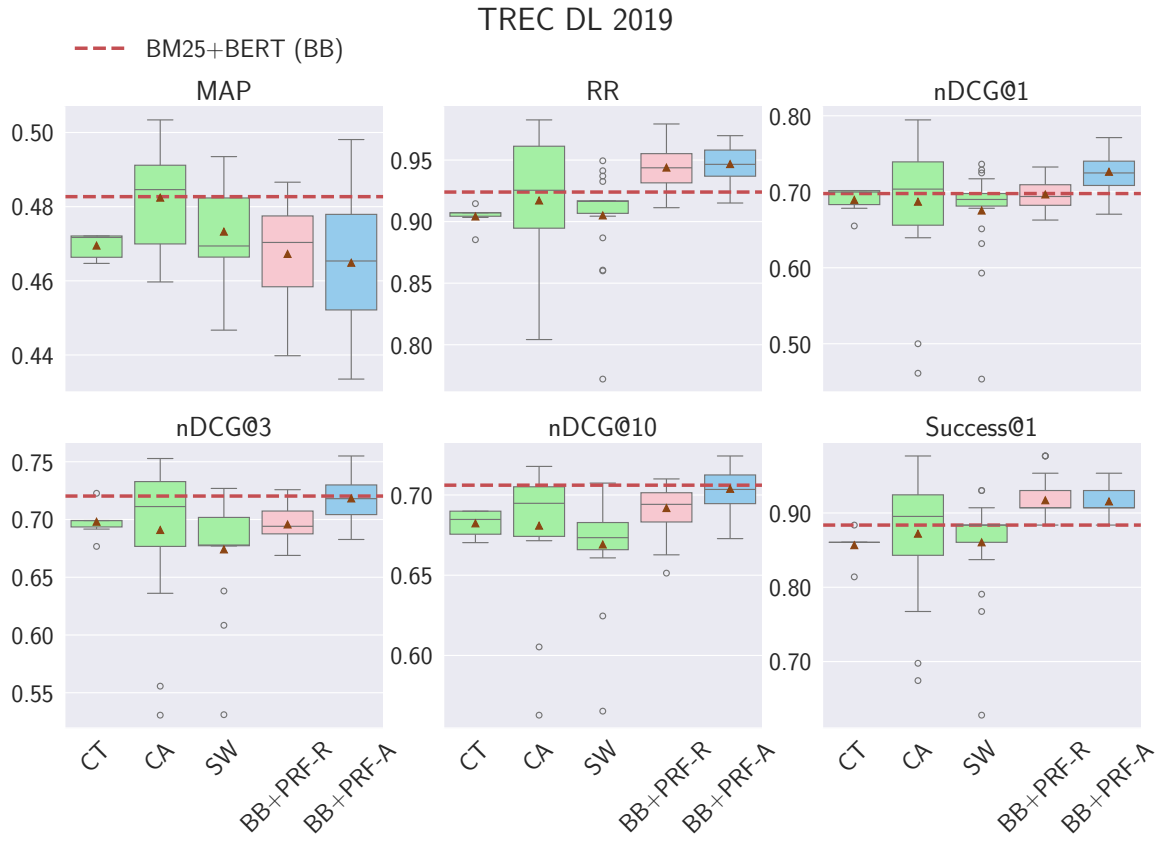


Figure 5.27: Impact of representations on the effectiveness (y-axis) of PRF approaches for the task of reranking on TREC DL 2019, where CT, CA and SW represent the text handling methods (from Text-based PRF) Concatenate and Truncate, Concatenate and Aggregate and Sliding Window, respectively, while BM25+BERT+PRF-RepBERT(BB+PRF-R) and BM25+BERT+PRF-ANCE(BB+PRF-A) are the dense representations for text. Baseline BM25+BERT(BB) is marked with a dashed red line.

Vector-based PRF affects not only retrieval effectiveness but also the downstream benefits of hybrid reranking.

For early precision metrics (e.g., RR, nDCG@1), both ANCE- and RepBERT-based hybrid models generally offer no improvement, and in many cases, performance is degraded. This suggests that Vector-based PRF primarily enhances depth-oriented retrieval quality, while offering limited benefit for top-ranked results. One possible explanation is that the feedback signal used in PRF, derived from top-ranked passages, may not consistently contain highly relevant information necessary to improve ranking at the very top positions (see Section 12.1 in Chapter 12 for detailed case studies). As a result, PRF-adjusted queries might slightly shift focus away from the most relevant item, leading to performance drops in early precision. This is especially observable in datasets like DL HARD and WebAP, where relevant passages are sparse or harder to detect. Even when deeper metrics such as MAP and Recall@1000 benefit from the expanded semantic representation offered by PRF, the refinement needed to boost shallow metrics appears more sensitive to the quality of initial top-ranked results, highlighting a potential limitation of applying Vector-based PRF without explicit mechanisms to guard against query drift or misalignment in early ranks.

For contextual comparison, results from text handling methods with Text-based PRF models (e.g., SW, CT, CA) are included in Figures 5.27 – 5.31. While some methods such as CA show

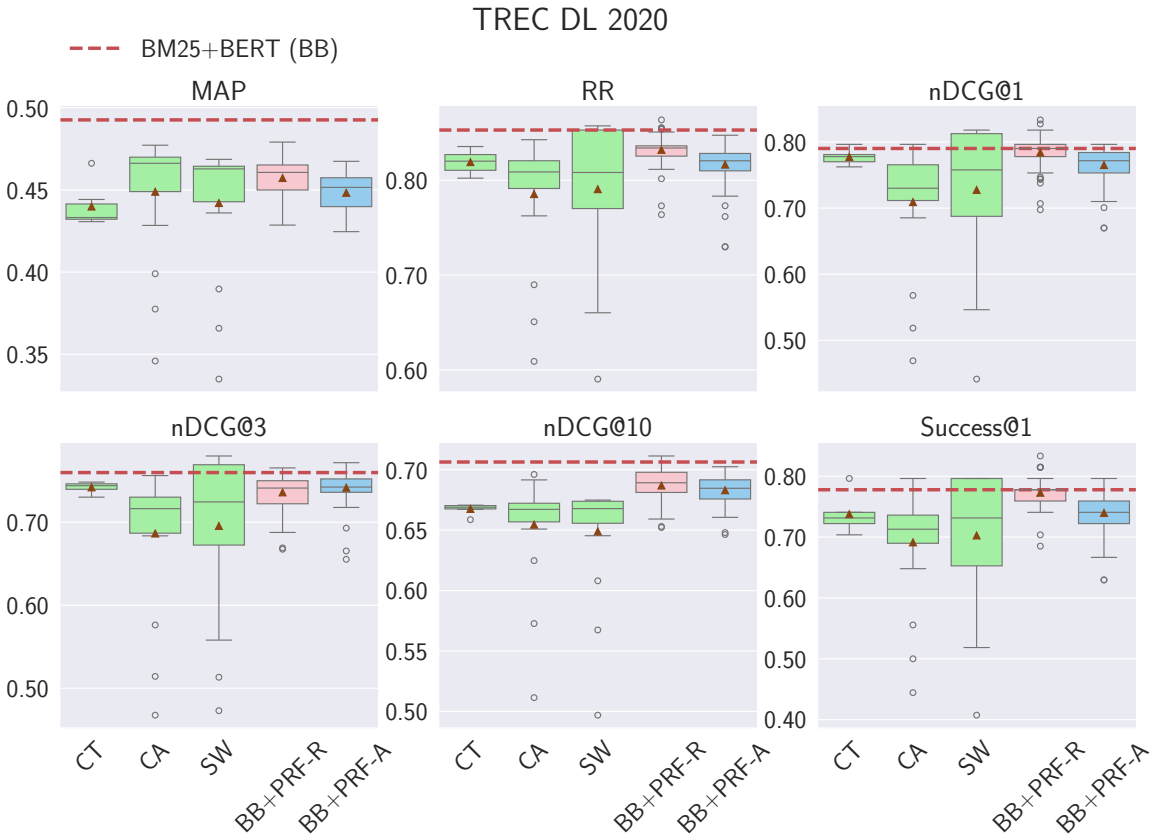


Figure 5.28: Impact of representations on the effectiveness (y-axis) of PRF approaches for the task of reranking on TREC DL 2020, where CT, CA and SW represent the text handling methods (from Text-based PRF) Concatenate and Truncate, Concatenate and Aggregate and Sliding Window, respectively, while BM25+BERT+PRF-RepBERT(BB+PRF-R) and BM25+BERT+PRF-ANCE(BB+PRF-A) are the dense representations for text. Baseline BM25+BERT(BB) is marked with a dashed red line.

strong performance on TREC DL 2019 (Figure 5.27), observable in MAP, RR, and nDCG@1, these gains are not consistent on other datasets. For example, performance on TREC DL 2020 and DL HARD is either flat or degraded across all text handling methods, and metrics such as MAP and nDCG@10 show limited sensitivity to these Text-based techniques. This inconsistency suggests that Text-based PRF approaches are more susceptible to query drift and the limitations imposed by sequence length constraints in BERT. In contrast, the Vector-based PRF models provide more consistent MAP improvements across both RepBERT- and ANCE-based dense retrievers and across datasets, demonstrating better robustness to data variation. These findings highlight the utility of dense representations in PRF, as they enable scalable and semantically rich query adjustments without incurring the efficiency and generalization drawbacks observed in Text-based PRF methods.

Summary For RQ1.3.2

To summarize, Vector-based PRF demonstrates strong and consistent effectiveness, particularly in retrieval. Among the two variants, A+PRF-A (using ANCE) consistently outperforms R+PRF-R (using RepBERT) across all metrics and datasets. This is largely due to ANCE's stronger baseline performance, especially in early precision, which results in higher-quality feedback passages and more

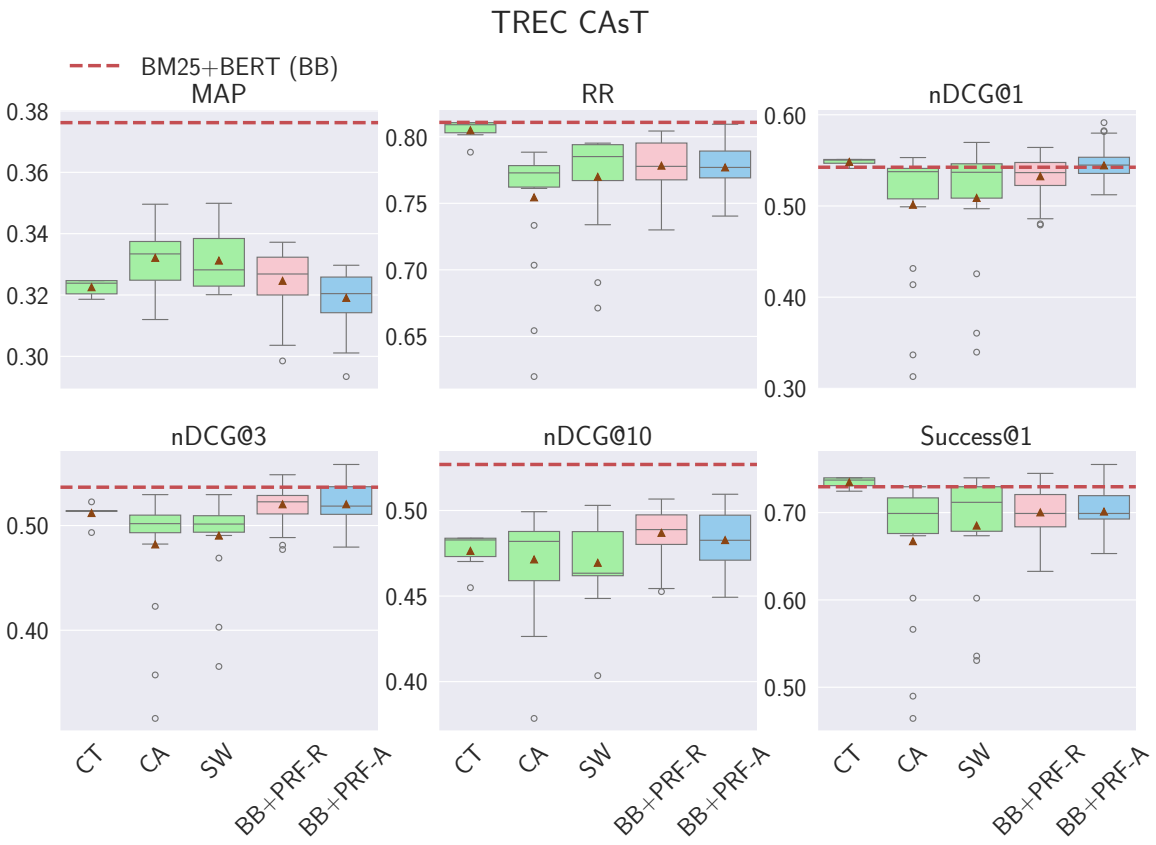


Figure 5.29: Impact of representations on the effectiveness (y-axis) of PRF approaches for the task of reranking on TREC CAsT, where CT, CA and SW represent the text handling methods (from Text-based PRF) Concatenate and Truncate, Concatenate and Aggregate and Sliding Window, respectively, while BM25+BERT+PRF-RepBERT(BB+PRF-R) and BM25+BERT+PRF-ANCE(BB+PRF-A) are the dense representations for text. Baseline BM25+BERT(BB) is marked with a dashed red line.

reliable PRF signals. In contrast, R+PRF-R inherits weaker initial rankings from RepBERT, leading to noisier feedback and reduced effectiveness.

When a BERT reranker is applied on top of Vector-based PRF (A+PRF+B, R+PRF+B), both ANCE-based and RepBERT-based models achieve consistent improvements in MAP across datasets. However, their impact on other metrics, including RR and nDCG@{1, 3, 10}, remains limited, with most changes being marginal and not statistically significant.

In contrast, Text-based PRF, which was discussed in Chapter 3, shows less consistent performance. While methods such as CA improve MAP more reliably, BB+PRF-R tends to boost RR, and BB+PRF-A shows better gains in nDCG@{1, 3, 10}. However, these improvements are generally modest and less stable across datasets. Moreover, all Text-based PRF methods are substantially more costly in terms of latency and more susceptible to query drift due to their dependence on explicit textual manipulation.

Overall, Vector-based PRF provides more stable, effective, and efficient gains, particularly in retrieval tasks, and demonstrates broader applicability than Text-based alternatives.

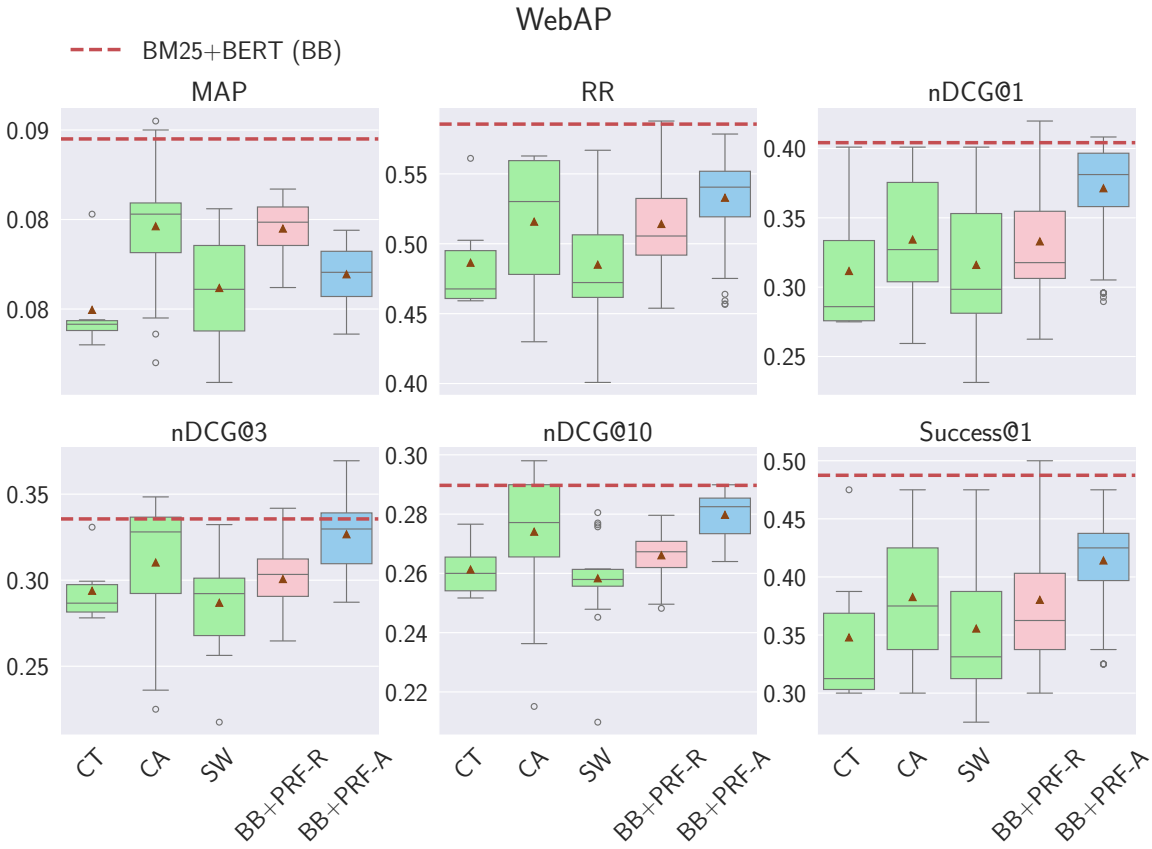


Figure 5.30: Impact of representations on the effectiveness (y-axis) of PRF approaches for the task of reranking on WebAP, where CT, CA and SW represent the text handling methods (from Text-based PRF) Concatenate and Truncate, Concatenate and Aggregate and Sliding Window, respectively, while BM25+BERT+PRF-RepBERT(BB+PRF-R) and BM25+BERT+PRF-ANCE(BB+PRF-A) are the dense representations for text. Baseline BM25+BERT(BB) is marked with a dashed red line.

5.4.3 Impact of Vector Fusion Method on the Effectiveness of Vector-Based PRF with Dense Retrievers

RQ1.3.3 explores how different vector fusion strategies affect the effectiveness of Vector-based PRF. To address this, we evaluate the performance of various fusion methods across multiple retrieval and reranking metrics. Where relevant, we also reference results from text aggregation techniques used in Text-based PRF (Figures 3.12 – 3.16, Chapter 3) to provide a comparative context.

Retrieval With Different Vector Fusions For Vector-Based PRF

To address research question **RQ1.3.3**, we vary the fusion methods used to combine query vector with feedback vectors and analyze their performance with fixed PRF depths and dense representations. Specifically, we evaluate three fusion approaches: simple averaging (V-A), Rocchio with a fixed query weight and varying feedback weight ($\mathcal{RC}_{\beta}$), and Rocchio with both query and feedback weights varied ($\mathcal{RC}_{\alpha,\beta}$). For contextual comparison, we additionally include the results from Text-based PRF methods, although the primary focus remains on evaluating the relative effectiveness of the Vector-based fusion strategies.

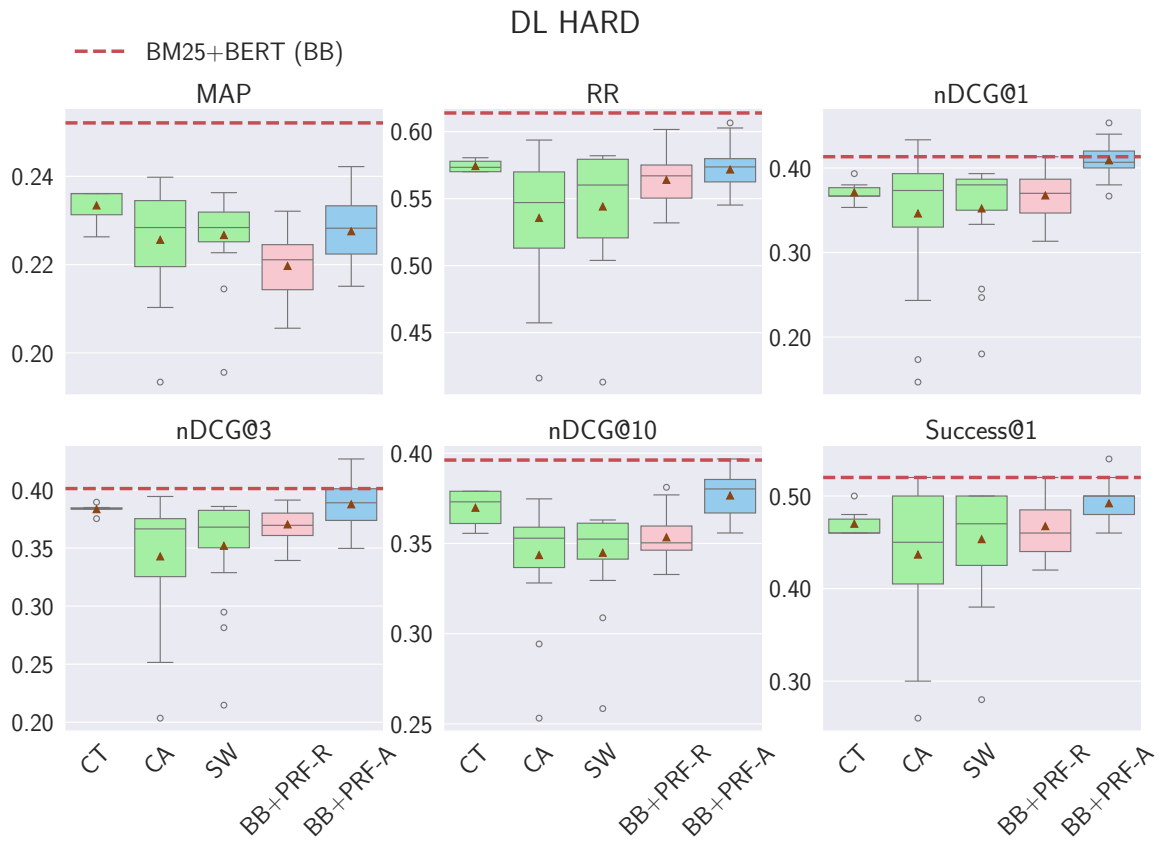


Figure 5.31: Impact of representations on the effectiveness (y-axis) of PRF approaches for the task of reranking on TREC DL 2019, where CT, CA and SW represent the text handling methods (from Text-based PRF) Concatenate and Truncate, Concatenate and Aggregate and Sliding Window, respectively, while BM25+BERT+PRF-RepBERT(BB+PRF-R) and BM25+BERT+PRF-ANCE(BB+PRF-A) are the dense representations for text. Baseline BM25+BERT(BB) is marked with a dashed red line.

For retrieval with vector fusion methods, results are presented in Figures 5.32 – 5.36. On TREC DL 2019 (Figure 5.32), A+PRF-A consistently outperforms its baseline in MAP, Recall@1000, and nDCG@10 across all fusion strategies, but underperforms in RR and early precision metrics (nDCG@{1, 3}). R+PRF-R shows similar improvements in deep metrics and achieves performance on par with the baseline for nDCG@3 when using $\mathcal{RC}_{\alpha,\beta}$.

For TREC DL 2020 (Figure 5.33), R+PRF-R achieves notable gains in Recall@1000, particularly under $\mathcal{RC}_{\beta}$, although neither R+PRF-R nor A+PRF-A improve over the baselines in RR. A+PRF-A also improves Recall@1000, although to a lesser extent than R+PRF-R. Both models perform comparably to their baselines in nDCG@10, with R+PRF-R showing marginal improvements when $\mathcal{RC}_{\beta}$ is applied.

On TREC CAsT 2019 (Figure 5.34), both A+PRF-A and R+PRF-R yield substantial gains in MAP. A+PRF-A also demonstrates consistent improvements in nDCG@{1, 3, 10}. However, for other metrics, both models perform similarly to or worse than their respective baselines, regardless of the vector fusion method used.

In WebAP (Figure 5.35), both A+PRF-A and R+PRF-R generally enhance MAP and Recall@1000. The exception is R+PRF-R with simple averaging (V-A), which performs on par with the baseline in

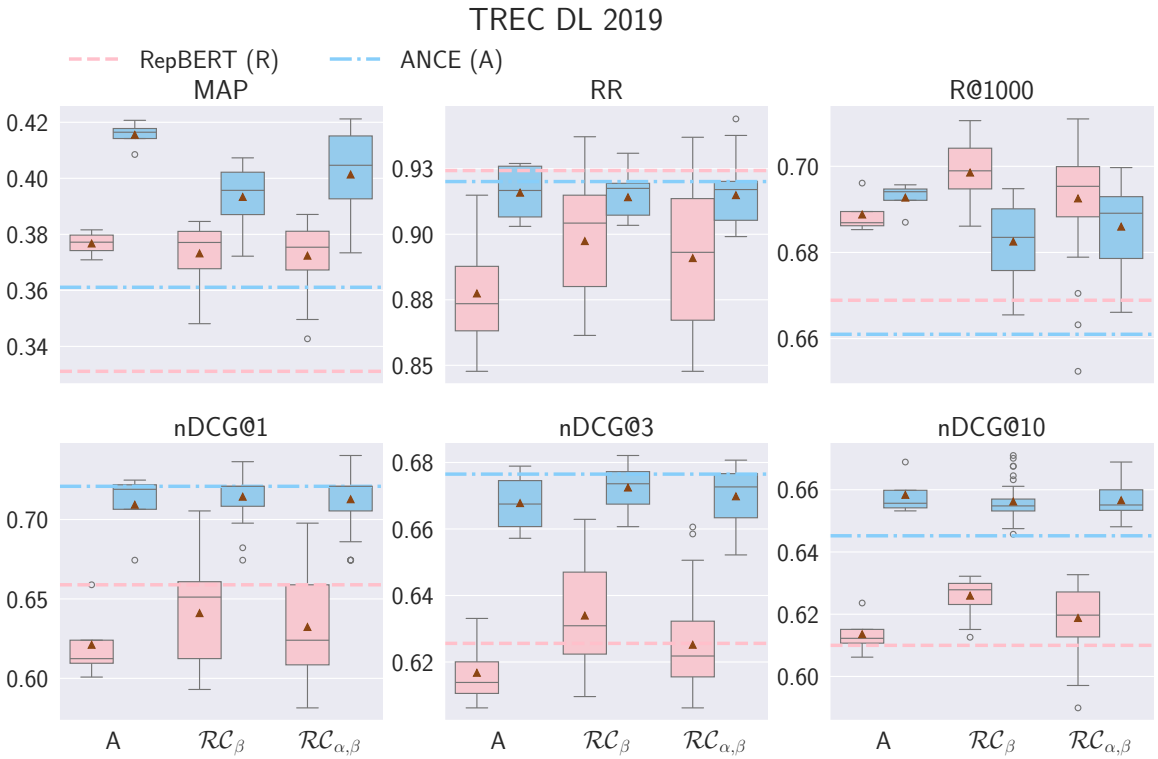


Figure 5.32: Vector-based PRF retrieval effectiveness (y-axis) by using different vector fusion methods on TREC DL 2019. Where A is Vector Average, $\mathcal{RC}_{\beta}$ is Rocchio with fixed α value, and $\mathcal{RC}_{\alpha,\beta}$ is Rocchio with α and β. Baseline RepBERT(R) is marked with dashed red line, ANCE(A) is marked with dash-dot blue line.

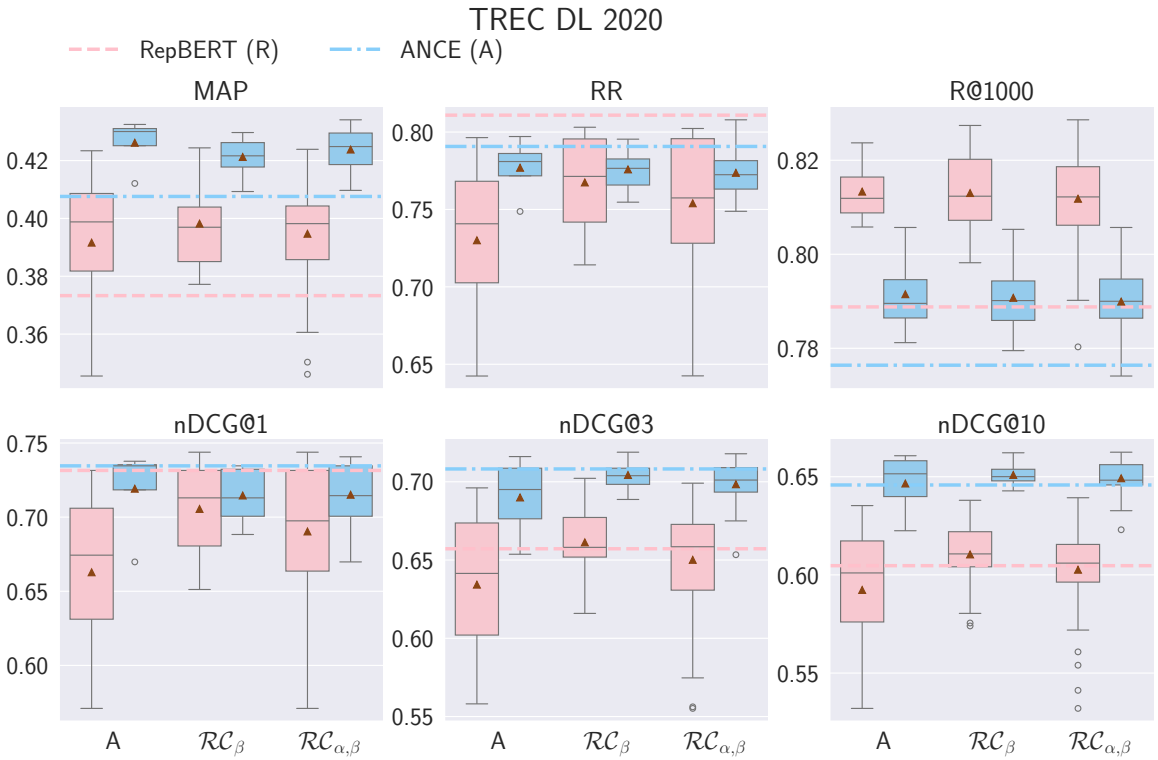


Figure 5.33: Vector-based PRF retrieval effectiveness (y-axis) by using different vector fusion methods on TREC DL 2020. Where A is Vector Average, $\mathcal{RC}_{\beta}$ is Rocchio with fixed α value, and $\mathcal{RC}_{\alpha,\beta}$ is Rocchio with α and β. Baseline RepBERT(R) is marked with dashed red line, ANCE(A) is marked with dash-dot blue line.

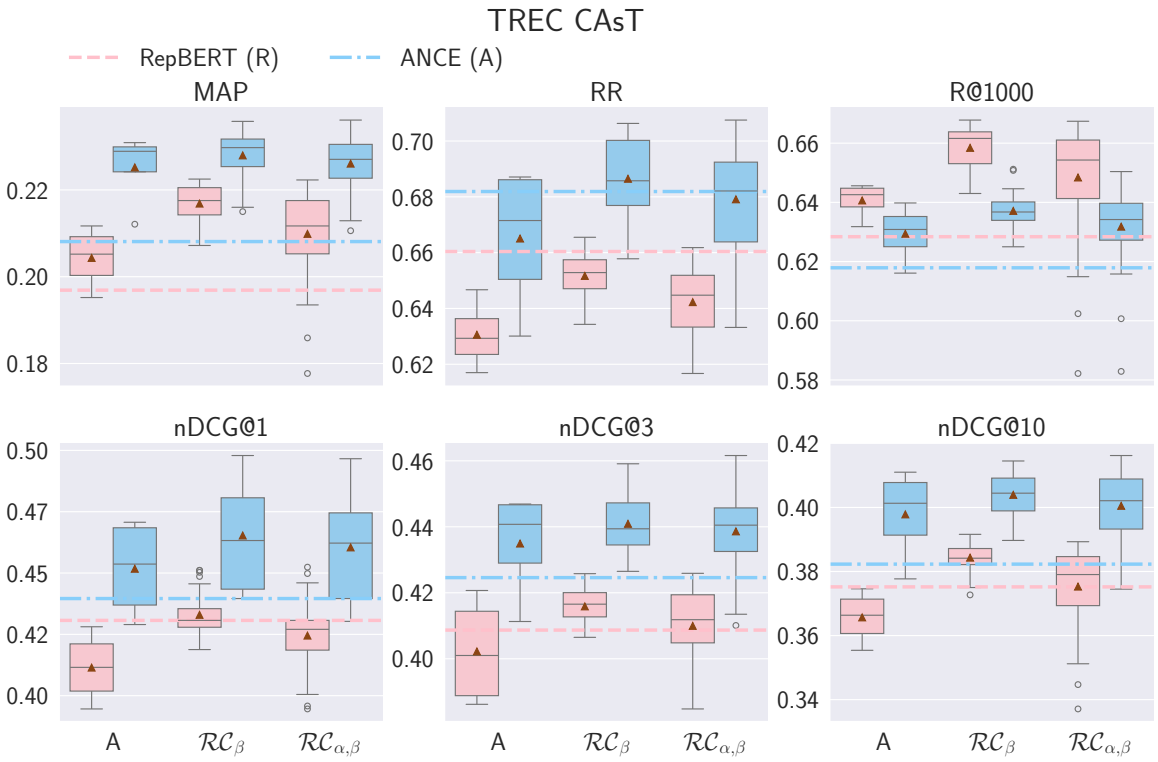


Figure 5.34: Vector-based PRF retrieval effectiveness (y-axis) by using different vector fusion methods on TREC CAsT. Where A is Vector Average, $\mathcal{RC}_{\beta}$ is Rocchio with fixed α value, and $\mathcal{RC}_{\alpha,\beta}$ is Rocchio with α and β. Baseline RepBERT(R) is marked with dashed red line, ANCE(A) is marked with dash-dot blue line.

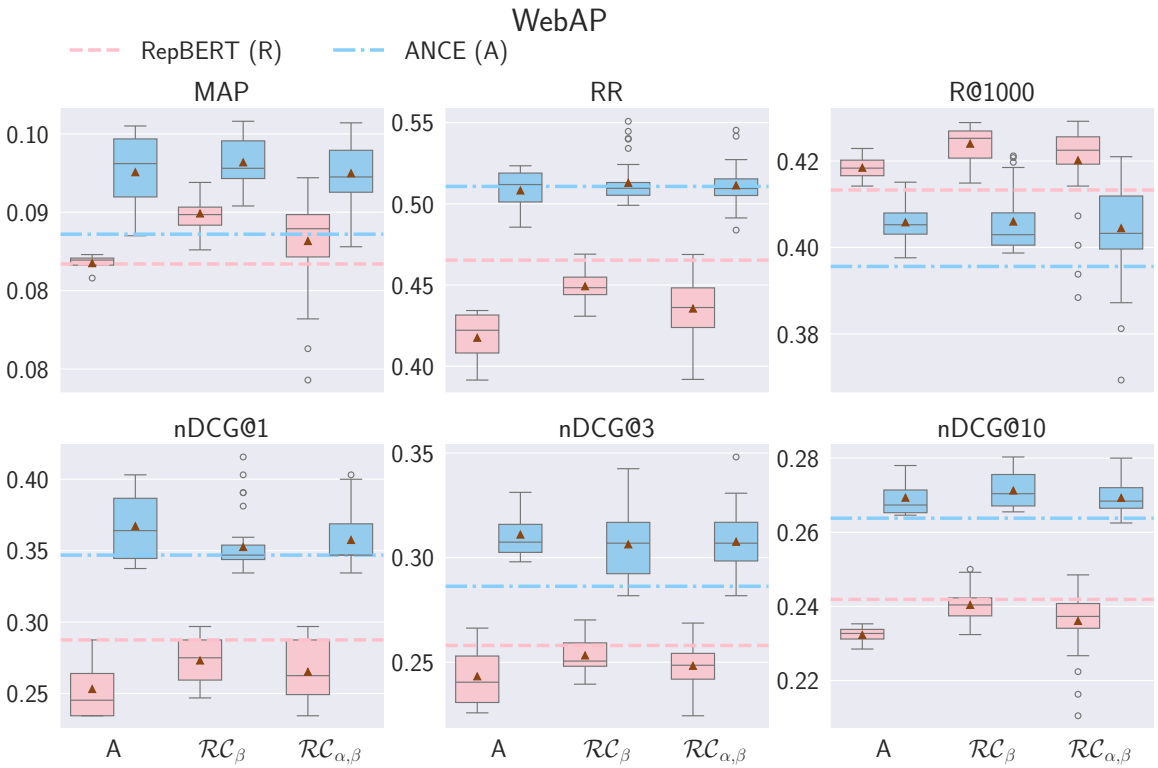


Figure 5.35: Vector-based PRF retrieval effectiveness (y-axis) by using different vector fusion methods on WebAP. Where A is Vector Average, $\mathcal{RC}_{\beta}$ is Rocchio with fixed α value, and $\mathcal{RC}_{\alpha,\beta}$ is Rocchio with α and β. Baseline RepBERT(R) is marked with dashed red line, ANCE(A) is marked with dash-dot blue line.

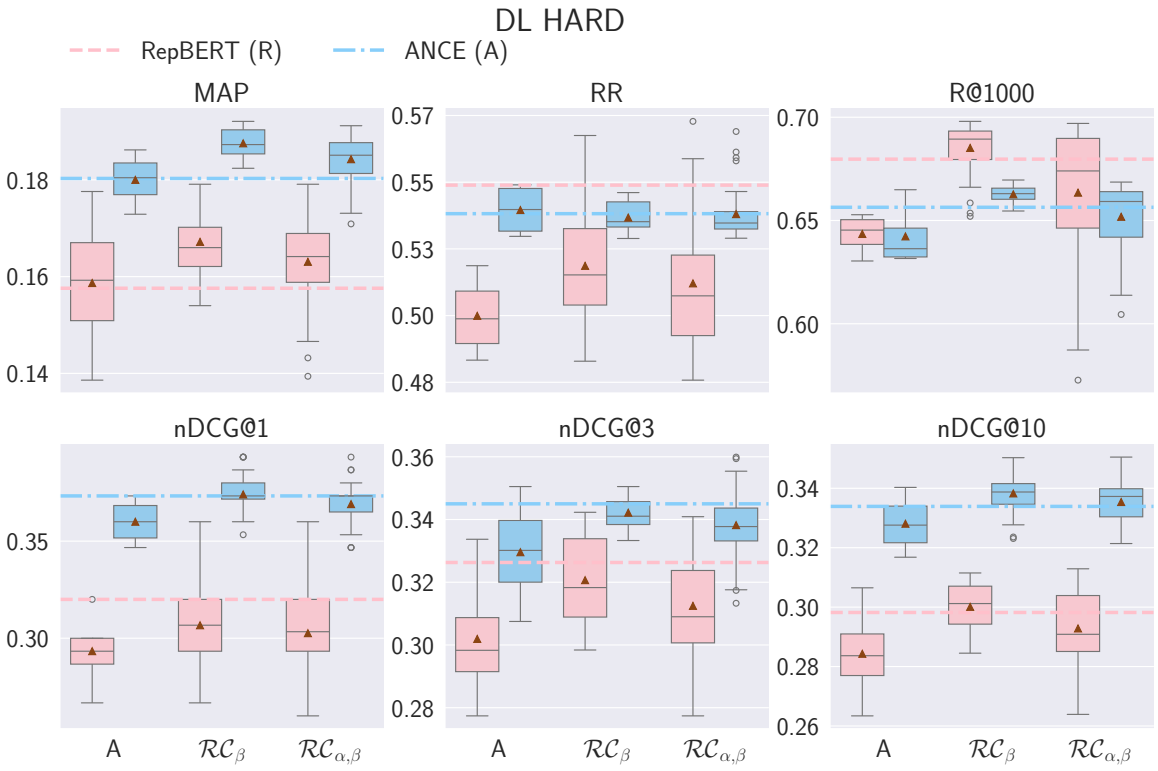


Figure 5.36: Vector-based PRF retrieval effectiveness (y-axis) by using different vector fusion methods on DL HARD. Where A is Vector Average, $\mathcal{RC}_{\beta}$ is Rocchio with fixed α value, and $\mathcal{RC}_{\alpha,\beta}$ is Rocchio with α and β. Baseline RepBERT(R) is marked with dashed red line, ANCE(A) is marked with dash-dot blue line.

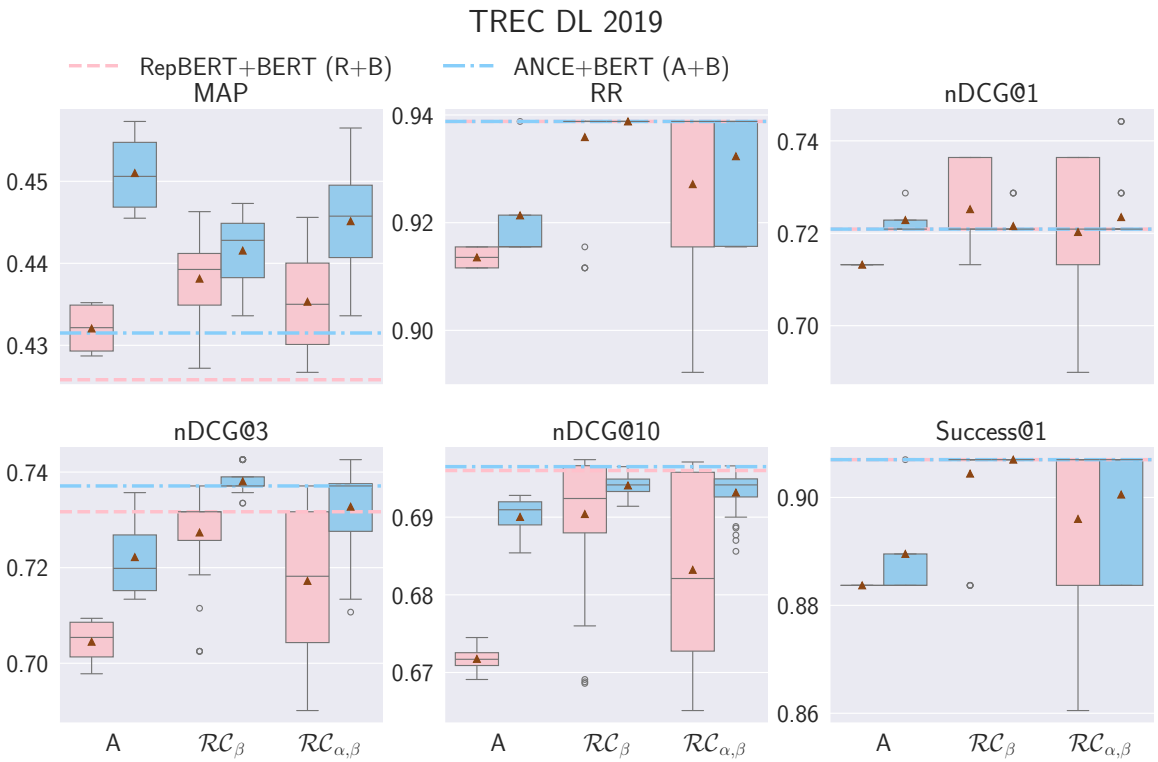


Figure 5.37: Reranking effectiveness (y-axis) by using different score estimation/vector fusion methods on TREC DL 2019. Where A is Vector Average, $\mathcal{RC}_{\beta}$ is Rocchio with fixed α value, and $\mathcal{RC}_{\alpha,\beta}$ is Rocchio with α and β. Baseline ANCE+BERT(A+B) and RepBERT+BERT(R+B) are marked with a dash-dot blue line and a dashed red line respectively.

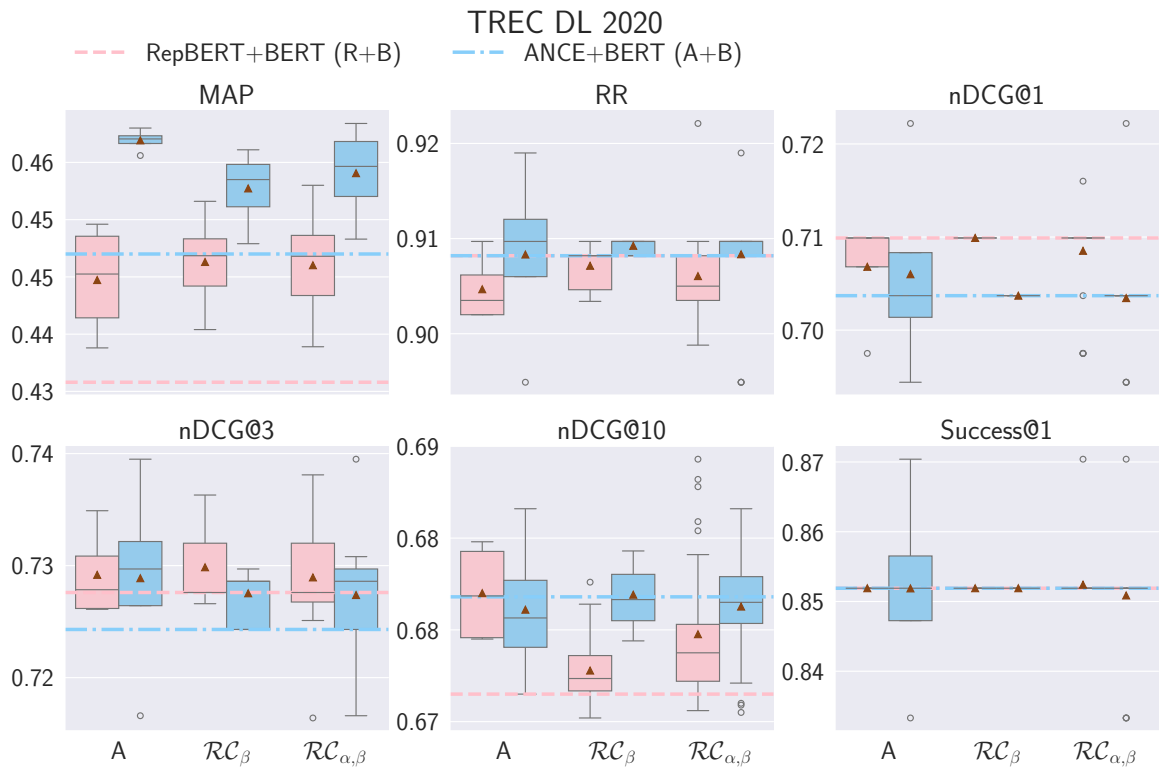


Figure 5.38: Reranking effectiveness (y-axis) by using different score estimation/vector fusion methods on TREC DL 2020. Where A is Vector Average, $\mathcal{RC}_\beta$ is Rocchio with fixed α value, and $\mathcal{RC}_{\alpha,\beta}$ is Rocchio with α and β. Baseline ANCE+BERT(A+B) and RepBERT+BERT(R+B) are marked with a dash-dot blue line and a dashed red line respectively.

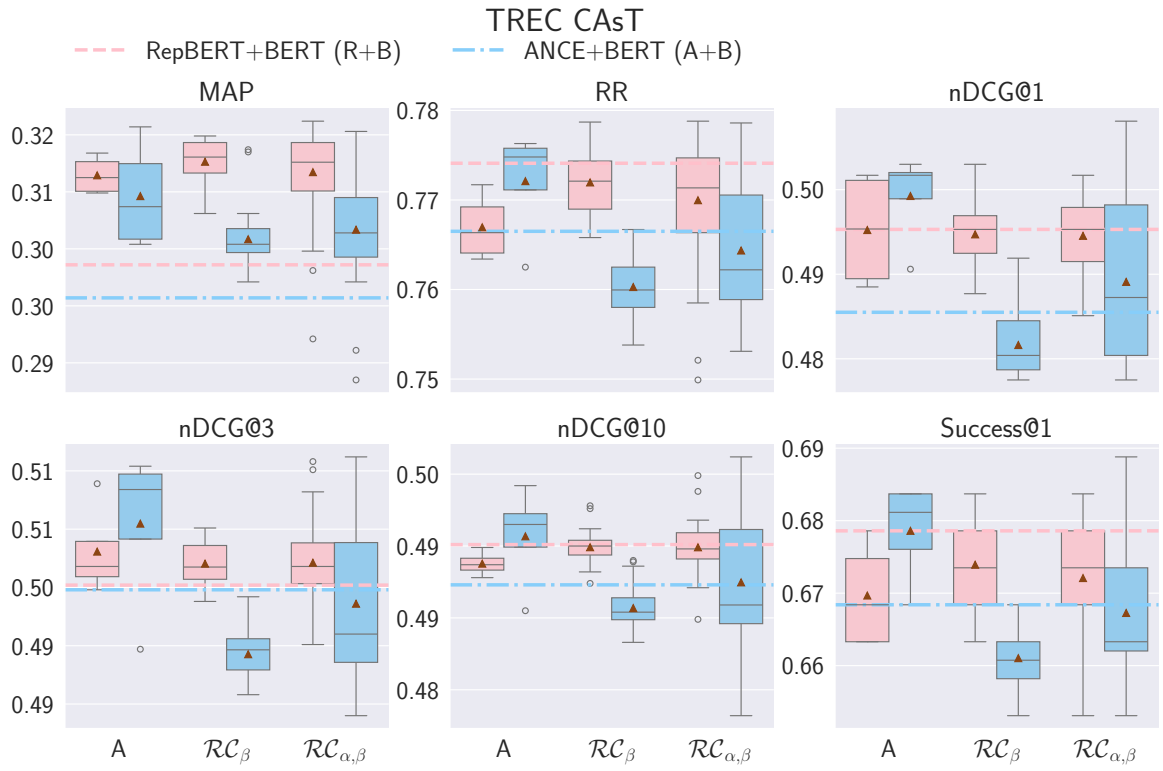


Figure 5.39: Reranking effectiveness (y-axis) by using different score estimation/vector fusion methods on TREC CAsT. Where A is Vector Average, $\mathcal{RC}_\beta$ is Rocchio with fixed α value, and $\mathcal{RC}_{\alpha,\beta}$ is Rocchio with α and β. Baseline ANCE+BERT(A+B) and RepBERT+BERT(R+B) are marked with a dash-dot blue line and a dashed red line respectively.

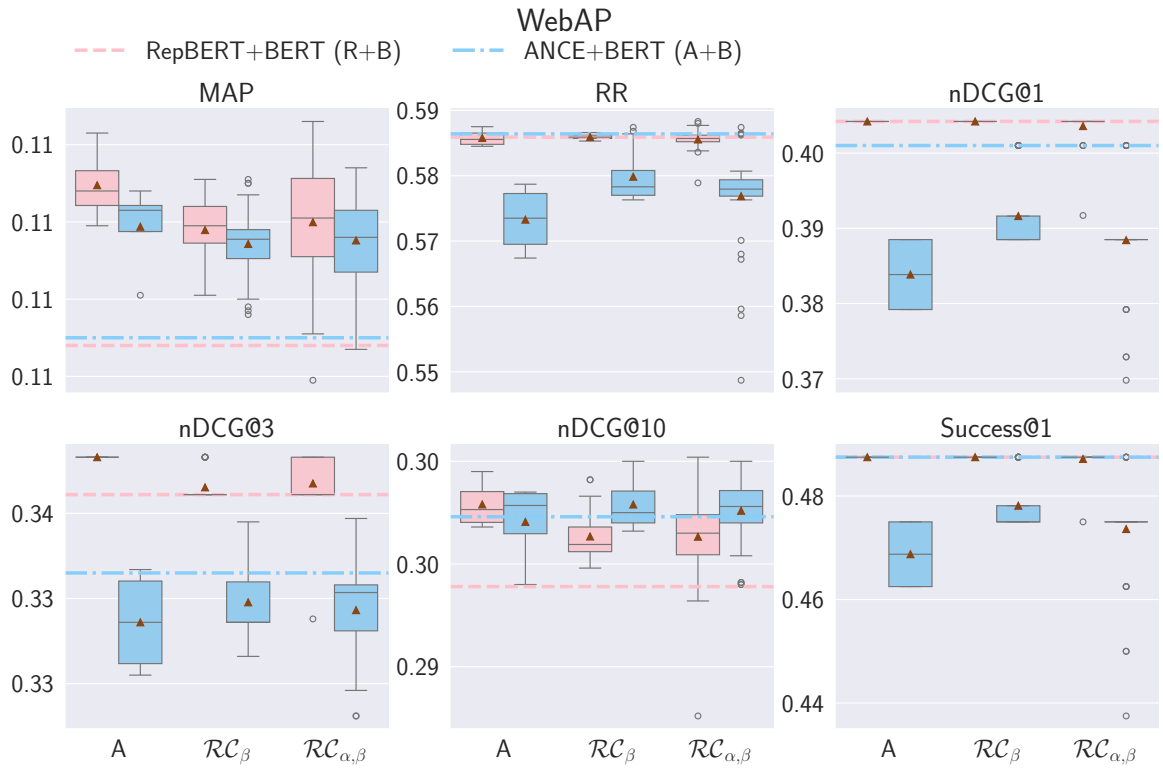


Figure 5.40: Reranking effectiveness (y-axis) by using different score estimation/vector fusion methods on WebAP. Where A is Vector Average, $\mathcal{RC}_\beta$ is Rocchio with fixed α value, and $\mathcal{RC}_{\alpha,\beta}$ is Rocchio with α and β. Baseline ANCE+BERT(A+B) and RepBERT+BERT(R+B) are marked with a dash-dot blue line and a dashed red line respectively.

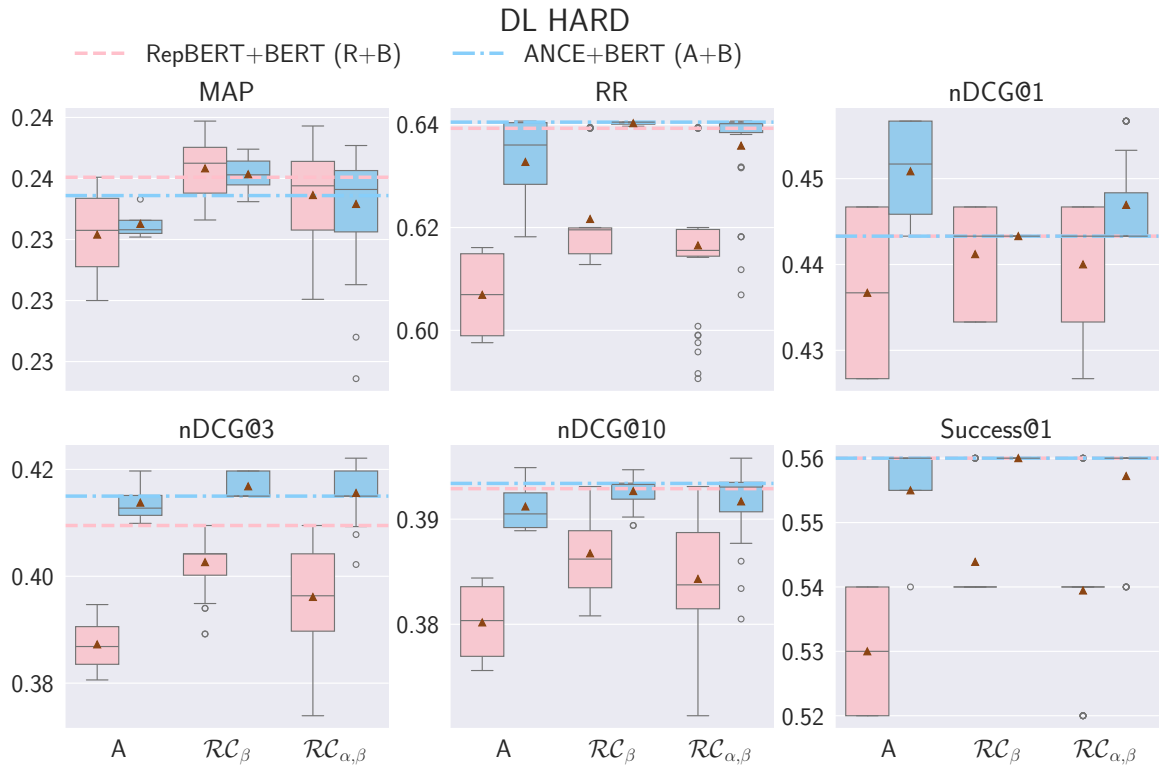


Figure 5.41: Reranking effectiveness (y-axis) by using different score estimation/vector fusion methods on DL HARD. Where A is Vector Average, $\mathcal{RC}_\beta$ is Rocchio with fixed α value, and $\mathcal{RC}_{\alpha,\beta}$ is Rocchio with α and β. Baseline ANCE+BERT(A+B) and RepBERT+BERT(R+B) are marked with a dash-dot blue line and a dashed red line respectively.

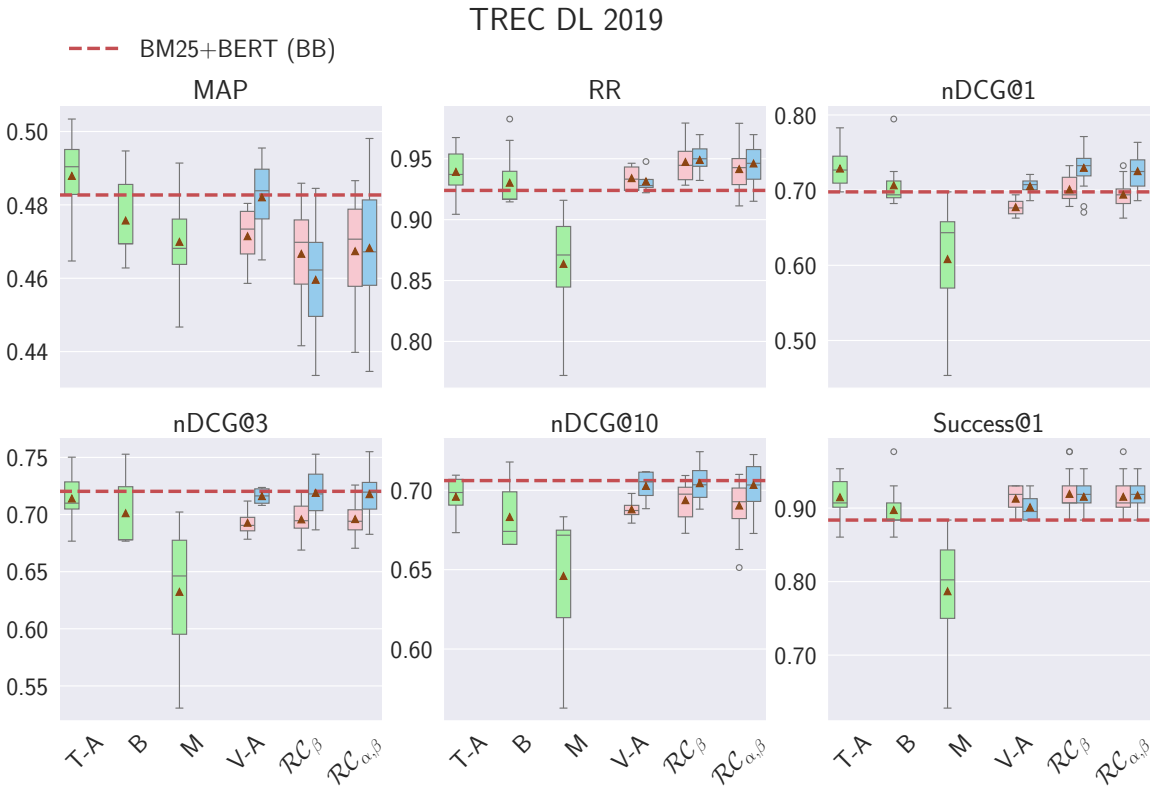


Figure 5.42: Reranking effectiveness (y-axis) by using different score estimation/vector fusion methods on TREC DL 2019. Where T-A is Text Average, B is Borda, M is Max, V-A is Vector Average, $\mathcal{RC}_\beta$ is Rocchio with fixed α value, and $\mathcal{RC}_{\alpha,\beta}$ is Rocchio with α and β. Baseline BM25+BERT(BB) is marked with dashed red line.

MAP. A+PRF-A either matches or improves upon the baseline across all metrics, while R+PRF-R tends to degrade performance in early precision metrics, including nDCG@{1, 3, 10}.

On DL HARD (Figure 5.36), we observe that while A+PRF-A achieves significant improvements in MAP under $\mathcal{RC}_\beta$, performance on other metrics remains comparable to or, in some cases, slightly below the baseline. This reinforces the finding from Chapter 4 that datasets with sparse relevance judgments or "hard" queries are challenging for feedback mechanisms, regardless of the fusion strategy. While vector fusion helps mitigate the severe degradations seen in Text-based PRF, it does not universally guarantee gains over the strong initial dense retriever in these specific adversarial settings.

Reranking With Different Vector Fusions For Vector-Based PRF and Comparison With Text-Based PRF

Figures 5.37 – 5.41 presents the results of applying a BERT reranker on top of Vector-based PRF, to further evaluating different vector fusion strategies. Across all datasets, significant MAP improvements are observed with $\mathcal{RC}_\beta$ and $\mathcal{RC}_{\alpha,\beta}$, confirming their effectiveness as robust fusion methods. These improvements are particularly notable on TREC DL 2019 (Figure 5.37), DL 2020 (Figure 5.38), CAsT (Figure 5.39), and WebAP (Figure 5.40), where MAP gains are both consistent and statistically significant. The averaging method (V-A) also performs well on TREC DL 2019 (Figure 5.37) and 2020 (Figure 5.38), though its improvements are generally more modest.

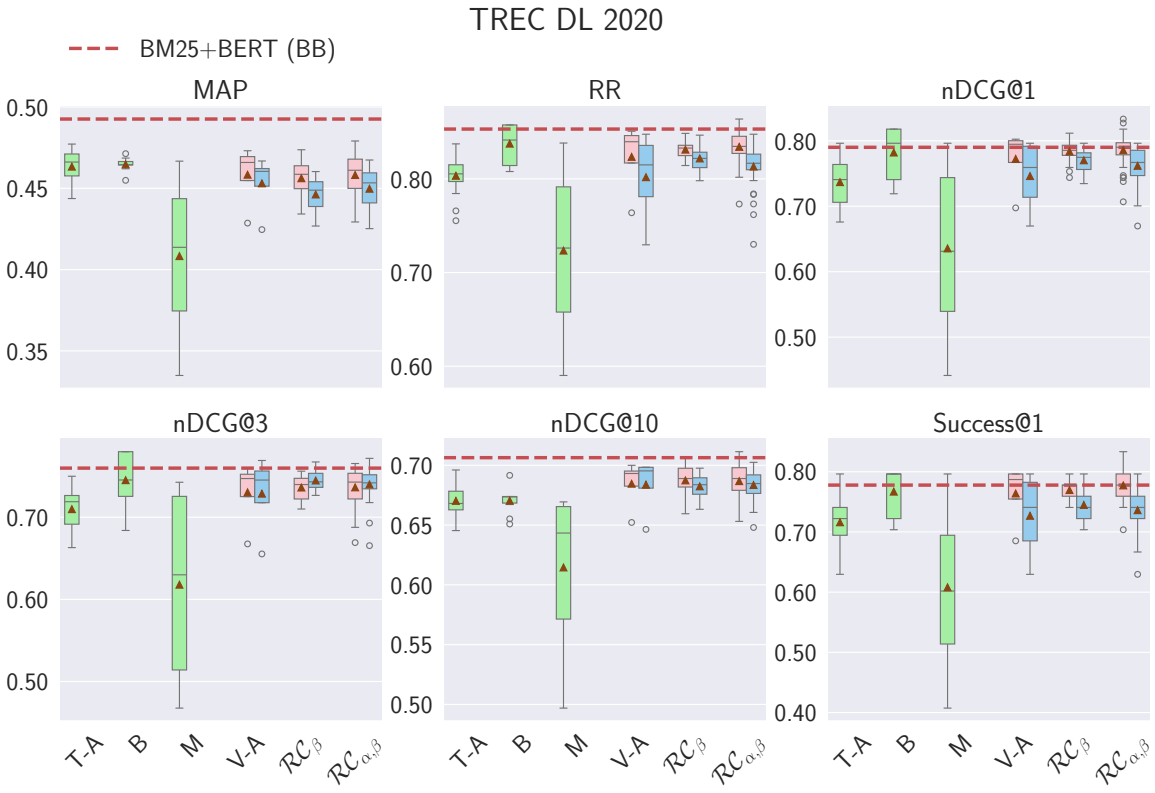


Figure 5.43: Reranking effectiveness (y-axis) by using different score estimation/vector fusion methods on TREC DL 2020. Where T-A is Text Average, B is Borda, M is Max, V-A is Vector Average, $\mathcal{RC}_\beta$ is Rocchio with fixed α value, and $\mathcal{RC}_{\alpha,\beta}$ is Rocchio with α and β. Baseline BM25+BERT(BB) is marked with dashed red line.

While MAP benefits are clear, gains on other metrics (e.g., RR and nDCG@{1, 3, 10}) are mostly marginal, and in some cases, statistically significant degradations are observed, indicating that the improvements do not consistently extend to early precision or ranking quality at shallow depths. An exception to this is found with ANCE-based PRF combined with averaging with a BERT reranker, which significantly improves nDCG@1 on TREC CAsT (Figure 5.39) and DL HARD (Figure 5.41). This suggests that even a simple fusion approach can enhance early ranking performance under certain conditions, especially when paired with a stronger underlying representation. However, these gains remain selective, and their generalizability across metrics and datasets is limited.

In contrast, Figures 5.42 – 5.46 compares Text-based PRF score estimation with Vector-based PRF reranking. While the Text-based method T-A shows strong results on TREC DL 2019 (Figure 5.42) (improving MAP, RR, and nDCG@1), it shows limited or no improvements on other datasets. Method M consistently reduces effectiveness, and B performs well only on a few metrics. Meanwhile, Vector-based rerankers such as BB+PRF-A and BB+PRF-R, particularly with $\mathcal{RC}_\beta$ and $\mathcal{RC}_{\alpha,\beta}$, deliver more stable and broader improvements across datasets, especially in RR and nDCG@1, while also avoiding the volatility seen in Text-based methods.

Overall, these results highlight the robustness of Vector-based PRF also with reranking, particularly when using well-weighted fusion strategies, and its superior consistency compared to Text-based PRF, especially in efficiency and MAP-oriented gains.

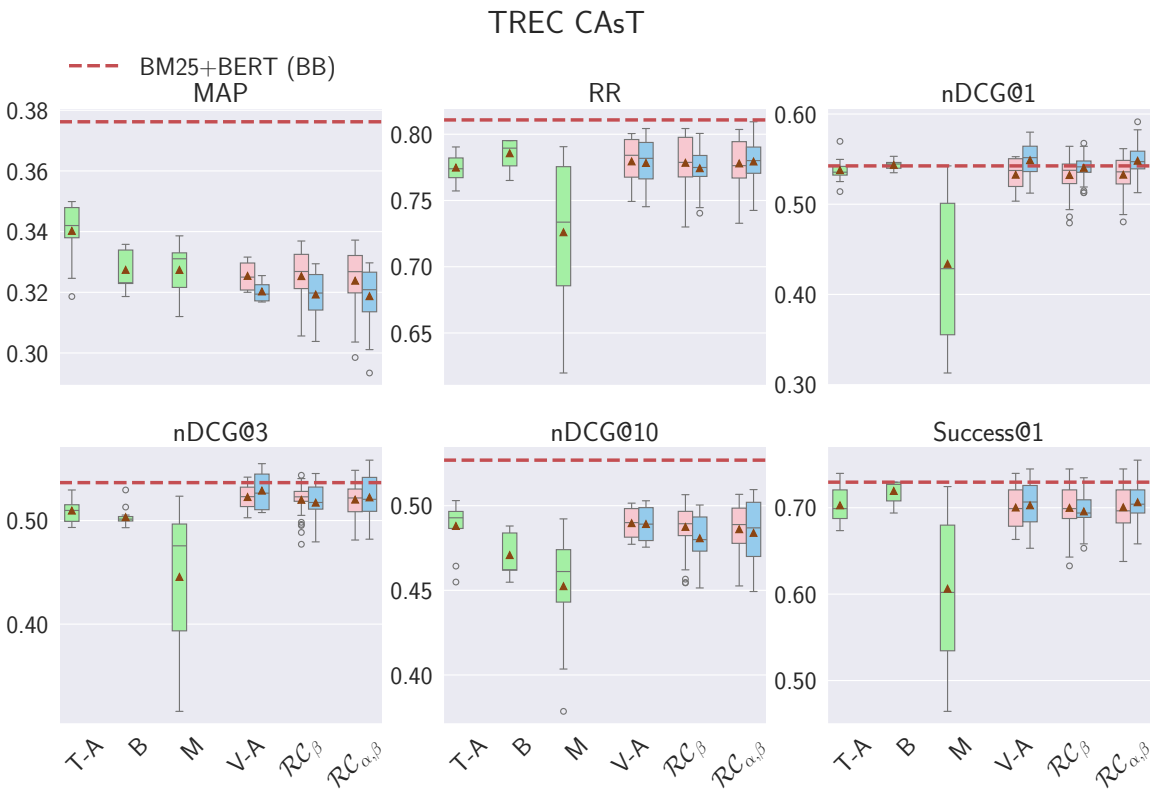


Figure 5.44: Reranking effectiveness (y-axis) by using different score estimation/vector fusion methods on TREC CAsT. Where T-A is Text Average, B is Borda, M is Max, V-A is Vector Average, $\mathcal{RC}_\beta$ is Rocchio with fixed α value, and $\mathcal{RC}_{\alpha,\beta}$ is Rocchio with α and β. Baseline BM25+BERT(BB) is marked with dashed red line.

Summary For RQ1.3.3

To summarize, in the retrieval task, Vector-based PRF demonstrates strong effectiveness when paired with $\mathcal{RC}_{\alpha,\beta}$ and $\mathcal{RC}_\beta$ fusion strategies. Among these, A+PRF-A consistently improves early precision metrics such as nDCG@1 and nDCG@3, while R+PRF-R more reliably enhances deep recall-oriented metrics like Recall@1000 and MAP. This contrast highlights the influence of the underlying dense retriever on how feedback signals are leveraged: ANCE being more effective for top-ranked relevance, and RepBERT providing broader recall improvements.

For the reranking task, both Rocchio variants ($\mathcal{RC}_\beta$ and $\mathcal{RC}_{\alpha,\beta}$) demonstrate strong performance. While $\mathcal{RC}_{\alpha,\beta}$ offers greater theoretical flexibility by tuning both query and feedback weights, our empirical results indicate that its performance is often comparable to the simpler $\mathcal{RC}_\beta$. Although $\mathcal{RC}_{\alpha,\beta}$ achieves the higher median effectiveness on specific datasets (e.g., TREC DL 2020), the overlap in performance distributions suggests that fixing the query weight (as in $\mathcal{RC}_\beta$) is often a sufficient heuristic. In contrast, method M (Max) performs the worst across the board, reinforcing the robustness of well-weighted Vector-based fusion strategies over naive or unstable textual formulations.

When applying a BERT reranker after Vector-based PRF, significant improvements in MAP are observed, especially with the Average fusion strategy, which slightly outperforms both Rocchio variants ($\mathcal{RC}_\beta$, $\mathcal{RC}_{\alpha,\beta}$) in MAP performance. Although Average tends to yield the highest overall effectiveness across datasets, gains beyond MAP remain modest, and improvements on metrics such

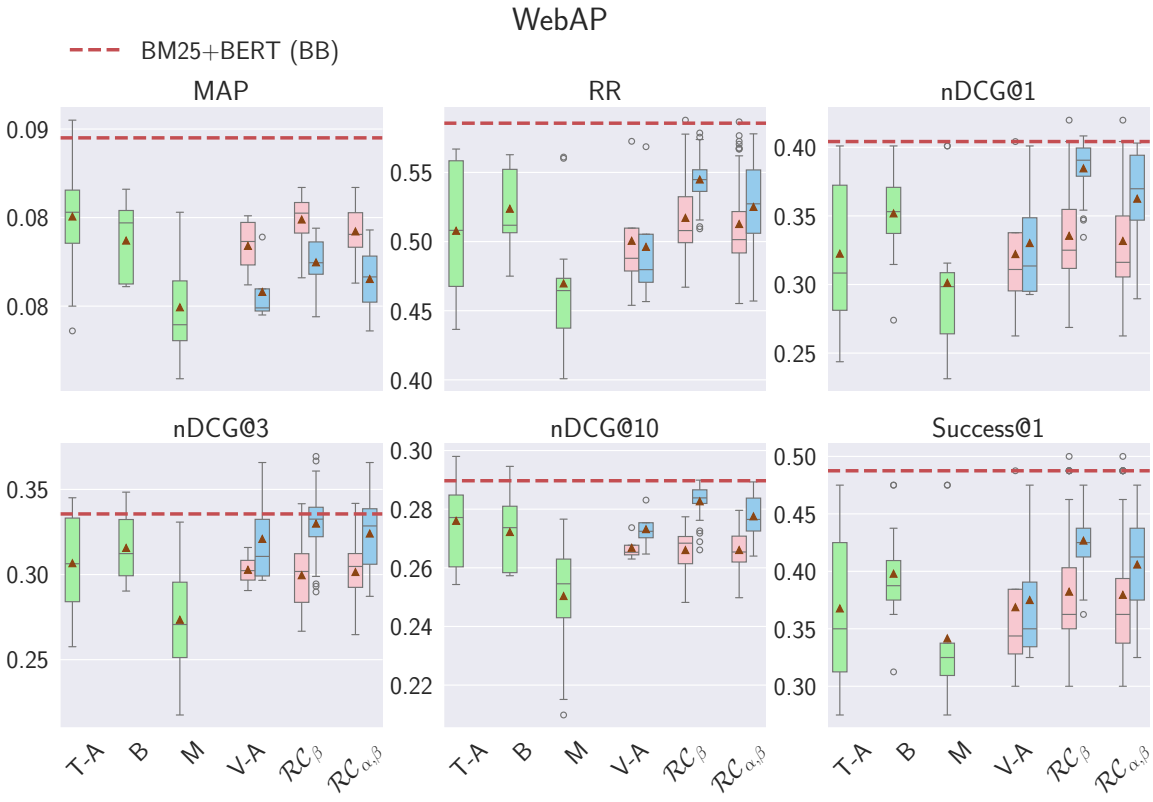


Figure 5.45: Reranking effectiveness (y-axis) by using different score estimation/vector fusion methods on WebAP. Where T-A is Text Average, B is Borda, M is Max, V-A is Vector Average, $\mathcal{RC}_\beta$ is Rocchio with fixed α value, and $\mathcal{RC}_{\alpha,\beta}$ is Rocchio with α and β. Baseline BM25+BERT(BB) is marked with dashed red line.

as nDCG@1 or RR are often marginal. Still, the combination of Vector-based PRF and lightweight fusion presents a compelling trade-off between performance and efficiency, especially when compared to more computationally intensive Text-based reranking methods.

5.4.4 Overall Effectiveness of Vector-Based PRF Models on the Task of Retrieval

RQ1.3.4 investigates the overall effectiveness of Vector-based PRF across both retrieval and reranking tasks. For a more comprehensive comparison, we also include the results of Text-based PRF methods from Table 3.2 as reference points. To address the research question **RQ1.3.4**, we focus exclusively on the best-performing configurations of each PRF model, using the optimal combination of parameters identified in prior experiments. The aggregated results are presented in Tables 5.1 – 5.5.

For each dataset, the middle three rows correspond to PRF-enhanced rerankers: BB+PRF, BB+PRF-R, and BB+PRF-A, while the bottom two rows represent PRF-enhanced retrievers: R+PRF-R and A+PRF-A. Here, BB, R, and A denote BM25+BERT, RepBERT, and ANCE, respectively. Note that Recall@1000 (R@1000) is reported only for retrieval models, particularly in scenarios where reranking with BERT is computationally infeasible.

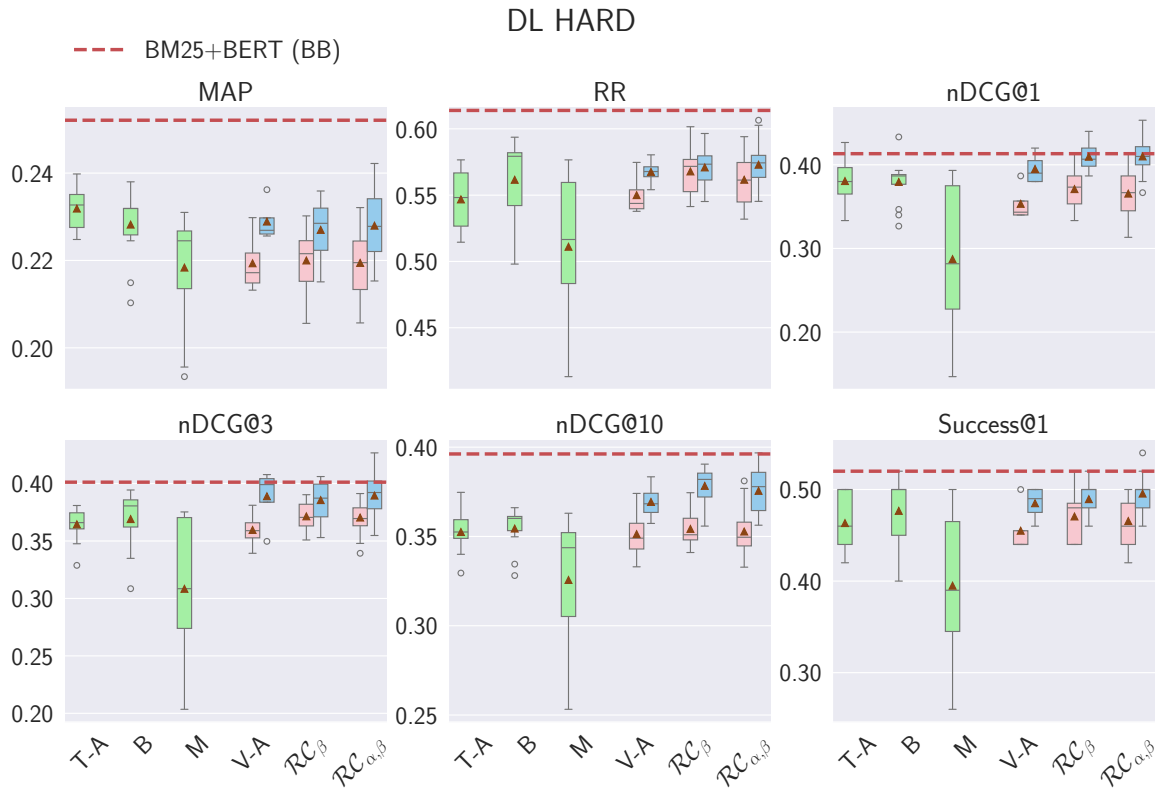


Figure 5.46: Reranking effectiveness (y-axis) by using different score estimation/vector fusion methods on DL HARD. Where T-A is Text Average, B is Borda, M is Max, V-A is Vector Average, $\mathcal{RC}_\beta$ is Rocchio with fixed α value, and $\mathcal{RC}_{\alpha,\beta}$ is Rocchio with α and β. Baseline BM25+BERT(BB) is marked with dashed red line.

Table 5.1: Results of PRF approaches for the tasks of retrieval and reranking for Vector-based and Text-based PRF approaches on TREC DL 2019. For each parametric method, the settings that achieve optimal effectiveness over all metrics are reported. Statistical significance (paired t-test) with $p < 0.05$ between PRF models and BM25 is marked with ^{a}, between PRF models and BM25+RM3 is marked with ^{b}, between PRF and the corresponding baseline is marked with ^{c}, between Vector-Based PRF+BERT and Dense Retriever+BERT is marked with ^{d} (For Vector-Based PRF+BERT we do not compare with other baselines). Best results with respect to each metric are highlighted in **Bold**.

	Model	MAP	RR	nDCG@1	nDCG@3	nDCG@10	R@1000
TREC DL 2019	BM25	0.3773	0.8245	0.5426	0.5230	0.5058	0.7389
	BM25+RM3	0.4270	0.8167	0.5465	0.5195	0.5180	**0.7882**
	BM25+BERT (BB)	0.4827	0.9240	0.6977	0.7203	0.7061	0.7389
	RepBERT+BERT (R+B)	0.4258	0.9388	0.7209	0.7317	0.6960	0.6689
	ANCE+BERT (A+B)	0.4315	0.9388	0.7209	0.7371	0.6965	0.6610
	RepBERT (R)	0.3311	0.9243	0.6589	0.6256	0.6100	0.6689
	ANCE (A)	0.3611	0.9201	0.7209	0.6765	0.6452	0.6610
	BB+PRF($k = 10$,CA,BORDA)	0.4947^{ab}	$\mathbf{0.9826}^{abc}$	$\mathbf{0.7946}^{abc}$	$\mathbf{0.7528}^{abc}$	0.7178^{ab}	–
	BB+PRF-R($k = 10, \beta = .3$)	0.4705^{a}	0.9793^{ab}	0.7326^{ab}	0.6963^{ab}	0.6993^{ab}	–
	BB+PRF-A($k = 10, \alpha = .3, \beta = .7$)	$\mathbf{0.4955}^{ab}$	0.9690^{ab}	0.7519^{ab}	0.7385^{ab}	$\mathbf{0.7210}^{ab}$	–
	R+PRF+B($k = 5, \beta = .5$)	0.4463^{d}	0.9388	0.7209	0.7371	0.6968	0.7097^{d}
	A+PRF+B($k = 5, \alpha = .4, \beta = .6$)	0.4514^{d}	0.9388	0.7209	0.7390	0.6953	0.6997^{d}
	R+PRF-R($k = 10, \beta = .3$)	0.3669^{c}	0.9368	0.7054^{c}	0.6559^{abc}	0.6252^{ab}	0.7012^{bc}
	A+PRF-A($k = 10, \alpha = .4, \beta = .6$)	0.4151^{c}	0.9440^{a}	0.7403^{ab}	0.6807^{ab}	0.6629^{ab}	0.6962^{bc}

Table 5.2: Results of PRF approaches for the tasks of retrieval and reranking for Vector-based and Text-based PRF approaches on TREC DL 2020. For each parametric method, the settings that achieve optimal effectiveness over all metrics are reported. Statistical significance (paired t-test) with $p < 0.05$ between PRF models and BM25 is marked with ^{a}, between PRF models and BM25+RM3 is marked with ^{b}, between PRF and the corresponding baseline is marked with ^{c}, between Vector-Based PRF+BERT and Dense Retriever+BERT is marked with ^{d} (For Vector-Based PRF+BERT we do not compare with other baselines). Best results with respect to each metric are highlighted in **Bold**.

	Model	MAP	RR	nDCG@1	nDCG@3	nDCG@10	R@1000
TREC DL 2020	BM25	0.2856	0.6585	0.5772	0.5021	0.4796	0.7863
	BM25+RM3	0.3019	0.6360	0.5648	0.4740	0.4821	**0.8217**
	BM25+BERT (BB)	**0.4926**	0.8531	0.7901	0.7598	0.7064	0.7863
	RepBERT+BERT (R+B)	0.4358	0.9082	0.7099	0.7276	0.6715	0.6593
	ANCE+BERT (A+B)	0.4470	0.9082	0.7037	0.7243	0.6768	0.6819
	RepBERT (R)	0.3733	0.8109	0.7315	0.6572	0.6047	0.7888
	ANCE (A)	0.4076	0.7907	0.7346	0.7082	0.6458	0.7764
	BB+PRF($k = 3$,SW,BORDA)	0.4644^{ab}	0.8575^{ab}	0.8179^{ab}	$\mathbf{0.7798}^{ab}$	0.6739^{ab}	–
	BB+PRF-R($k = 5, \alpha = .4, \beta = .6$)	0.4778^{ab}	0.8638^{ab}	$\mathbf{0.8333}^{ab}$	0.7544^{ab}	$\mathbf{0.7111}^{ab}$	–
	BB+PRF-A($k = 1, \alpha = .5, \beta = .5$)	0.4606^{abc}	0.8476^{ab}	0.7963^{ab}	0.7691^{ab}	0.6984^{ab}	–
	R+PRF+B($k = 3, \alpha = .4, \beta = .6$)	0.4530^{d}	0.9050	0.7099	0.7320	0.6750	0.7022^{d}
	A+PRF+B($k = 3, \alpha = .4, \beta = .6$)	0.4584^{d}	**0.9097**	0.7037	0.7297	0.6791	0.7019^{d}
	R+PRF-R($k = 1, \alpha = .6, \beta = .4$)	0.4239^{abc}	0.7951^{ab}	0.7315^{ab}	0.6991^{ab}	0.6393^{ab}	0.8159^{c}
	A+PRF-A($k = 3, \alpha = .4, \beta = .6$)	0.4341^{abc}	0.8079^{ab}	0.7407^{ab}	0.7117^{ab}	0.6598^{ab}	0.7948

Table 5.3: Results of PRF approaches for the tasks of retrieval and reranking for Vector-based and Text-based PRF approaches on TREC CAsT. For each parametric method, the settings that achieve optimal effectiveness over all metrics are reported. Statistical significance (paired t-test) with $p < 0.05$ between PRF models and BM25 is marked with ^{a}, between PRF models and BM25+RM3 is marked with ^{b}, between PRF and the corresponding baseline is marked with ^{c}, between Vector-Based PRF+BERT and Dense Retriever+BERT is marked with ^{d} (For Vector-Based PRF+BERT we do not compare with other baselines). Best results with respect to each metric are highlighted in **Bold**.

	Model	MAP	RR	nDCG@1	nDCG@3	nDCG@10	R@1000
TREC CAsT	BM25	0.2936	0.6502	0.3631	0.3542	0.3526	**0.8326**
	BM25+RM3	0.3132	0.6556	0.3971	0.3829	0.3817	0.8246
	BM25+BERT (BB)	**0.3762**	**0.8108**	0.5425	0.5366	**0.5269**	**0.8326**
	RepBERT+BERT (R+B)	0.3036	0.7741	0.4953	0.5002	0.4901	0.6284
	ANCE+BERT (A+B)	0.3007	0.7665	0.4855	0.4998	0.4890	0.6179
	RepBERT (R)	0.1969	0.6604	0.4307	0.4087	0.3752	0.6284
	ANCE (A)	0.2081	0.6819	0.4396	0.4246	0.3823	0.6179
	BB+PRF($k = 10$,CC)	0.3247^{ac}	0.8106^{ab}	0.5510^{abc}	0.5140^{ab}	0.4838^{abc}	–
	BB+PRF-R($k = 3, \alpha = .5, \beta = .5$)	0.3372^{ac}	0.7985^{ab}	0.5480^{ab}	0.5468^{ab}	0.5067^{abc}	–
	BB+PRF-A($k = 3, \alpha = .3, \beta = .7$)	0.3274^{ac}	0.8093^{ab}	$\mathbf{0.5914}^{ab}$	$\mathbf{0.5583}^{ab}$	0.5055^{abc}	–
	R+PRF+B($k = 5, \alpha = .4, \beta = .6$)	0.3162^{d}	0.7722	0.4915	0.5023	0.4909	0.6635^{d}
	A+PRF+B($k = 3, \alpha = .3, \beta = .7$)	0.3153^{d}	0.7725	0.4991	0.5043	0.4939	0.6432^{d}
	R+PRF-R($k = 10, \alpha = .8, \beta = .2$)	0.2150^{abc}	0.6618	0.4498^{a}	0.4146^{a}	0.3844^{c}	0.6566^{abc}
	A+PRF-A($k = 3, \beta = .9$)	0.2347^{abc}	0.6826	0.4626^{a}	0.4434^{abc}	0.4138^{ac}	0.6508^{abc}

Table 5.4: Results of PRF approaches for the tasks of retrieval and reranking for Vector-based and Text-based PRF approaches on WebAP. For each parametric method, the settings that achieve optimal effectiveness over all metrics are reported. Statistical significance (paired t-test) with $p < 0.05$ between PRF models and BM25 is marked with ^{a}, between PRF models and BM25+RM3 is marked with ^{b}, between PRF and the corresponding baseline is marked with ^{c}, between Vector-Based PRF+BERT and Dense Retriever+BERT is marked with ^{d} (For Vector-Based PRF+BERT we do not compare with other baselines). Best results with respect to each metric are highlighted in **Bold**.

	Model	MAP	RR	nDCG@1	nDCG@3	nDCG@10	R@1000
WEBAP	BM25	0.0436	0.3099	0.1667	0.1604	0.1404	0.2944
	BM25+RM3	0.0536	0.2767	0.1344	0.1316	0.1376	0.3472
	BM25+BERT (BB)	0.0845	0.5856	0.4042	0.3356	0.2897	0.2944
	RepBERT+B (R+B)	0.1088	0.5859	0.4042	0.3361	0.2939	0.4133
	ANCE+BERT (A+B)	0.1090	0.5846	0.4010	0.3315	0.2973	0.3956
	RepBERT (R)	0.0867	0.4653	0.2875	0.2580	0.2419	0.4133
	ANCE (A)	0.0886	0.5107	0.3469	0.2863	0.2638	0.3956
	BB+PRF($k = 15$,CA,AVG)	0.0855^{ab}	0.5459^{ab}	0.3271^{ab}	$\mathbf{0.3444}^{ab}$	0.2980^{ab}	–
	BB+PRF-R($k = 1, \alpha = .7, \beta = .3$)	0.0809^{ab}	0.5866^{ab}	$\mathbf{0.4198}^{ab}$	0.3418^{ab}	0.2708^{ab}	–
	BB+PRF-A($k = 3, \beta = .8$)	0.0790^{abc}	0.5502^{ab}	0.4083^{ab}	0.3394^{ab}	0.2842^{ab}	–
	R+PRF+B($k = 3, \alpha = .3, \beta = .7$)	$\mathbf{0.1146}^{d}$	**0.5880**	0.4042	0.3383	0.2995^{d}	**0.4253**
	A+PRF+B($k = 5, \alpha = .4, \beta = .6$)	0.1134^{d}	0.5790	0.3885	0.3308	**0.2996**	0.4202
	R+PRF-R($k = 3, \beta = .1$)	0.0887^{abc}	0.4690^{ab}	0.2969^{ab}	0.2594^{ab}	0.2433^{ab}	0.4206^{abc}
	A+PRF-A($k = 3, \beta = .9$)	0.0953^{abc}	0.5134^{ab}	0.3563^{ab}	0.2928^{ab}	0.2710^{ab}	0.4027^{ab}

Effectivaness For Retrieval With Vector-Based PRF (R+PRF-R, A+PRF-A)

For R+PRF-R, consistent trends are observed across all datasets: the model improves upon the RepBERT baseline on all reported metrics: MAP, RR, Recall@1000, and nDCG@{1, 3, 10}, with the exception of TREC DL 2020 (Table 5.2), where RR slightly declines relative to the baseline.

For A+PRF-A, the model consistently outperforms the ANCE baseline across all evaluation metrics and datasets. Statistically significant improvements in MAP are observed on TREC DL 2019 (Table 5.1), TREC DL 2020 (Table 5.2), TREC CAsT (Table 5.3), and WebAP (Table 5.4). Notable gains in Recall@1000 are achieved on TREC DL 2019 (Table 5.1), TREC CAsT (Table 5.3), and DL HARD (Table 5.5). For nDCG@{3, 10}, significant improvements are observed primarily on TREC CAsT (Table 5.3). Overall, A+PRF-A demonstrates superior effectiveness compared to R+PRF-R, attributable in part to the stronger retrieval quality of the ANCE baseline, which encodes richer semantic signals. Consequently, the PRF mechanism operating on ANCE embeddings is better positioned to extract and incorporate relevance information from feedback passages.

In summary, both A+PRF-A and R+PRF-R perform robustly across datasets and metrics, delivering substantial gains over traditional baselines including BM25, BM25+RM3, and in several cases, BM25+BERT. These results highlight the effectiveness of our Vector-based PRF framework when integrated with strong dense retrievers.

Effectiveness For Reranking with BERT on Top of Vector-Based PRF (R+PRF+B, A+PRF+B)

R+PRF+B yields significant improvements in MAP and Recall@1000 across all datasets except WebAP (Table 5.4). Other metrics are either on par with or slightly better than the baseline, though not statistically significant. Notably, nDCG@10 significantly improves on WebAP (Table 5.4); however, the overall gain in Recall@1000 is primarily attributed to the transition from R to R+PRF, indicating that the reranking step contributes minimally to this improvement.

A+PRF+B follows a similar pattern, achieving significant gains in MAP and Recall@1000 across all datasets except WebAP. While it achieves the highest nDCG@10 on WebAP (Table 5.4), the difference is not statistically significant. Other metrics remain largely unchanged or show marginal, non-significant improvements over the baseline.

In summary, applying a BERT reranker on top of Vector-based PRF significantly enhances MAP, but improvements in other metrics are generally modest and not statistically significant.

Effectiveness For Reranking with Vector-Based PRF (BB+PRF-R, BB+PRF-A)

When using RepBERT as the base model (BB+PRF-R), TREC DL 2019 shows improvements in RR and nDCG@1, but no gains on other metrics. For TREC DL 2020 (Table 5.2), improvements are observed only in shallow metrics – RR and nDCG@1, 10. On TREC CAsT 2019 (Table 5.1), while nDCG@1, 3 improves, both MAP and nDCG@10 degrade significantly. WebAP (Table 5.4) shows modest gains in RR and nDCG@1, 3, while DL HARD (Table 5.5) shows no improvement across any metric and performs significantly worse than the baseline on MAP and nDCG@10.

With ANCE as the base model (BB+PRF-A), TREC DL 2019 (Table 5.1) exhibits consistent gains across all shallow metrics (RR, nDCG@1, 3, 10) as well as MAP. For TREC DL 2020 (Table 5.2) and DL HARD (Table 5.5), improvements are observed in nDCG@1, 3, 10. On TREC CAsT (Table 5.3) and WebAP (Table 5.4), the model improves nDCG@1, 3, though deeper metrics remain unchanged.

In summary, the proposed Vector-based PRF rerankers (BB+PRF-R, BB+PRF-A): (1) improve effectiveness over BM25+BERT on several metrics across all datasets; (2) outperform BM25, BM25+RM3, and RepBERT/ANCE baselines in most settings, with the exception of DL HARD (Table 5.5); (3) show mixed results when compared to BM25+BERT, with no consistent trend across metrics or datasets.

Effectiveness For Reranking with Text-Based PRF (BB+PRF) For Comparison With Vector-Based PRF

Compared to Vector-based PRF, BB+PRF delivers only marginal gains. On TREC DL 2019 (Table 5.1), it improves shallow metrics (RR, nDCG@1, 3) significantly, while gains on DL 2020 (Table 5.2) are limited and not significant. It fails to improve over BM25+BERT on TREC CAsT (Table 5.3), likely due to alignment between short queries and the reranker's training. On WebAP (Table 5.4), it improves MAP and nDCG@3, 10, where BERT struggles with longer passages. For DL HARD (Table 5.5), it performs poorly, with a significant drop in nDCG@10 due to noisy feedback and query drift. In

Table 5.5: Results of PRF approaches for the tasks of retrieval and reranking for Vector-based and Text-based PRF approaches on DL-HARD. For each parametric method, the settings that achieve optimal effectiveness over all metrics are reported. Statistical significance (paired t-test) with $p < 0.05$ between PRF models and BM25 is marked with ^{a}, between PRF models and BM25+RM3 is marked with ^{b}, between PRF and the corresponding baseline is marked with ^{c}, between Vector-Based PRF+BERT and Dense Retriever+BERT is marked with ^{d} (For Vector-Based PRF+BERT we do not compare with other baselines). Best results with respect to each metric are highlighted in **Bold**.

	Model	MAP	RR	nDCG@1	nDCG@3	nDCG@10	R@1000
DL-HARD	BM25	0.1845	0.5422	0.3533	0.3137	0.2850	0.6288
	BM25+RM3	0.1925	0.4381	0.2467	0.2508	0.2555	0.6522
	BM25+BERT (BB)	**0.2521**	0.6139	0.4133	0.4012	0.3962	0.6288
	RepBERT+B (R+B)	0.2401	0.6393	0.4433	0.4095	0.3929	0.6797
	ANCE+BERT (A+B)	0.2386	**0.6405**	0.4433	0.4150	0.3934	0.6564
	RepBERT (R)	0.1576	0.5489	0.3200	0.3263	0.2982	0.6797
	ANCE (A)	0.1803	0.5382	0.3733	0.3450	0.3339	0.6564
	BB+PRF($k = 3$,CA,BORDA)	0.2380^a	0.5937^b	0.4333^b	0.3944^b	0.3550^{abc}	–
	BB+PRF-R($k = 5, \alpha = .8, \beta = .2$)	0.2255^c	0.5843^{ab}	0.3867^b	0.3861^b	0.3646^{abc}	–
	BB+PRF-A($k = 5, \alpha = .4, \beta = .6$)	0.2422^a	0.5904^{ab}	0.4333^b	$\mathbf{0.4267}^{ab}$	$\mathbf{0.3968}^{ab}$	–
	R+PRF+B($k = 5, \beta = .2$)	0.2439^d	0.6394	**0.4433**	0.4095	0.3926	$\mathbf{0.6968}^d$
	A+PRF+B($k = 5, \alpha = .8, \beta = .2$)	0.2419^d	**0.6405**	**0.4433**	0.4150	0.3941	0.6683^d
	R+PRF-R($k = 5, \alpha = .9, \beta = .1$)	0.1654	0.5504^b	0.3333	0.3368	0.3030	0.6929^c
	A+PRF-A($k = 10, \beta = .4$)	0.1865	0.5426	0.3933^b	0.3453	0.3380^b	0.6681^c

contrast, Vector-based PRF shows broader, more consistent improvements and is not constrained by passage length or BERT-specific limitations.

Vector-Based PRF Generalizability to Other Dense Retrievers

The results shown in Tables 5.6 and 5.7 indicate that the effectiveness of Vector-based PRF varies depending on the strength of the underlying retriever and the evaluation metric. While VPRF demonstrates a tendency to improve recall-oriented metrics (MAP, nDCG@100, and R@1000) across most models on TREC DL 2019 and 2020, its impact on early precision is mixed. For stronger dense retrievers (e.g., TCT-ColBERT V2), the original baseline often performs on par with or outperforms the VPRF-enhanced model in terms of RR and nDCG@1. This suggests that while VPRF successfully enriches the result set with relevant passages (aiding Recall and MAP), it does not guarantee improved top-rank precision for retrievers that are already highly optimized.

Summary For RQ1.3.4

To address **RQ1.3.4**, our results indicate that Vector-based PRF offers meaningful effectiveness gains, particularly in retrieval settings, though the magnitude of improvement depends on the metric. Specifically, R+PRF-R and A+PRF-A improve upon their respective baselines in the majority of deep metrics (MAP, Recall, nDCG@10) across datasets. However, gains in shallow metrics (RR, nDCG@1) are less uniform, with the baseline sometimes retaining the advantage. In contrast, BB+PRF

Table 5.6: Results of Vector-based PRF for the task of retrieval on TREC DL 2019, using dense retrievers more effective than ANCE and RepBERT. To demonstrate the generalizability of our approach without model-specific tuning, we fix the hyperparameters to the settings that were optimal for the ANCE baseline ($\alpha = 0.4$ and $\beta = 0.6$). We use a fixed PRF depth of 3 for Average and 5 for Rocchio. The best results for each model are marked in **Bold**. Significant improvements are marked with †.

	Model	Method	MAP	RR	nDCG@1	nDCG@3	nDCG@10	nDCG@100	R@1000
TREC DL 2019	TCT-ColBERT V1	Original	0.3864	**0.9512**	**0.7326**	0.6874	0.6700	0.5730	0.7207
		Average	0.4457	0.8999	0.6705	0.6779	0.6639	0.6119	0.7570
		Rocchio	**0.4479**†	0.9368	0.7093	**0.7083**†	**0.6875**	**0.6143**†	**0.7720**
	TCT-ColBERT V2 HN+	Original	0.4626	**0.9767**	**0.8023**	0.7410	0.7204	0.6318	0.7603
		Average	0.5123	**0.9767**	0.7713	**0.7454**	**0.7312**	**0.6719**	0.8115
		Rocchio	**0.5161**†	0.9244	0.7248	0.7129	0.7111	0.6684	**0.8147**†
	DistilBERT KD	Original	0.3759	0.9306	**0.7558**	**0.7370**	0.6994	0.5765	0.6853
		Average	0.4362	0.9253	0.7481	0.7241	**0.7096**	**0.6217**	0.7180
		Rocchio	**0.4378**†	**0.9345**	0.7442	0.7286	0.7052	0.6189	**0.7291**†
	DistilBERT Balanced	Original	0.4761	**0.9510**	**0.7558**	**0.7494**	0.7210	0.6360	0.7826
		Average	0.5057	0.9458	0.7364	0.7383	0.7190	0.6526	0.8054
		Rocchio	**0.5249**†	0.9359	0.7364	0.7386	**0.7231**	**0.6684**	**0.8352**†
	SBERT	Original	0.4097	**0.9767**	**0.8372**	**0.7642**	0.6930	0.5985	0.7201
		Average	0.4565	0.9413	0.7403	0.7326	**0.7001**	**0.6149**	0.7357
		Rocchio	**0.4578**	0.9355	0.7558	0.7448	0.6952	**0.6149**	**0.7405**

Table 5.7: Results of Vector-based PRF for the task of retrieval on TREC DL 2020, using dense retrievers more effective than ANCE and RepBERT. Consistent with the previous table, we apply the fixed hyperparameters optimized for the ANCE baseline ($\alpha = 0.4$ and $\beta = 0.6$) to test robustness across different architectures. We use a fixed PRF depth of 3 for Average and 5 for Rocchio. The best results for each model are marked in **Bold**. Significant improvements are marked with †.

	Model	Method	MAP	RR	nDCG@1	nDCG@3	nDCG@10	nDCG@100	R@1000
TREC DL 2020	TCT-ColBERT V1	Original	0.4290	0.8183	0.7500	0.7245	0.6678	0.5826	0.8181
		Average	**0.4725**	0.8220	0.7346	0.7253	**0.6957**	**0.6101**	**0.8667**
		Rocchio	0.4625	**0.8392**	**0.7840**†	**0.7410**	0.6945	0.6056	0.8576
	TCT-ColBERT V2 HN+	Original	0.4754	**0.8392**	**0.7932**	0.7199	**0.6882**	0.6206	0.8429
		Average	0.4811	0.8212	0.7870	**0.7386**	0.6836	0.6228	**0.8579**
		Rocchio	**0.4860**†	0.8154	0.7685	0.7273	0.6804	**0.6254**	0.8518
	DistilBERT KD	Original	0.4159	**0.8215**	**0.7284**	**0.7113**	**0.6447**	0.5728	0.7953
		Average	**0.4214**	0.7715	0.7130	0.6911	0.6316	0.5755	0.8403
		Rocchio	0.4145	0.7703	0.7037	0.6823	0.6289	**0.5760**	**0.8433**†
	DistilBERT Balanced	Original	0.4698	0.8350	0.7593	0.7426	0.6854	0.6346	0.8727
		Average	**0.4887**	0.8380	0.7809	0.7510	**0.7086**	0.6449	**0.9030**
		Rocchio	0.4879	**0.8641**†	**0.8056**†	**0.7564**	0.7083	**0.6470**	0.8926
	SBERT	Original	0.4124	**0.7995**	**0.7346**	0.6870	0.6344	0.5734	0.7937
		Average	0.4258	0.7619	0.6728	0.6723	0.6412	0.5781	0.8169
		Rocchio	**0.4342**†	0.7941	0.7160	**0.7032**	**0.6559**†	**0.5851**	**0.8226**

Table 5.8: Query latency of the investigated methods on TREC DL 2019: the lower latency, the better (faster).

	Models	Latency (ms/q)
Baselines	BM25 (Anserini Default)	81
	BM25 + RM3 (Anserini Default, $k = 5$)	140
	RepBERT(R)	93
	ANCE(A)	94
	RepBERT+BERT(R+B)	3,324
	ANCE+BERT(A+B)	3,327
BERT Reranker	BM25 + BERT(BB)	3,246
	BM25 + BERT Large	9,209
Vector-based PRF Retriever	R+PRF-R-Average ($k = 5$)	163
	R+PRF-R-Rocchio ($k = 5, \alpha = 0.4, \beta = 0.6$)	163
	A+PRF-A-Average ($k = 5$)	173
	A+PRF-A-Rocchio ($k = 5, \alpha = 0.4, \beta = 0.6$)	174
Vector-based PRF Reranker	BB+PRF-R-Average ($k = 5$)	3,411
	BB+PRF-R-Rocchio ($k = 5, \alpha = 0.4, \beta = 0.6$)	3,414
	BB+PRF-A-Average ($k = 5$)	3,409
	BB+PRF-A-Rocchio ($k = 5, \alpha = 0.4, \beta = 0.6$)	3,414
Text-based PRF Reranker	BB+PRF($k = 5$)-CT	6,889
	BB+PRF($k = 5$)-CA	17,266
	BB+PRF($k = 5$)-SW	22,314
Vector-based PRF with BERT Reranker	R+PRF(Average)+B ($k = 5$)	3,395
	R+PRF(Rocchio)+B ($k = 5, \alpha = 0.4, \beta = 0.6$)	3,397
	A+PRF(Average)+B ($k = 5$)	3,419
	A+PRF(Rocchio)+B ($k = 5, \alpha = 0.4, \beta = 0.6$)	3,421

variants primarily improve shallow metrics compared to sparse baselines, but struggle to consistently outperform the strong dense retriever baselines.

Applying a BERT reranker on top of Vector-based PRF (i.e., R+PRF+B and A+PRF+B) yields gains in MAP, with mixed results across other metrics, many of which are not statistically significant.

Compared to Vector-based PRF, Text-based PRF yields more marginal and dataset-dependent gains, with notable failures on datasets like TREC CAsT and DL HARD. In contrast, Vector-based PRF achieves broader utility across varying query types, demonstrating greater robustness to passage length variations and retrieval contexts, even if it does not universally improve every metric.

5.4.5 Impact of PRF on the Efficiency with Neural Models

RQ1.3.5 examines the efficiency implications of integrating PRF with neural retrieval models. For a comprehensive comparison, we include both the Text-based PRF methods from Chapter 3 and the Vector-based PRF approaches discussed in this chapter.

To address this research question, we evaluate the efficiency of the PRF methods alongside their corresponding baseline models. Low query latency is a critical requirement for search applications.

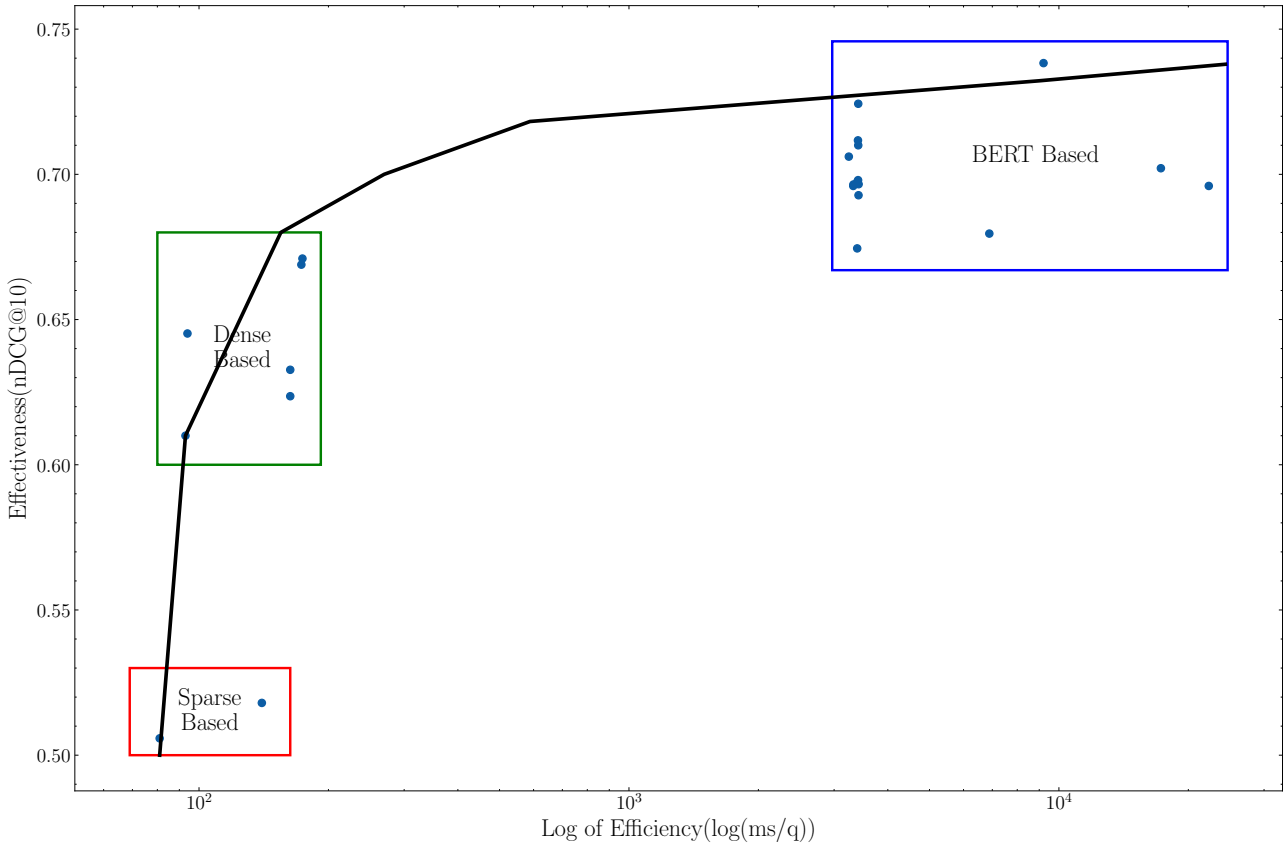


Figure 5.47: Trade-off between effectiveness and efficiency for all methods in our experiments. Effectiveness is measured using nDCG@10, and efficiency is measured using $\log(ms/q)$. The sparse baselines (BM25 and BM25+RM3) cluster on the left bottom corner (red box). Dense based approaches, including ANCE, RepBERT, and our Vector-based PRF approaches cluster on the center left (green box). All rerankers, i.e., BM25+BERT(base/large), Text-based PRF, BM25+BERT+Vector-based PRF, Vector-based PRF+BERT reranker, cluster on the top right side (blue box); these methods present the worst efficiency compared to others. The black line shows the trade-off trend between effectiveness and efficiency.

The latency measurements for all evaluated methods are summarized in Table 5.8 and visualized in Figure 5.47[1].

The dense retrievers studied (RepBERT and ANCE) exhibit query latencies comparable to BM25, with BM25 being 10ms faster. Applying Vector-based PRF (R+PRF-R, A+PRF-A) introduces latency similar to BM25+RM3, which remains 23 – 34ms faster. These latency levels are compatible with search requirements.

In contrast, Vector-based PRF applied to BM25+BERT (BB+PRF-R, BB+PRF-A) results in high latency, similar to BM25+BERT alone. The two-stage BM25+BERT-Large is the slowest approach (up to 100× slower than others) except for Text-based PRF models (e.g., CA, SW), which generate multiple queries per original input, further increasing latency beyond even BERT-Large reranking.

Applying BERT reranking on top of Vector-based PRF (R+PRF+B, A+PRF+B) introduces additional inference time and results in significantly lower efficiency compared to Vector-based PRF alone, making it unsuitable for real-time use.

[1]A fully annotated version of this figure, including model names, is available online at: `https://github.com/ielab/Neural-Relevance-Feedback-Public/blob/master/figures/trade-off-with-label.pdf`

We also examined the impact of query length on latency. For BM25 and BM25+RM3, longer queries lead to increased latency due to additional posting list traversals. In contrast, RepBERT, ANCE, and Vector-based PRF maintain stable latency since all queries are encoded into fixed-length vectors, independent of their textual length. For Text-based PRF, inputs are truncated and padded to 512 tokens, resulting in consistent per-query latency. However, latency scales with PRF depth, as each additional feedback passage requires a separate BERT inference. For instance, with depth 5, methods like CA generate 5 extra queries, resulting in 6 total BERT inferences per original query (see Chapter 3, Section 3.3.5 and Table 5.8), leading to significantly higher latency.

5.4.6 Trade-Offs When Integrating Text-Based and Vector-Based PRF into Neural Models

This section addresses **RQ1.4**, examining the trade-offs of integrating Text-based (From Chapter 3) and Vector-based PRF (This Chapter) into neural models. The two main challenges are computational cost and input length constraints, corresponding broadly to efficiency and effectiveness. Prior work has largely focused on effectiveness while mitigating efficiency costs by applying PRF as a second-stage reranking step over a limited candidate set. For example, Zheng et al. [244] and Wang et al. [206] employed BERT-based reranking after initial retrieval, while others constrained the reranking stage to the top 500 – 1000 documents [94, 233]. However, such approaches often rely on a weak first-stage retriever (e.g., BM25), which can bottleneck overall performance regardless of downstream enhancements.

To evaluate the applicability, effectiveness, and efficiency of Text-based, Vector-based, and hybrid PRF approaches, we compared them across two tasks: retrieval (for Vector-based PRF) and reranking (for Text-based and hybrid PRF). While Text-based and hybrid methods address input length limitations, they offer only marginal effectiveness gains and suffer from high latency. For instance, BERT reranking 1,000 passages takes over 9,200 ms, making these approaches impractical for real-time search (Figure 5.47). In contrast, our Vector-based PRF method processes a query in just 163 – 174 ms and significantly improves retrieval effectiveness over ANCE and RepBERT, and shallow reranking metrics compared to BERT.

We also evaluated the impact of applying a BERT reranker on top of Vector-based PRF. While this setup improves effectiveness across all metrics, significant gains are consistently observed only for MAP, and for nDCG@10 on WebAP with R+PRF+B. Notably, R@1000 improves across several datasets, but these gains stem from Vector-based PRF alone rather than the reranking step.

The BERT reranker used is an off-the-shelf model fine-tuned on MS MARCO, where queries are much shorter than the longer, PRF-augmented queries used in our experiments. This mismatch between training and test data likely reduces its effectiveness on Text-based PRF inputs. Although some improvements are observed, Text-based PRF often underperforms compared to BM25+BERT. Future work could explore fine-tuning BERT on long PRF queries to mitigate this mismatch.

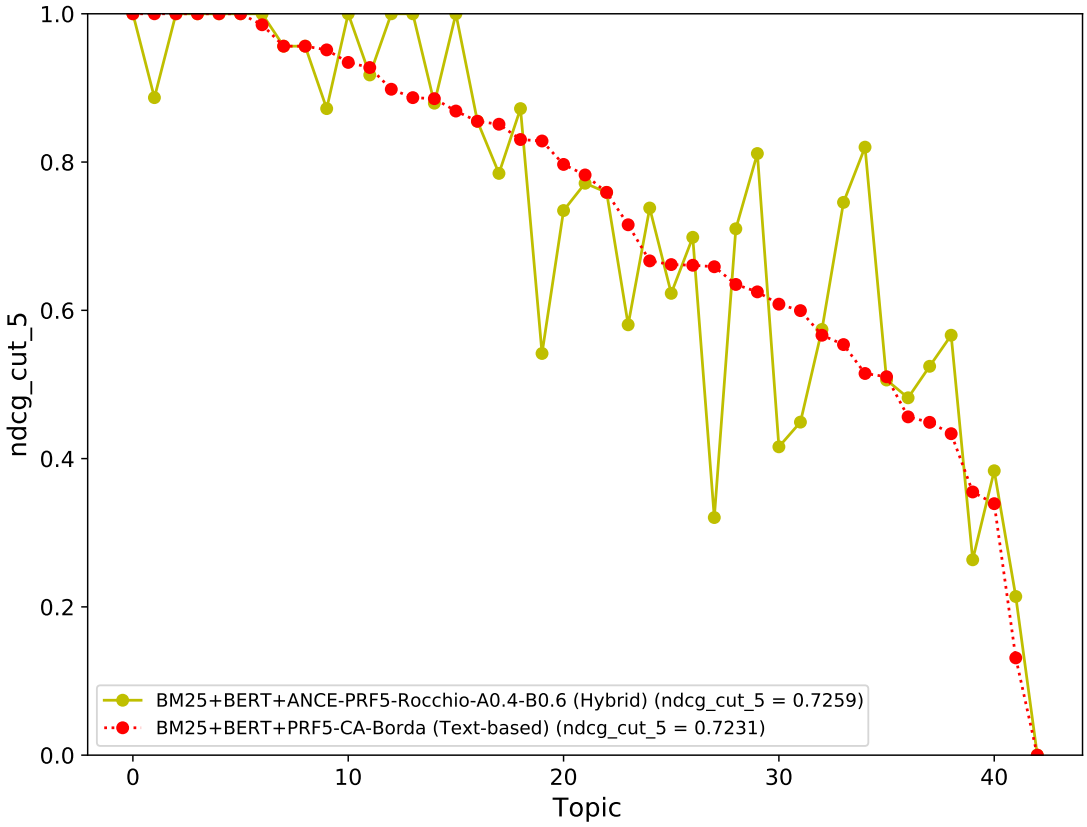


Figure 5.48: Query-by-query comparative analysis between Text-based PRF and Hybrid PRF approaches.

In contrast, Vector-based PRF avoids this issue entirely, as its query vectors remain fixed in length regardless of PRF depth. Moreover, unlike Text-based PRF, it remains efficient and feasible for real-time applications, even without additional fine-tuning.

As complementary experiments, we also evaluated retrieval and reranking on the MS MARCO dev set. While detailed results are omitted for brevity, they are available in the online appendix[2]. We provide a summary of key observations as follows: on the MS MARCO dev set, our Vector-based PRF methods showed no improvement over non-PRF baselines for both retrieval and reranking, unlike the consistent gains observed on TREC DL datasets. We attribute this to MS MARCO's sparse relevance annotations, in contrast to the more complete annotations in TREC DL. While some neural PRF methods like ANCE-PRF [234] (Chapter 4) report improvements on MS MARCO, these models are trained to bring query representations closer to a single "gold" relevant passage. In contrast, our non-learned Vector-based PRF modifies the query based on the top-k retrieved passages, without optimizing toward any specific target passage. As such, these methods are less effective in settings like MS MARCO that resemble known-item retrieval, but remain well-suited for tasks involving multiple relevant passages, as showcased by their strong performance on TREC DL.

[2] https://github.com/ielab/Neural-Relevance-Feedback-Public/blob/master/Vector_Based_PRF/README.md

A more direct comparison is possible between Text-based and hybrid PRF, analyzed in Figure 5.48. However, results are inconclusive, as Text-based PRF shows no clear improvement and hybrid PRF only marginal gains. We acknowledge that PRF can lead to topic or query drift, a well-known issue explored in prior work [61, 116, 219, 226, 251]. A rigorous analysis of query drift in Vector-based PRF would require systematic testing under varying levels of noise in the PRF signal, we will further investigate in Chapter 7 and Chapter 12.

Nonetheless, future work could improve Vector-based PRF by filtering out weakly related feedback passages. In our current setup, all top-k passages are treated as equally relevant, though many may only be loosely related. Applying a filtering mechanism to select highly relevant passages could mitigate drift and enhance PRF effectiveness.

5.5 Summary

This chapter investigated the design and evaluation of Vector-based PRF methods integrated with dense retrieval models. Unlike Text-based PRF, which is restricted to reranking and suffers from efficiency bottlenecks, Vector-based PRF offers a lightweight alternative applicable to both retrieval and reranking that scales well to real-time applications and generalize well to other dense retrievers, while maintaining competitive effectiveness.

We addressed six sub research questions to systematically evaluate the proposed methods.

RQ1.3.1 explored the impact of PRF depth: Vector-based PRF achieved consistent gains with moderate feedback depth (3–5), while deeper feedback (e.g., 10) often led to diminished returns or query drift.

RQ1.3.2 examined the role of the dense encoder used in query and passage representation. ANCE-based PRF (A+PRF-A) consistently outperformed RepBERT-based PRF (R+PRF-R) across most datasets and metrics, particularly in shallow ranking performance.

RQ1.3.3 analyzed vector fusion strategies for aggregating feedback. Rocchio-style methods, particularly $\mathcal{RC}_{\beta}$ and $\mathcal{RC}_{\alpha,\beta}$, are the most effective, offering robust gains across retrieval and reranking tasks.

RQ1.3.4 assessed overall effectiveness, showing that Vector-based PRF improves over dense baselines (ANCE, RepBERT) and sparse ones (BM25, BM25+RM3), and performs competitively when combined with a BERT reranker, especially in MAP, while remaining far more efficient than Text-based PRF.

RQ1.3.5 evaluated efficiency: our Vector-based PRF approach introduced minimal latency compared to BM25+RM3 and was orders of magnitude faster than Text-based methods, making it suitable for low-latency environments.

RQ1.4 examined the trade-offs in integrating PRF into neural models. Our analysis revealed that Text-based and hybrid PRF approaches offer only marginal effectiveness gains at high computational cost, making them impractical for real-time systems. In contrast, Vector-based PRF offers a favorable

balance: it improves effectiveness without violating latency constraints, even when reranking is layered on top.

Compared to ANCE-PRF, our Vector-based PRF method offers several key advantages: it requires no end-to-end fine-tuning, imposes no constraints on PRF depth, and can be applied to any dense retrievers in a plug-and-play manner, making it more flexible and easier to integrate into various retrieval pipelines. ANCE-PRF, by contrast, trains the query encoder to align the PRF query with a single "gold" passage, this is an approach well-suited to benchmarks like MS MARCO that emphasize known-item retrieval. However, Vector-based PRF modifies the query vector directly in embedding space without training, which is more effective in settings involving multiple relevant passages, such as those found in TREC DL.

In summary, Vector-based PRF effectively addresses key limitations identified in previous chapters, namely the inefficiency of Text-based PRF and the inflexibility of ANCE-PRF, offering a scalable and generalizable solution for enhancing dense retrievers mainly for retrieval, but also for reranking stages.

We also integrated our Vector-based PRF methods with Pyserini [112] for the ease of reproducibility[3].

Prior work by Wang et al. [211] demonstrated that integrating BM25 with BERT-based dense retrievers can lead to notable effectiveness gains. While Vector-based PRF achieves strong performance with low latency, the question of how to optimally incorporate sparse retrieval signals (e.g., BM25) into this pipeline remains open. The next chapter builds on this motivation by systematically exploring where sparse-dense interpolation is most effective within the PRF process: before, after, or at both stages, towards a more unified and robust retrieval framework.

[3] `https://github.com/castorini/pyserini/blob/master/docs/experiments-vector-prf.md`

Chapter 6

Interpolation Strategies for Integrating Sparse and Dense Signals with Pseudo-Relevance Feedback

While dense retrievers have demonstrated strong performance in ranking highly relevant results, their effectiveness often declines when dealing with low-relevance passages [51, 145, 200, 211]. This has a direct impact on the quality of the feedback signals within the PRF pipeline [99]. In contrast, traditional unsupervised sparse retrievers such as BM25, despite their limitations in precision, continue to offer high recall and are widely used in initial retrieval stages due to their simplicity and efficiency. These two families of retrieval models, sparse and dense, are known to capture different relevance signals [113], and recent work [111, 115, 211] has shown that combining them through interpolation can yield more effective hybrid systems.

This chapter investigates whether and how interpolation between sparse and dense retrievers can be beneficial in the context of PRF, particularly when applied to VPRF methods we proposed in Chapter 5. VPRF modifies the query embedding using feedback signals from initially retrieved dense representations and conducts a second round of retrieval based on this refined query vector. To this end, we explore the impact of interpolation between dense and sparse retrievers at different stages of the VPRF pipeline. Specifically, we consider three settings: (1) interpolation applied before PRF (Pre-PRF), influencing the feedback input; (2) interpolation applied after PRF (Post-PRF), affecting the final results; and (3) interpolation applied both before and after (Both-PRF), integrating sparse signals throughout the process.

This chapter addresses two research questions:

RQ1.5: Does dense-sparse interpolation help? When?

RQ1.6: Does the underlying sparse retriever matter?

Through extensive experiments, we observe that interpolation can substantially improve the performance of VPRF, particularly when applied after the feedback stage or at both stages of the retrieval pipeline. Furthermore, learned sparse retrievers (e.g., uniCOIL [111]) consistently outperform tradi-

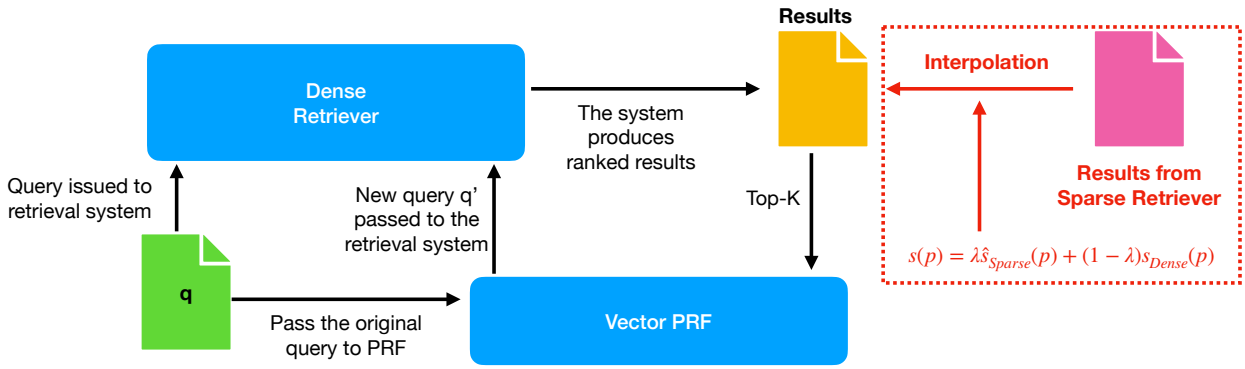


Figure 6.1: Overview of interpolation strategy in our experiments.

tional sparse models (e.g., BM25) when combined with dense retrievers, highlighting the importance of interpolation strategy, its placement within the PRF pipeline, and the choice of sparse retriever in hybrid feedback frameworks.

6.1 Interpolation for Sparse and Dense Signals With PRF

In this section, we introduce our method for interpolating sparse and dense retrievers within the context of PRF, more specifically with vector-based PRF we introduced in Chapter 5. Our goal is to assess how the integration of sparse retrieval signals, either before or after the PRF stage, affects the overall retrieval effectiveness. To combine the outputs of sparse and dense retrievers, we adopt the linear interpolation strategy proposed by Wang et al. [211], which has been shown to be effective for hybrid retrieval. The interpolation is applied by computing a weighted sum of the relevance scores from the sparse and dense retrievers. This simple yet effective technique enables flexible integration of heterogeneous signals without requiring retraining of the retrieval models.

As described in the original paper [211], the final score of a passage is computed as a linear interpolation between its score from a sparse retriever and its score from a dense retriever. This is modulated by a weighting parameter $\lambda \in [0,1]$, which determines the relative contribution of the sparse retriever to the final score. Formally, the interpolated score is calculated as:

$$s(p) = \lambda \hat{s}_{Sparse}(p) + (1-\lambda) s_{Dense}(p) \tag{6.1}$$

where $\hat{s}_{Sparse}(p)$ and $s_{Dense}(p)$ denote the retrieval scores assigned to passage p by the sparse and dense retrievers, respectively. In our experiments, `Sparse` corresponds to either BM25 or uniCOIL [111], and `Dense` refers to any dense retrieval model under consideration (e.g., ANCE [223] or TCT-ColBERT [115]).

This interpolation strategy is applied uniformly across all passages retrieved in a given round, as shown in Figure 6.1. It offers a straightforward yet effective means of augmenting dense retrievers with complementary signals from sparse models, particularly useful for identifying passages that may only weakly match the query semantics but contain important lexical cues. By tuning λ, we can adjust

the balance between semantic and lexical matching, allowing the hybrid model to better handle diverse query characteristics and mitigate the recall limitations inherent to dense-only retrieval.

As mentioned, for PRF, we used the Rocchio Vector-PRF approach from Chapter 5 [104], which consists of two sequential retrieval stages [104]. The first round of retrieval is to generate a candidate set of passages from which the top-k are selected as pseudo relevant feedback signals to refine the original query representation. A second round of retrieval is then performed using this updated query vector. Given this two-stage structure, interpolation between sparse and dense retriever outputs can be integrated at either stage. In this chapter, the interpolation with PRF are categorized into three different types; we discuss each type with detail in the following subsections.

6.1.1 Pre-PRF Interpolation

In the **Pre-PRF interpolation** setting, we apply interpolation during the first retrieval stage. Specifically, we linearly combine the scores from a sparse retriever (BM25 or uniCOIL) and a dense retriever to produce a single ranked list. This hybrid ranking is then used to select the top-k passages for pseudo-relevance feedback. The selected passages are subsequently encoded to form a feedback representation that is used to update the original query embedding, following the vector-based PRF formulation. The modified query is then passed through the second round of dense retrieval without any additional interpolation applied at that stage.

This design allows the feedback signal to benefit directly from the complementary strengths of sparse and dense retrieval. Since the interpolated ranking used for feedback selection differs from the original dense-only ranking, the PRF process may be influenced in both content and quality of the selected feedback passages. Consequently, Pre-PRF interpolation not only alters the candidate set used for feedback but also indirectly affects the final retrieval outcome by modifying the query representation used in the second stage.

6.1.2 Post-PRF Interpolation

In addition to applying interpolation before the PRF stage, we also investigate interpolation **after** PRF has been applied, that is, following the second round of retrieval using the PRF-enhanced query representations. In this **Post-PRF interpolation** setting, we retain the standard vector-based PRF pipeline for the first and second retrieval rounds: the original query is used to retrieve initial passages via a dense retriever, from which the top-k passages are selected to construct a feedback representation. This representation is then used to update the query vector, which is subsequently used to perform the second round of dense retrieval.

Rather than altering the feedback stage, interpolation is applied after the final retrieval results have been produced. Specifically, we linearly combine the scores from the dense retriever (obtained using the PRF-updated query) with those from a sparse retriever for the same query. This produces a final reranked list based on the fused scores.

This approach preserves the original feedback mechanism and its influence on query expansion, while allowing sparse retrieval signals to contribute only at the final ranking stage. By doing so, Post-PRF interpolation serves as a late fusion strategy, enabling sparse models to complement dense retrieval results without affecting the feedback selection process or query representation. This separation of concerns helps isolate the effect of interpolation on the final retrieval output, independent of feedback quality.

6.1.3 Both-PRF interpolation

The **Both-PRF interpolation** strategy combines the benefits of both Pre-PRF and Post-PRF approaches by applying interpolation at both stages of the PRF pipeline. This method aims to fully leverage the complementary strengths of sparse and dense retrievers in both the feedback construction and final result ranking phases.

The process begins with an interpolation between the scores from the sparse retriever and those from the dense retriever during the first round of retrieval, same as Pre-PRF interpolation. Then this interpolated ranking is used to select the top-k passages for pseudo-relevance feedback. These passages then serve as input to the vector-based PRF mechanism, which modifies the original query representation by aggregating the dense embeddings of the feedback passages. The resulting query vector is used to perform the second round of dense retrieval, generating an updated list of candidate passages.

After the second round of retrieval, interpolation is applied again between the scores from the dense retriever (using the PRF-updated query) and those from the sparse retriever (for the original query). The final ranked list is generated based on this second interpolation, the same as Post-PRF interpolation.

By incorporating sparse signals both before and after the feedback loop, Both-PRF allows sparse retrievers to influence not only the selection of feedback candidates but also the final passage ranking. Empirically, this approach has been shown to consistently achieve high retrieval effectiveness across multiple dense retrievers and datasets, often outperforming single-stage interpolation strategies. However, this comes at the cost of additional computational overhead due to two rounds of interpolation and the need to maintain sparse retrieval scores throughout the pipeline. Nonetheless, the method provides a comprehensive hybridisation of sparse and dense signals in neural feedback-based retrieval.

6.2 Empirical Evaluation

To investigate the interpolation of sparse retrievers with dense retriever PRF approaches, we devise a number of empirical experiments aimed at investigating: 1) the impact of interpolation on different dense retriever PRF approaches; 2) the impact of interpolating sparse retrievers before/after/both the PRF; 3) the impact of interpolating on different sparse retrievers, unsupervised (BOWs) or learned.

Therefore, we propose the following sub research questions under the main research questions to guide our empirical evaluation:

RQ1.5.1: Does dense-sparse interpolation help in the context of PRF?

RQ1.5.2: When is it better to do interpolation? Pre-PRF, Post-PRF or Both-PRF?

RQ1.6.1: Does different sparse retrievers matter when interpolating?

RQ1.6.2: Which sparse retriever is more effective, unsupervised (BOWs) or learned?

6.2.1 Datasets

For all of our experiments in this Chapter, we use the TREC Deep Learning Track 2019 [27] (DL19) and 2020 [26] (DL20). DL19 contains 43 judged queries, while DL20 contains 54 judged queries. The relevance judgement levels for both datasets range from 0 (not relevant) to 3 (highly relevant).

We treat passages with relevance label 1 as not relevant when we compute the binary relevance metrics (i.e., MAP, Recall). The passage collection in our experiments is the MS MARCO Passage Ranking Dataset [151], which is a benchmark English dataset for ad-hoc retrieval that contains ≈8.8 million passages. The average judgements per query for DL19 and DL20 are 215.3 and 210.9, whereas the MS MARCO Passage Ranking Dataset only has ≈1 judgement per query.

6.2.2 Baselines

We include:

- `ANCE`: First stage dense retriever [223]. We use the model implemented in Pyserini[1] [112] for inference;
- `Vector-based PRF (VPRF)`: A simple Rocchio PRF approach based on dense retrievers from Chapter 5. We use the model implemented in Pyserini[2];
- `TCT ColBERT V2 HN+ (TCTv2)`: A BERT-style distilled dense retriever learned from ColBERT with reduced query/passage embedding dimensions [115];
- `TCT ColBERT V2 HN+ VPRF (TCTv2+VPRF)`: The application of the Rocchio VPRF from Chapter 5 on top of TCT ColBERT V2 HN+ dense retriever. This model is also made available in Pyserini[2] [112];
- `DistilBERT KD TASB (DBB)`: A DistilBERT-style dense retriever with balanced topic aware sampling training strategy [73]. We use the model implemented in Pyserini[3] by the original authors;

[1] `https://github.com/castorini/pyserini/blob/master/docs/experiments-ance.md`
[2] `https://github.com/castorini/pyserini/blob/master/docs/experiments-vector-prf.md`
[3] `https://github.com/castorini/pyserini/blob/master/docs/experiments-distilbert_tasb.md`

- `DistilBERT KD TASB + VPRF (DBB+VPRF)`: The application of the Rocchio VPRF from Chapter 5 on top of DistilBERT KD TASB dense retriever. This model is implemented and available to use in Pyserini[2].

In our experiments, we use the parameters $\alpha = 0.4$, $\beta = 0.6$, and PRF depth $k = 3$ for Rocchio VPRF. These values were selected based on the empirical findings in Chapter 5, where this specific combination was identified as the global optimum for the TREC DL 2020 dataset (Table 5.2). Although TREC DL 2019 achieves peak effectiveness at a deeper feedback depth ($k = 10$) as shown in Table 5.1, we standardize on $k = 3$ to use a moderate PRF depth and ensure experimental consistency across datasets and to prioritize the configuration that maximizes performance on the more recent benchmark. At this fixed depth of $k = 3$, $\alpha = 0.4$ and $\beta = 0.6$ proved to be the most robust weighting scheme across different retrievers. In terms of the interpolation parameter λ, we use $\lambda = 0.5$ for all experiments, assigning equal weight to sparse and dense scores. This choice follows the best practices established by Wang et al. [211] for hybrid retrieval, providing a neutral baseline to assess general compatibility without overfitting to specific dataset characteristics. For generating the BM25 runs, we use the implementation provided by Pyserini with default parameters ($k_1 = 0.9, b = 0.4$), and for uniCOIL, we use the pre-built index provided by the toolkit.

6.2.3 Evaluation Metrics

We use the official evaluation metrics from DL19 and DL20: nDCG@10 and Recall@1000. We also report MAP as a complementary metric. Statistical significance is assessed using a two-tailed paired t-test with Bonferroni correction.

6.3 Results

Next, we examine the results of our empirical investigation, guided by the research questions proposed in Section 6.2. The main results are presented in Table 6.1. We deliberately employ the original model checkpoints provided by the authors rather than training new instances. As demonstrated in Table 4.2 of Chapter 4, variations in effectiveness due to stochastic weight initialization during retraining are minor and statistically insignificant. Therefore, using the standard public checkpoints ensures reproducibility and facilitates fair comparison with existing literature. Furthermore, we focus our analysis on the VPRF-Rocchio configuration (as opposed to VPRF-Average) because Chapter 5 identified it as the more effective and flexible variant, capable of better balancing query and feedback signals through its parameterized weighting scheme. This choice ensures that our interpolation experiments build upon the strongest available vector-based feedback baseline.

6.3.1 Dense-Sparse Interpolation in PRF Pipeline

To answer the first two sub research questions:**RQ1.5.1: Does dense-sparse interpolation help in the context of PRF?** and **RQ1.5.2: When is it better to do interpolation? Pre-PRF, Post-PRF, or**

Dataset			DL19			DL20		
Sparse Model	**Dense Model**	**PRF-Interpolation**	**MAP**	**nDCG@10**	**Recall@1000**	**MAP**	**nDCG@10**	**Recall@1000**
	ANCE		0.3710	0.6452	0.7554	0.4076	0.6458	0.7764
	ANCE-VPRF		0.3831	0.6512	0.7611	0.4118	0.6479	0.7800
BM25	ANCE	No-PRF	0.4264	0.6888	**0.8607**	0.4067	0.6264	0.8643
		Pre-PRF	0.3868	0.6620	0.7638$^{\overline{\wedge}}$	**0.4175**	**0.6548**	0.7911$^{\overline{\wedge}}$
		Post-PRF	0.4322$^{\overline{\wedge}}$	0.6885	0.8604*	0.4140	0.6353	**0.8666**$^{\overline{\wedge}\star}$
		Both-PRF	**0.4345**$^{\overline{\wedge}}$	**0.6895**	**0.8607**	0.4100$^{\overline{\wedge}}$	0.6274	0.8662$^{\overline{\wedge}}$
uniCOIL	ANCE	No-PRF	0.4587	0.6908	0.8459	0.4644	0.6984	0.8482
		Pre-PRF	0.3857$^{\overline{\wedge}}$	0.6602$^{\overline{\wedge}}$	0.7584$^{\overline{\wedge}}$	0.4154$^{\overline{\wedge}}$	0.6545$^{\overline{\wedge}}$	0.7892$^{\overline{\wedge}}$
		Post-PRF	0.4617*	0.6910*	**0.8495***	0.4693$^{\overline{\wedge}\star}$	**0.7024***	**0.8500***
		Both-PRF	**0.4622**	**0.6917**	0.8458	**0.4699**$^{\overline{\wedge}}$	0.7012	**0.8500**$^{\overline{\wedge}}$
	TCTv2		0.4469	0.7204	0.8261	0.4754	0.6882	0.8429
	TCTv2-VPRF		0.4626	0.7219	0.8377	0.4863	0.6952	0.8462
BM25	TCTv2	No-PRF	0.4474	0.7067	0.8753	0.4468	0.6696	0.8872
		Pre-PRF	**0.4698**	**0.7268**	0.8436	**0.4879**$^{\overline{\wedge}}$	**0.6987**	0.8455$^{\overline{\wedge}}$
		Post-PRF	0.4547$^{\overline{\wedge}}$	0.7045	0.8788	0.4511$^{\overline{\wedge}\star}$	0.6683	**0.8918**$^{\overline{\wedge}\star}$
		Both-PRF	0.4574$^{\overline{\wedge}}$	0.7061	**0.8820**$^{\overline{\wedge}}$	0.4490	0.6659	0.8902$^{\overline{\wedge}}$
uniCOIL	TCTv2	No-PRF	0.4771	0.7245	0.8557	0.4895	0.7180	0.8683
		Pre-PRF	0.4659	0.7246	0.8415	0.4897	0.7083	0.8484$^{\overline{\wedge}}$
		Post-PRF	0.4826$^{\overline{\wedge}\star}$	**0.7347**	0.8663$^{\overline{\wedge}}$	**0.4926**	0.7184	0.8718*
		Both-PRF	**0.4831**	0.7268	**0.8606**$^{\overline{\wedge}}$	0.4920$^{\overline{\wedge}}$	**0.7190**	**0.8723**$^{\overline{\wedge}}$
	DBB		0.4590	0.7210	0.8406	0.4698	0.6854	0.8727
	DBB-VPRF		0.4667	0.7285	0.8479	0.4804	**0.7027**	0.8767
BM25	DBB	No-PRF	0.4584	0.6993	0.8622	0.4417	0.6491	0.8948
		Pre-PRF	**0.4711**	**0.7319**	0.8526	**0.4812**$^{\overline{\wedge}}$	0.6968$^{\overline{\wedge}}$	0.8755
		Post-PRF	0.4652$^{\overline{\wedge}}$	0.7053	0.8711	0.4509*	0.6522*	0.8974
		Both-PRF	0.4665$^{\overline{\wedge}}$	0.7019	**0.8720**	0.4442	0.6474	**0.8977**
uniCOIL	DBB	No-PRF	0.4779	0.7288	0.8542	0.4859	0.7041	0.8808
		Pre-PRF	0.4726	0.7255	0.8468	0.4815	0.7002	0.8760
		Post-PRF	0.4822	0.7298	0.8658	**0.4915**	**0.7097**	**0.8836**
		Both-PRF	**0.4858**$^{\overline{\wedge}}$	**0.7334**	**0.8669**	0.4880$^{\overline{\wedge}}$	0.7062	0.8834

Table 6.1: The results of all baseline runs and No-PRF, Pre-PRF, Post-PRF and Both-PRF interpolation runs of all dense models with the Rocchio Vector PRF approach. Statistical significance tests are conducted between Pre- and Post-PRF models, significant difference are marked with $\star$. We also tested the statistical significance with Pre-PRF interpolation versus No-PRF interpolation, and Post-PRF interpolation versus No-PRF interpolation, and Both-PRF interpolation versus no-PRF interpolation, significant difference are marked with $\overline{\wedge}$. Best performance among each base sparse model is marked as **Bold**.

Both-PRF?, we conduct a systematic evaluation of three interpolation strategies: Pre-PRF, Post-PRF, and Both-PRF. Each strategy is applied across three dense retrievers: ANCE, TCTv2, and DistillBERT (DBB). Interpolation is used to integrate sparse retriever scores with dense retriever outputs during different stages of the PRF process.

To establish whether interpolation is helpful in the context of PRF (**RQ1.5.1**), we begin by comparing the effectiveness of PRF alone and interpolation alone with that of interpolation combined with PRF, as shown in Table 6.1. Across dense retrievers, we observe that adding interpolation to the PRF pipeline consistently improves performance over using PRF alone. This confirms that interpolation is not only compatible with PRF but also beneficial in improving retrieval effectiveness. The gain is especially clear when examining metrics such as MAP and nDCG@10, where combined strategies outperform both individual components. This supports the idea that PRF and interpolation

are complementary: PRF enhances query representations using feedback from top-ranked passages, while interpolation enriches the ranking with additional scoring signals during either or both stages.

To answer **RQ1.5.2**, we turn to a detailed comparison of the three interpolation strategies. Pre-PRF interpolation is applied before feedback, affecting the selection of feedback passages; Post-PRF interpolation is applied after feedback, adjusting the final ranking; and Both-PRF performs interpolation at both stages. We observe that Both-PRF most frequently achieves the highest effectiveness across dense retrievers and datasets. This suggests that introducing sparse signals at both the feedback construction and final ranking stages results in more robust retrieval performance.

Post-PRF interpolation, while computationally less demanding than Both-PRF, also yields substantial improvements, particularly on deeper ranking metrics such as Recall@1000. This configuration retains the original feedback process and leverages interpolation solely at the output stage, offering a clean late-fusion mechanism. On the other hand, Pre-PRF interpolation generally performs less effectively than the other two strategies. Its impact is limited to modifying the feedback selection without contributing to the final ranking. As a result, it yields improvements in early precision but lacks the holistic integration offered by Post-PRF and Both-PRF.

Further breakdown by dense model and dataset shows that Both-PRF consistently outperforms the others in overall effectiveness, especially in scenarios where both stages of the PRF pipeline benefit from complementary signals. However, the difference between Post-PRF and Both-PRF is often small and not always statistically significant. In contrast, the gap between Pre-PRF and the other strategies is more pronounced, with Pre-PRF underperforming on both precision- and recall-oriented metrics across most configurations.

In comparing early precision versus deep recall patterns across strategies, we find that Pre-PRF interpolation tends to enhance precision-focused metrics such as nDCG@10 and MAP, aligning with its influence on feedback candidate quality. Post-PRF interpolation, by contrast, demonstrates stronger gains on recall-oriented metrics, as it directly affects the final ranking through the addition of sparse signals. Both-PRF, by design, balances both aspects, yielding improvements across the full range of metrics.

To summarise, the results confirm that interpolation is beneficial in the context of PRF (**RQ1.5.1**), and that the timing of interpolation significantly influences its effectiveness (**RQ1.5.2**). Among the three configurations, Both-PRF consistently performs best across models and datasets, while Post-PRF offers a strong and computationally efficient alternative. Pre-PRF, although capable of marginal early precision gains, is generally the least effective. These findings validate the utility of combining sparse and dense signals within the PRF pipeline and underscore the importance of interpolation placement in achieving optimal retrieval performance.

6.3.2 Different Sparse Retrievers For Interpolation

To address the sub-research questions **RQ1.6.1: Does different sparse retrievers matter when interpolating?** and **RQ1.6.2: Which sparse retriever is more effective, unsupervised (BOWs) or**

learned?, we conduct a detailed comparative analysis using two representative sparse retrievers: BM25 and uniCOIL [111]. BM25 is a classical unsupervised bag-of-words (BOWs) model that relies on exact term matching and term frequency-based scoring. In contrast, uniCOIL is a learned sparse retriever that uses a BERT-based encoder trained with contrastive learning to assign contextual impact scores to individual tokens. To expand the coverage of relevant terms, uniCOIL applies docTquery-T5 [158] to augment all corpus passages prior to encoding.

Our analysis investigates how these two sparse retrievers influence performance when used for interpolation in both PRF and No-PRF conditions, the results are in Table 6.1.

Across all dense retrievers and datasets, and under No-PRF, Post-PRF, and Both-PRF conditions, we observe that interpolation using uniCOIL consistently yields higher effectiveness than interpolation with BM25 on top-ranked metrics such as MAP and nDCG@10. This pattern holds for every dense model and dataset. However, BM25 achieves higher Recall@1000 compared to uniCOIL, indicating that it retrieves a broader range of documents, but with less precise ranking. This trade-off is expected, as BM25 offers wide recall due to lexical coverage, while uniCOIL focuses more on matching meaningful contextual terms, benefiting early precision. These findings confirm that sparse retriever choice significantly impacts the quality of interpolated rankings, addressing **RQ1.6.1**.

Interestingly, this performance gap remains consistent even when PRF is incorporated through Post-PRF and Both-PRF interpolation. In these settings, uniCOIL-based interpolation improves both precision-oriented and ranking-sensitive metrics, suggesting strong compatibility with PRF-enhanced query embeddings. Conversely, BM25 tends to yield lower MAP and nDCG@10, while retaining marginally better Recall@1000. This demonstrates that learned sparse models like uniCOIL are more effective than traditional BOWs-based models (supporting **RQ1.6.2**) when used in hybrid configurations that combine sparse and dense signals, even when PRF is applied.

However, the comparative performance of BM25 and uniCOIL diverges in the Pre-PRF interpolation setting. On DL19, BM25 consistently outperforms uniCOIL across nearly all dense retrievers, with the only exception of MAP when using DBB. In this case, the use of BM25 during the first retrieval stage appears to result in higher-quality feedback candidates, likely due to BM25's exhaustive lexical matching, which prioritizes highly relevant terms that influence feedback positively. On DL20, the performance pattern is mixed: when paired with TCTv2 or DBB, uniCOIL outperforms BM25, while ANCE paired with BM25 yields higher scores. These inconsistencies suggest that the effectiveness of a sparse retriever under Pre-PRF interpolation in first stage retrieval is both model-dependent and dataset-sensitive.

These observations reveal that Pre-PRF interpolation is particularly sensitive to the initial ranking shift caused by sparse-dense score blending. Since the feedback candidates are drawn from the interpolated top-k passages, the choice of sparse retriever can directly influence the feedback signal. We hypothesize that learned sparse retrievers like uniCOIL may introduce greater variation in passage rankings due to their sensitivity to semantic context, whereas BM25 may lead to more stable and conservative selection due to its surface-level token reliance. This helps explain the inconsistent

Table 6.2: The results of all models with no PRF but interpolation only. In this table we show the nCG@3 and Jaccard Similarity (JS) scores between the model itself and the model after interpolation. The + indicates the interpolation operation.

Dataset	**DL19**		**DL20**	
Model	**nCG@3**	**JS**	**nCG@3**	**JS**
ANCE	0.6173	0.3721	0.6348	0.3963
BM25+ANCE	0.6458		0.5856	
TCTV2	0.6611	0.3860	0.6183	0.3926
BM25+TCTV2	0.6557		0.6195	
DBB	0.6739	0.3512	0.6195	0.3667
BM25+DBB	0.6557		0.6004	

advantage of uniCOIL in Pre-PRF, in contrast to its dominant performance under Post-PRF and Both-PRF.

To further investigate how interpolation alters feedback candidates in Pre-PRF, we analyze top-k passage differences before and after interpolation in the first round of retrieval. Specifically, we measure the change in top-ranked relevance using nDCG@3 and compute Jaccard Similarity of the top-10 passages. Results are presented in Table 6.2. We find that, except for ANCE-based models on DL19 and TCTv2-based models on DL20, interpolation generally reduces nDCG@3 scores, suggesting that the top-3 passages after interpolation are less relevant than those from the original dense-only retrieval. This indicates that the feedback selection stage is sensitive to even modest shifts in ranking caused by interpolation, particularly with learned sparse retrievers. However, the decrease in early relevance may not reflect a rise in irrelevant passages but rather an increase in unjudged content. This interpretation is supported by the Jaccard Similarity results, which show substantial divergence between original and interpolated top-10 rankings, demonstrating that interpolation substantially reshapes the candidate set used for feedback.

Taken together, these results confirm that the effectiveness of interpolation is significantly influenced by the sparse retriever used (**RQ1.6.1**). Learned sparse retrievers like uniCOIL consistently lead to better precision and ranking metrics when integrated into dense retriever pipelines, both with and without PRF (**RQ1.6.2**). However, their benefits are less stable in the Pre-PRF setting, where ranking perturbations affect the feedback phase. In contrast, BM25, while generally less effective in terms of MAP and nDCG@10, provides higher recall and more stable early rankings in Pre-PRF. Finally, the choice between BM25 and uniCOIL should be guided by both the interpolation strategy employed and the target evaluation metric, highlighting the important role of sparse retriever selection in hybrid retrieval systems.

6.4 Summary

Prior work has highlighted the complementary strengths of sparse and dense retrievers, noting that each captures distinct relevance signals within a passage [113, 211]. This observation has motivated research into hybrid retrieval techniques, particularly score interpolation between sparse and dense retrievers as a means of enhancing retrieval effectiveness [211]. Building upon this foundation, this chapter conducted a systematic investigation of interpolation strategies in the context of PRF, with a specific focus on their integration into the scalable and effective Vector-PRF method [104].

Our study explored three interpolation strategies: Pre-PRF, Post-PRF, and Both-PRF, across combinations of three dense retrievers (ANCE, TCTv2, DistillBERT) and two sparse retrievers (BM25 and uniCOIL). The experimental results demonstrate that interpolation consistently improves retrieval performance across dense retrievers and PRF configurations. In particular, the Both-PRF strategy, which integrates sparse signals both before and after the feedback stage, yields the highest effectiveness across most settings. Post-PRF interpolation also provided substantial improvements while being computationally more efficient. Pre-PRF interpolation was comparatively less effective, showing sensitivity to feedback candidate variations.

Furthermore, our analysis revealed that the choice of sparse retriever significantly influences interpolation performance. While traditional unsupervised models like BM25 offer higher recall, learned sparse retrievers such as uniCOIL achieve better early precision and overall ranking metrics (e.g., MAP, nDCG@10), especially under Post-PRF and Both-PRF conditions. These findings highlight the importance of selecting an appropriate sparse model depending on the interpolation strategy and retrieval objective. Additionally, we observed that Pre-PRF interpolation introduces considerable variation in the top-ranked feedback candidates, therefore impacting the downstream feedback process, which is varified through nDCG@3 and Jaccard Similarity analyses.

While this chapter primarily focused on effectiveness, the efficiency implications of the proposed interpolation strategies are inherently favourable. The pipeline relies on components with established low-latency profiles: the Vector-based PRF framework, which operates at approximately 174ms per query (as detailed in Chapter 5), and highly efficient sparse retrievers like BM25 and uniCOIL, which can achieve latencies as low as $\approx$ 7ms. Since the interpolation mechanism itself involves only a simple linear combination of pre-computed scores, incurring negligible overhead ($<$ 1ms), the total execution time remains well within the bounds of real-time interaction. This represents that interpolation enhances robustness without the prohibitive computational cost.

Although our experiments leveraged both pre-trained and retrained dense retriever checkpoints, it is worth noting that dense models exhibit some degree of non-determinism due to initialization and training stochasticity. However, as previously reported in our earlier study in Chapter 4, the observed variations when retraining dense models are minor and do not result in statistically significant differences.

This chapter concludes Part 1 of the thesis, which has investigated a series of strategies for integrating pseudo-relevance feedback into modern neural retrieval pipelines. These strategies range

from classical text-based methods (Chapter 3), to vector-based feedback mechanisms (Chapter 5), and culminate in the hybrid sparse-dense interpolation techniques explored in this chapter. Furthermore, Chapter 4 also examined the challenges and limitations of integrating PRF into trained dense retrievers. When considered alongside the constraints identified in Chapter 3, these findings highlight the need for a more adaptable and efficient solution, which is an objective addressed in Chapter 5 through the proposed vector-based PRF approach. Collectively, these investigations demonstrate the versatility of PRF across diverse retrieval architectures and signal representations, and lay the foundation for the next part of the thesis, which shifts focus to the reliability and effectiveness of feedback signals themselves.

As we transition into Part 2 of this thesis, the focus shifts from the integration of PRF into neural retrieval pipelines to the development of adaptive, lightweight, and practical feedback mechanisms. A central theme in this part is understanding how the quality of feedback signals, specifically the top-ranked passages used during PRF, affects overall retrieval effectiveness. This question, identified repeatedly throughout Part 1 as a critical limitation of existing approaches, forms the starting point for our investigation. The next chapter addresses this gap by systematically examining the sensitivity of PRF effectiveness to feedback signal quality, using controlled variations in passage relevance to assess their impact across both sparse and dense PRF methods. This exploration establishes the groundwork for building more robust PRF strategies that can adapt to varying retrieval conditions and maintain effectiveness in real-world applications where feedback quality is not guaranteed.

Part 2

Adaptive, Lightweight, and Practical Feedback Mechanism

Introduction to Part 2

As demonstrated in Part 1, PRF is an effective strategy for improving retrieval effectiveness in neural information retrieval pipelines. However, many of the existing PRF methods [94, 116, 206, 214, 214, 218, 233, 234] remain impractical for real-world deployment due to their computational overhead, dependent on model-specific tuning, and limited generalizability across different retrieval pipelines. In particular, neural PRF methods that require the additional model fine-tuning or full re-encoding are often infeasible in latency-sensitive or resource-constrained environments. This motivates the need for more efficient and extensible PRF approaches that can be easily integrated into diverse retrieval systems.

In this part of the thesis, we aim to address the research question:

RQ2: How to make Pseudo-Relevance Feedback models more practical and extensible?

To this end, Chapter 7 begins by investigating the quality and reliability of feedback signals in dense retrieval scenarios [99]. We propose a diagnostic framework that isolates the effects of feedback selection from the query reformulation process, enabling a clearer understanding of the impact of different feedback signal qualities. Our findings reveal that effectiveness gains from PRF are highly sensitive to the characteristics of initial retrieval results, exposing the limitations of static feedback mechanisms when facing noisy or inconsistent signals.

Addressing the need for a more adaptive yet efficient solution, Chapter 8 introduces a transformer-based PRF method (TPRF) [105]. Unlike static approaches, TPRF utilizes a lightweight transformer encoder to learn how to dynamically aggregate signals from feedback passage embeddings. This design not only improves robustness to signal variation but also removes the need for text-level query expansion and repeated use of large pre-trained models. We demonstrate that TPRF can match or exceed the performance of computationally expensive approaches while maintaining the low inference latency required for real-world deployment.

Chapter 9 addresses the challenge of infrastructural standardization. We propose a modular framework within the Pyserini [112] toolkit [101] designed to serve as a generalized blueprint for dense feedback integration. Using Vector-based PRF (VPRF) as a foundational case study, we demonstrate how to decouple feedback logic from the underlying retriever architecture. This standardized interface establishes the necessary engineering groundwork to support the broader family of methods explored in this thesis, from static vector operations to learned architectures like TPRF, thereby bridging the gap between research prototypes and reproducible, production-ready systems.

Together, the contributions in this part establish a practical foundation for implementing adaptive and efficient PRF methods in modern retrieval pipelines. We demonstrate that with the vector-based feedback modeling, lightweight architectures, and modular software support, it is possible to close the gap between research-oriented PRF experiments and deployable systems. These findings also pave the way for the next part of the thesis, where we explore how PRF can be further adapted to emerging LLM-based retrieval paradigms.

Chapter 7

Feedback Signal Quality Affect Pseudo-Relevance Feedback Effectiveness

The effectiveness of PRF methods, as discussed extensively in Part 1, is often constrained by the quality of the underlying feedback signals. This chapter focuses on investigating this limitation by exploring the following research questions:

RQ2.1: How does feedback quality affect Pseudo-Relevance Feedback?

RQ2.2: How robust are Pseudo-Relevance Feedback methods under noisy signals?

These questions emerge from a core assumption adopted by a majority of PRF methods: the top-k passages retrieved in the initial stage are sufficiently relevant to serve as a reliable signal for query reformulation [44, 224].

This assumption is formalized in the classic Rocchio method [179], and is frequently used in both sparse and neural IR settings. In its simplified PRF formulation, by excluding negative feedback for brevity, the updated query vector $\vec{q}'$ is computed as:

$$\vec{q}' = \alpha\vec{q} + \beta\frac{1}{|Rel|}\sum_{d_i \in Rel}\vec{d}_i \tag{7.1}$$

where $\vec{d}_i$ represents each of the top-k feedback document vectors. Similar approaches are adopted across traditional bag-of-words models [9, 91, 125, 177, 179, 237] and modern neural retrievers [102, 104, 234]. Despite their popularity, these approaches often overlook the reliability of the relevance signal: what happens when the top-k documents are predominantly irrelevant? Can PRF techniques maintain robustness, or do they suffer from performance collapse under such noise?

These questions are of both theoretical and practical importance. While prior research on selective query expansion [18, 28, 242] and query performance prediction [7, 64] has sought to identify and mitigate poor feedback, there remains a need to systematically quantify the resilience of modern dense PRF architectures against varying levels of noise. Furthermore, standard evaluations typically rely on a fixed initial retriever (e.g., BM25), effectively testing only a single, static level of signal quality. However, real-world systems differ significantly in retrieval strength, implying a wide spectrum of

signal qualities. This necessitates a more granular analysis of how PRF methods perform across these diverse noise regimes, rather than assuming a fixed baseline quality.

To bridge this gap, we conduct a systematic empirical study of how PRF methods behave under controlled variations in different feedback signal qualities in this chapter. We evaluate our VPRF-Rocchio method (Chapter 5) as well as two recent PRF techniques for dense retrievers under both high-quality and noisy signal conditions. Our findings reveal that:

1. PRF effectiveness is highly sensitive to the quality of the feedback signal, with notable degradations observed when the top-k documents contain a high proportion of non-relevant content;
2. Dense PRF methods that outperform others under clean feedback conditions may suffer more severely under noise, indicating limited robustness;
3. Among the evaluated methods, simpler vector-based PRF approaches (e.g., Rocchio in dense settings) exhibit more stable behavior across signal quality regimes, suggesting greater adaptability.

These results highlight the need to reconsider how PRF methods are evaluated and selected in practice. Robustness to feedback quality should be treated as a critical evaluation criteria, particularly for real-world deployments where retrieval conditions may be suboptimal or noisy.

7.1 Investigating the Impact of Pseudo-Relevance Feedback Signals

The objective of this study is to systematically investigate the impact of feedback signal quality on the effectiveness of PRF methods. To this end, we design a comprehensive experimental setup that enable controlled manipulation of PRF signals. This section describes the methodology used to generate and apply these controlled feedback signals across various initial retrieval methods, therefore allowing a consistent evaluation of PRF robustness under varying signal conditions.

7.1.1 Controlling the Quality of the Feedback Signal

We begin our investigation by constructing feedback signals derived from first-stage retrievers. Specifically, we consider one sparse retriever, BM25 [177], and three representative dense retrievers: ANCE [223], TCTv2-HN+ [115], and DistillBERT-Balanced [73]. These retrievers span different classes of retrieval models, enabling a comprehensive analysis of how feedback signal quality varies across architectures.

For each query, we retrieve the top 1,000 passages using the specified first-stage retriever and remove all unjudged passages based on TREC relevance assessments. To simulate distinct levels of feedback quality, we partition the remaining judged passages into three disjoint categories:

- **Strong Signal:** Passages with relevance labels 2 (highly relevant) or 3 (perfectly relevant).

- **Moderate Signal:** Passages with relevance label 1 (relevant).
- **Weak Signal:** Passages with relevance label 0 (non-relevant).

From these pools, we uniformly sample exactly 12 passages per query for each signal category. Queries containing fewer than 12 passages in any category are excluded to ensure consistency. This threshold was determined empirically as the largest value satisfying a per-query minimum across all signal types, while also being divisible by our target feedback depths ($k = 1$ and $k = 3$). The final dataset statistics are summarized in Table 7.1. Note this filtering removed 7 queries from TREC DL 2019 and 12 from TREC DL 2020 due to insufficient judged passages[1].

7.1.2 Hierarchical Averaging Strategy

To robustly evaluate PRF effectiveness while mitigating the influence of any single feedback passage, we employ a hierarchical averaging strategy for both feedback depths. This procedure is applied identically across Strong, Moderate, and Weak signal conditions:

Feedback Depth $k = 1$: We construct 12 separate PRF queries for each original query by using each of the 12 sampled feedback passages individually. We perform retrieval for each PRF query, compute the evaluation metrics (e.g., MAP, nDCG) for the resulting ranked lists, and average these 12 scores to produce a single performance estimate for that query. For example, in the filtered TREC DL 2019 dataset (36 queries), this results in 432 total retrieval runs (36×12).

Feedback Depth $k = 3$: We randomly partition the 12 sampled passages into four non-overlapping groups of three. Each group serves as the feedback set for a separate PRF run. The final performance for the query is obtained by averaging the metric scores of these four runs.

This rigorous sampling and averaging approach ensures that our reported results reflect the general impact of a specific signal quality level, rather than artifacts of specific passage combinations.

7.2 Empirical Evaluation

To guide our systematic evaluation of the impact of feedback signal quality, we formulate the following research questions:

RQ2.1.1: To what extent does the effectiveness of PRF methods vary when using strong, moderate, and weak feedback signals?

RQ2.1.2: How does the choice of first-stage dense retrievers influence the quality of the feedback signal and the subsequent effectiveness of PRF?

[1]TREC DL 2019 removed queries: ’1124210’, ’443396’, ’855410’, ’1117099’, ’1037798’, ’1121709’, ’131843’; TREC DL 2020 removed queries: ’1116380’, ’405163’, ’42255’, ’1105792’, ’1115210’, ’324585’, ’1131069’, ’673670’, ’336901’, ’768208’, ’1030303’, ’258062’. These queries were excluded due to having fewer than 12 judged passages for at least one relevance label.

RQ2.2.1: Which combinations of PRF methods and retrieval representations (sparse vs. dense) have the most stable performance when feedback signal quality degrades, particularly under weak feedback signals?

RQ2.2.2: How does the degree of performance degradation from strong to weak feedback signals differ across PRF methods using different underlying dense retrievers?

7.2.1 Datasets

Our experiments use the TREC Deep Learning Track Passage Retrieval Task 2019 [27] (TREC DL 2019) and 2020 [26] (TREC DL 2020). The detailed statistics for each dataset are listed in Table 7.1. Reader can refer to previous chapters for detailed description of these two datasets (e.g., Section 4.2.1 in Chapter 4, Section 6.2.1 in Chapter 6)

TREC DL 2019 and 2020 originally have 200 queries each. However, only 43 queries and 54 queries are judged in 2019 and 2020 respectively according to the QRels file, thus we discard all other queries that do not have judgements. In our experiments, some queries are further discarded because they do not have enough PRF feedback passage candidates. After removing the queries that do not meet the requirement, TREC DL 2019 has 36 valid queries, and TREC DL 2020 has 42 valid queries.

7.2.2 Evaluation Metrics

We employ MAP, Reciprocal Rank (RR), nDCG@{1,3,10}, and Recall@1000 for our experiments. We select these metrics because they are the common measures reported for these datasets. Recall is reported because we would like to know whether a gain in MAP is produced because of a higher number of retrieved relevant passages, or because of a better ranking (i.e. the ordering of the same number of relevant passages). For all results, we perform statistical significance test using a two-tailed paired t-test with Bonferroni correction.

7.2.3 Considered PRF Methods

While there are many methods of retrieval and PRF being proposed in the literature, in this first investigation we consider a subset of these methods that allows us to understand what the impact of feedback quality is on PRF effectiveness with respect to representation type, i.e. bag-of-words vs. dense vectors, and PRF type, i.e. learned vs. not learned.

Based on this, we decided to use BM25 as a representative bag-of-words method, noting that differences with other methods such as Language Modelling are often not substantial, along with ANCE, TCTv2-HN+, and DistillBERT-Balanced as representative dense retrievers. Note that ANCE is based on RoBERTA, TCTv2-HN+ on BERT and DistillBERT-Balanced on a reduced version of BERT (learned with knowledge distillation), and thus do differ to some extent in terms of representation.

Similarly, we selected the ANCE-PRF method [234] and its extensions to TCTv2-HN+ and DistillBERT-Balanced introduced in Chapter 4, as representative *learned* PRF methods. In these

Table 7.1: Statistics of the two datasets considered in our experiments. The statistics of the datasets after we remove the queries that do not have enough judged passages are labeled with (Filtered). We use the Filtered datasets in our experiments.

	#Queries	#Passages	Avg #J/Q	#Judgements
TREC DL 2019	43	8,841,823	215.3	9,260
TREC DL 2019 (Filtered)	36	8,841,823	217.7	7,838
TREC DL 2020	54	8,841,823	210.9	11,386
TREC DL 2020 (Filtered)	42	8,841,823	212.8	8,936

methods, in fact, a PRF encoder is fine-tuned to the relevance feedback task. We note that bag-of-words models do not have a corresponding complex trainable method (often tuning is performed but involves optimizing one or a handful of parameters, not the millions of parameters in the considered transformer-based models). We then selected the Vector-PRF method introduced in Chapter 5; specifically we used the Rocchio variant, which follows the general Rocchio PRF formula of Equation 7.1, but where the vectors are the actual dense representations from the dense encoders used for the first stage retrieval (details of parameters used are in Section 7.3.1). The method can be applied on top of any dense retriever, and we apply it to the 3 dense retrievers considered here. This method has an obvious correspondence in the bag-of-words space: it's the original Rocchio method – thus we consider Rocchio PRF on top of BM25, rather than the more popular RM3 method, to have a direct comparison between bag-of-words and dense retrievers under the same PRF strategy.

7.2.4 Baselines

We considering the following baselines:

- `BM25`: traditional first stage retriever, using the Anserini toolkit [229] with $k_1 = 0.82$ and $b = 0.68$.
- `BM25+RM3`: RM3 pseudo relevance feedback method [1] on top of BM25, as implemented in Anserini (using default parameters: 10 feedback terms, 10 feedback documents, original query weight 0.5). We use this approach as a representative bag-of-words PRF method, since previous research has found alternative bag-of-words PRF approaches achieve similar effectiveness [146]. We note that BM25+RM3 is a standard baseline for TREC DL.
- `BM25+Rocchio`: Rocchio pseudo relevance feedback method [179], we implement this approach as described in Rocchio algorithm.
- `ANCE`: first stage dense retriever [223]. We use the implementation in Pyserini [112] to reproduce the results.
- `TCTV2-HN+`: first stage dense retriever [115]. We use the implementation in Pyserini to reproduce the results.

- `DistilBERT-Balanced`: first stage dense retriever [73]. We use the implementation in Pyserini to reproduce the results.

- `{ANCE,TCTV2-HN+,DistilBERT-Balanced}+VPRF-Rocchio`:
  these are vector-based PRF approaches on top of ANCE [223], TCTV2-HN+ [115], and DistilBERT-Balanced [73] dense retrievers, proposed in Chapter 5, we use the same implementation from previous chapter.

- `{ANCE,TCTV2-HN+,DistilBERT-Balanced}-PRF`: transformer-based PRF method on top of ANCE [223], TCTV2-HN+ [115], and DistilBERT-Balanced [73], proposed by Yu et al. [234], and extended by Li et al. [102], we use the checkpoint provided by the authors and use the implementation in Pyserini and from Li et al. [102].

For all methods, be it bag-of-words or dense retrievers, learned PRF (a.k.a. ANCE-PRF and derivatives) or Vector-PRF, we use the implementations available in Anserini/Pyserini and the checkpoints made available by the corresponding authors of the techniques. We implement Rocchio PRF on top of the bag-of-words model in Pyserini and add this implementation to the GitHub repository here: `https://github.com/ielab/Noise-PRF`.

7.3 Results

7.3.1 Impact of Feedback Signal Quality on Different PRF Methods

To address **RQ2.1.1** and **RQ2.1.2**, we analyze how the retrieval effectiveness of different PRF methods changes across varying levels of feedback signal quality (strong, moderate, and weak) and how the underlying first-stage retriever influences this interaction. Tables 7.2 and 7.3 present results across multiple evaluation metrics including Mean Average Precision (MAP), Reciprocal Rank (RR), nDCG@{1,3,10}, and Recall@1000 (R@1000). For conciseness, we report results for PRF depth $k = 3$, as the trends observed for $k = 1$ are consistent and follow similar patterns, same for the figures included in this chapter.

Rocchio-Based PRF Methods

We evaluate both the classical Rocchio PRF [179] applied to BM25 and the Vector PRF variant (Chapter 5) for dense retrievers including ANCE, TCTv2-HN+, and DistillBERT. Parameter settings are summarized in Table 7.4. We use $k_1 = 0.82$ and $b = 0.68$ for BM25, $\alpha = 0.75$ and $\beta = 0.15$ for Rocchio on top of BM25 (all of these parameters are determined by performing a grid search on a random subset of MS MARCO development set), while for dense retrievers, we use $\alpha = 0.6$, $\beta = 0.4$ for TREC DL 2019 and TREC DL 2020 (when $k = 3$), and $\alpha = 0.5$, $\beta = 0.5$ for TREC DL 2020 (when $k = 1$), parameters are chosen based on the optimal VPRF's α, β configurations for the specific k on TREC DL 2019 and 2020 datasets (Tables 5.1 and 5.2 in Chapter 5).

Table 7.2: Effectiveness of PRF methods across different representations and PRF signal qualities on TREC DL 2019 with feedback signals from initial retrieved results. *R* stands for the Rocchio PRF method for bag-of-words, baselines are the PRF runs without control of the PRF signal quality (i.e., standard PRF on top *k* retrieved documents). For each signal quality, the PRF models are divided into three categories: Rocchio PRF on bag-of-words, VectorPRF-Rocchio on dense retrievers, and trained PRF on dense retrievers. Statistical significance analysis is performed using two-tailed paired t-test with Bonferroni correction; significant differences are marked with †. The best results under each metric is marked with **Bold**.

	Models	TREC DL 2019					
		MAP	RR	R@1000	nDCG@1	nDCG@3	nDCG@10
BASELINE	BM25	0.2697	0.7044	0.7687	0.5972	0.5298	0.4971
	ANCE	0.3908	0.8501	0.8031	0.7222	0.7022	0.6767
	TCTV2-HN+	0.4676	0.8788	0.8794	0.8009	0.7488	0.7309
	DistilBERT	0.4832	0.8763	0.8905	0.7454	0.7466	0.7319
	BM25+R	0.2350†	0.6328	0.8044	0.5000†	0.4963	0.4483
	ANCE+VPRF-R	0.4300†	0.8177	0.8179	0.7037	0.7023	0.6790
	TCTV2+VPRF-R	0.4949†	0.8682	0.8942	0.7917	0.7464	0.7406
	DistilBERT+VPRF-R	0.5156†	0.8606	0.8928	0.7731	0.7411	0.7387
	ANCE-PRF	0.4423†	0.8721	0.8293	0.7361	0.7204	0.7074
	TCTV2-PRF	0.4901†	0.8615	0.8888	0.7500	0.7606	0.7456
	DistilBERT-PRF	0.4996	0.8588	0.8968	0.7546	0.7648	0.7386
STRONG SIGNAL	BM25+R	0.3706†	0.8334†	0.8412	0.6505	0.6536†	0.6076†
	ANCE+VPRF-R	0.5119†	0.8816†	0.8531†	0.7708	0.7690	0.7511
	TCTV2+VPRF-R	0.5769†	0.9136	0.9292	**0.8002**	0.7957	0.7773
	DistilBERT+VPRF-R	**0.5931**†	**0.9410**	**0.9336**	0.7967	**0.8068**	**0.7971**
	ANCE-PRF	0.4907†	0.9060	0.8388	0.7986	0.7722	0.7484
	TCTV2-PRF	0.5326†	0.8807	0.9134	0.7654	0.7641	0.7600
	DistilBERT-PRF	0.5408†	0.8954	0.9124	0.7963	0.7821	0.7630
MODERATE SIGNAL	BM25+R	0.2641	0.5738†	0.7979	0.4842†	0.4899	0.4871
	ANCE+VPRF-R	0.4365	0.8278	0.8477	0.7161	0.6951	0.6845
	TCTV2+VPRF-R	0.4675	0.7999	0.9088	0.7164†	0.7060	0.7052
	DistilBERT+VPRF-R	0.4847	0.8143	0.9193	0.7029	0.7049	0.7075
	ANCE-PRF	0.4369†	0.7740	0.8324	0.6620	0.6905	0.6936
	TCTV2-PRF	0.4841	0.8550	0.8977	0.7558	0.7403	0.7338
	DistilBERT-PRF	0.5049	0.8779	0.9069	0.7755	0.7472	0.7369
WEAK SIGNAL	BM25+R	0.1957†	0.3917†	0.7641	0.2118†	0.2518†	0.2902†
	ANCE+VPRF-R	0.3915	0.7886	0.8130	0.6713	0.6526	0.6480
	TCTV2+VPRF-R	0.4408	0.8036	0.8875	0.6921†	0.6608	0.6692
	DistilBERT+VPRF-R	0.4441	0.8022	0.8970	0.6667†	0.6622†	0.6561
	ANCE-PRF	0.3929	0.7121†	0.8159	0.5232†	0.5733†	0.5906†
	TCTV2-PRF	0.4705	0.8487	0.8890	0.7315	0.7110	0.7053
	DistilBERT-PRF	0.5047	0.8757	0.9002	0.7639	0.7386	0.7262

Table 7.3: Effectiveness of PRF methods across different representations and PRF signal qualities on TREC DL 2020 with feedback signals from initial retrieved results. *R* stands for the Rocchio PRF method for bag-of-words, baselines are the PRF runs without control of the PRF signal quality (i.e., standard PRF on top *k* retrieved documents). For each signal quality, the PRF models are divided into three categories: Rocchio PRF on bag-of-words, VectorPRF-Rocchio on dense retrievers, and trained PRF on dense retrievers. Statistical significance analysis is performed using two-tailed paired t-test with Bonferroni correction; significant differences are marked with †. The best results under each metric is marked with **Bold**.

	Models	TREC DL 2020					
		MAP	RR	R@1000	nDCG@1	nDCG@3	nDCG@10
BASELINE	BM25	0.2870	0.6531	0.7938	0.5595	0.5155	0.4959
	ANCE	0.4047	0.8275	0.7804	0.7619	0.7479	0.6806
	TCTV2-HN+	0.4047	0.8275	0.7804	0.7619	0.7479	0.6806
	DistilBERT	0.4742	0.8677	0.8770	0.7778	0.7887	0.7207
	BM25+R	0.1936†	0.5198†	0.7447	0.4206†	0.4246†	0.3864†
	ANCE+VPRF-R	0.4220†	0.8377	0.7958	0.7540	0.7472	0.6841
	TCTV2+VPRF-R	0.4904†	0.8321	0.8655†	0.7817	0.7592	0.7144
	DistilBERT+VPRF-R	0.4974†	0.8899	0.9101†	0.8135	0.7973	0.7513
	ANCE-PRF	0.4340†	0.8881†	0.8286	0.8571†	0.7792	0.7275
	TCTV2-PRF	0.4864†	0.8774	0.8562†	0.8254†	0.8038	0.7331
	DistilBERT-PRF	0.4860	0.8810	0.8777	0.7976	0.7803	0.7306
STRONG SIGNAL	BM25+R	0.3936†	0.8859†	0.8422	0.7695†	0.7040†	0.6411†
	ANCE+VPRF-R	0.5259†	0.9164†	0.8377	0.8132	0.8158	0.7670†
	TCTV2+VPRF-R	0.6018†	0.9484†	0.9142†	0.8323	0.8227	0.7925†
	DistilBERT+VPRF-R	**0.6022**†	**0.9738**†	**0.9255**	**0.8462**	**0.8260**	**0.7955**†
	ANCE-PRF	0.4798†	0.8771	0.8241	0.7798	0.7685	0.7249
	TCTV2-PRF	0.5348†	0.9220	0.8780†	0.8307	0.8223	0.7683†
	DistilBERT-PRF	0.5264†	0.9082	0.8915	0.8185	0.8084	0.7542
MODERATE SIGNAL	BM25+R	0.1960†	0.3855†	0.7956	0.3740†	0.3821†	0.4039†
	ANCE+VPRF-R	0.3541†	0.6839†	0.8174	0.6177†	0.6286†	0.6091
	TCTV2+VPRF-R	0.3754	0.5763†	0.8830†	0.5321†	0.5665†	0.5792†
	DistilBERT+VPRF-R	0.4075†	0.6412†	0.8889	0.5747†	0.6140†	0.6440
	ANCE-PRF	0.3703	0.6882†	0.8143	0.6359†	0.6409†	0.6302
	TCTV2-PRF	0.4684†	0.8540	0.8583	0.7791	0.7545	0.7113
	DistilBERT-PRF	0.4701	0.8215	0.8816	0.7229	0.7499	0.7018
WEAK SIGNAL	BM25+R	0.1839†	0.3712†	0.7433	0.2153†	0.2389†	0.2866†
	ANCE+VPRF-R	0.3180†	0.6949†	0.7832	0.5575†	0.5609†	0.5250†
	TCTV2+VPRF-R	0.3440†	0.5926†	0.8352	0.4484†	0.4785†	0.4792†
	DistilBERT+VPRF-R	0.3904†	0.7441†	0.8603	0.5903†	0.5873†	0.5702†
	ANCE-PRF	0.3594	0.7185†	0.7911	0.6181†	0.6142†	0.5935
	TCTV2-PRF	0.4703†	0.8827	0.8551†	0.7748	0.7488	0.7001
	DistilBERT-PRF	0.4746	0.8626	0.8747	0.7474	0.7675	0.6991

Table 7.4: The Rocchio parameter settings for both datasets, with different PRF depths and different models.

	Depth		TREC DL 2019	TREC DL 2020
BOW	all	α	0.75	0.75
		β	0.15	0.15
Dense Retrievers	1	α	0.6	0.5
		β	0.4	0.5
	3	α	0.6	0.6
		β	0.4	0.4

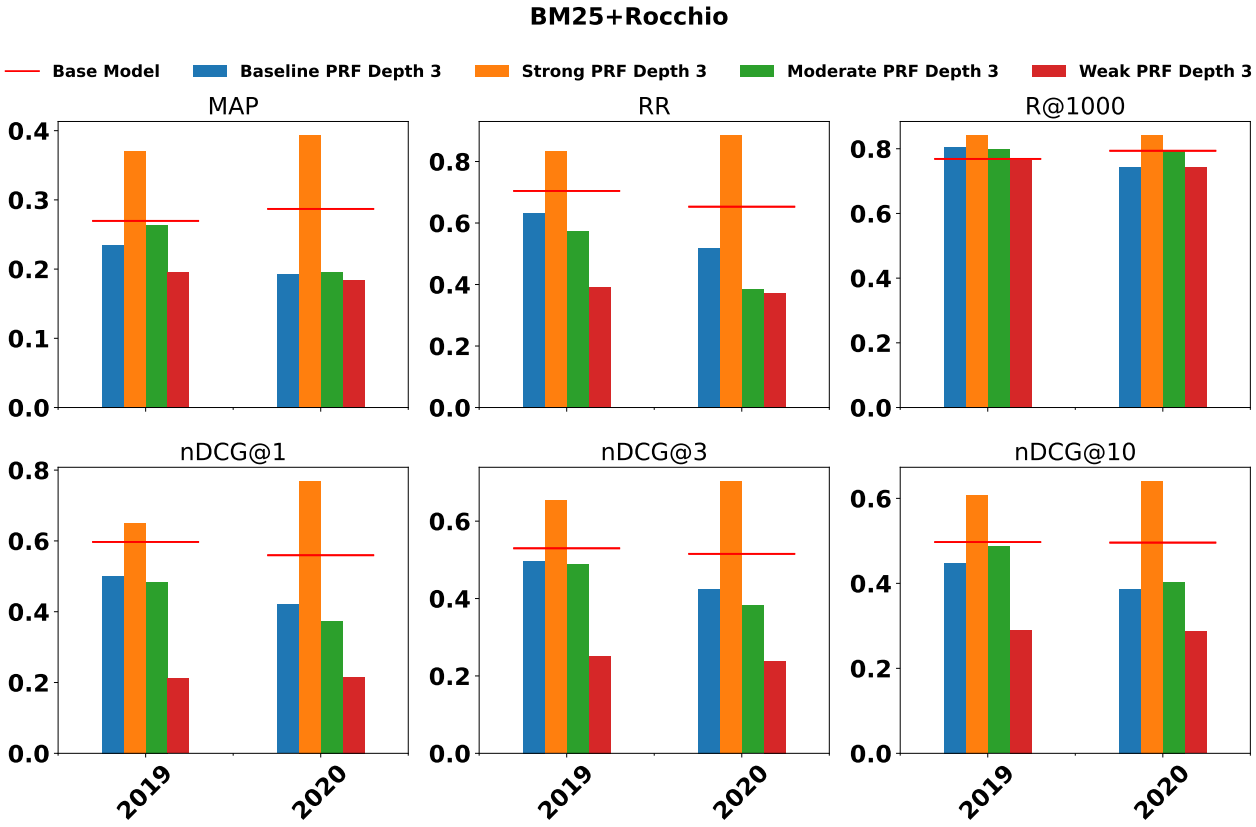


Figure 7.1: The effectiveness of BM25+Rocchio under different PRF signal qualities. Where 2019 represents the TREC DL 2019 dataset and 2020 represents the TREC DL 2020 dataset. Red line indicates the base model BM25's performance on each dataset.

When using uncontrolled PRF signals (i.e., the top-k documents from first-stage retrieval without any filtering), for BM25+Rocchio, as shown in Figure 7.1, shows inconsistent behavior. On TREC DL 2019, slight gains are observed in R@1000 and nDCG@3 with PRF, while MAP, RR, and nDCG@1 decline or remain unchanged. On TREC DL 2020, improvements are often minimal or absent, with nDCG@1 remaining roughly equivalent to the BM25 baseline. These results suggest that uncontrolled PRF may introduce harmful noise, especially when non-relevant documents dominate the top-k.

In contrast, when we explicitly control the quality of the feedback signal, performance improves significantly. With strong signals, all PRF methods (as shown in Figure 7.1, Figure 7.2, Figure 7.3, and Figure 7.4) exhibit consistent gains across all metrics and datasets. For instance, MAP, nDCG@1, and

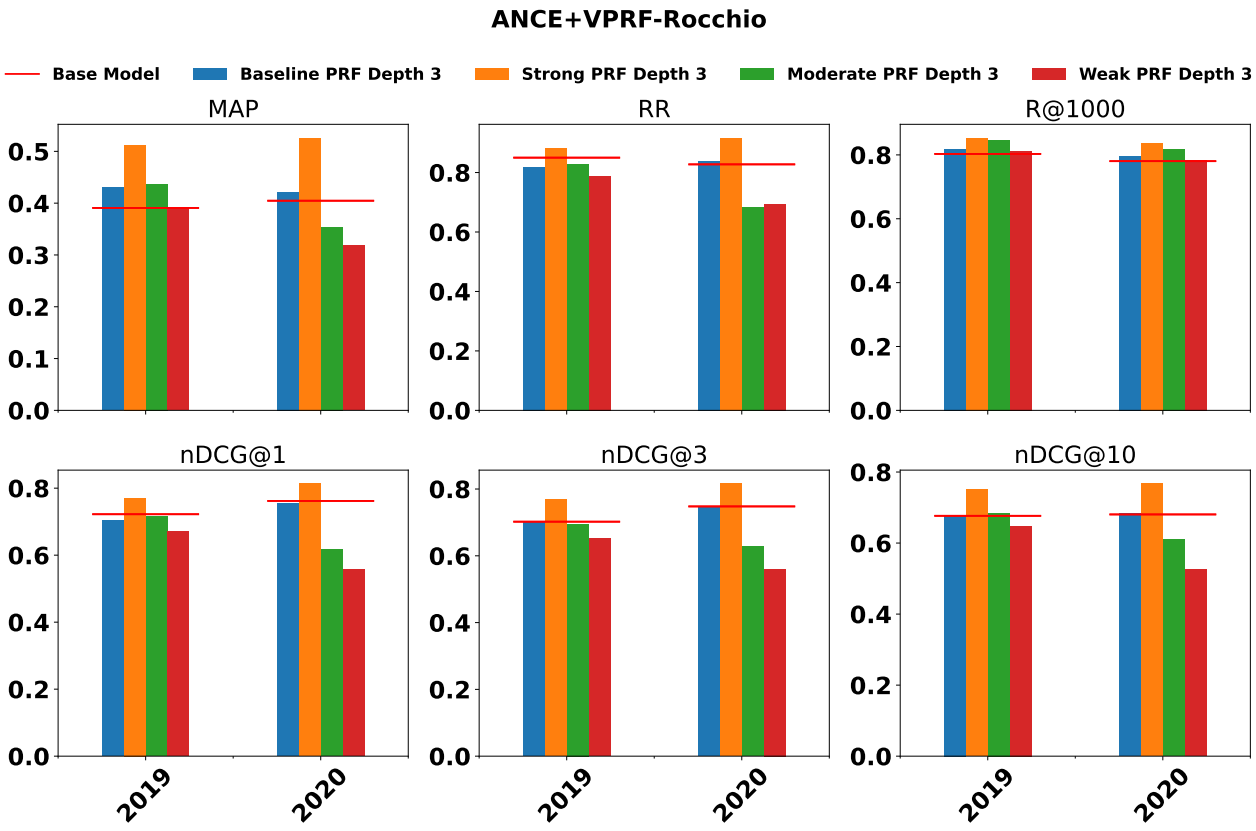


Figure 7.2: The effectiveness of ANCE+VPRF-Rocchio under different PRF signal qualities. Where 2019 represents the TREC DL 2019 dataset and 2020 represents the TREC DL 2020 dataset. Red line indicates the base model ANCE's performance on each dataset.

R@1000 all improve significantly under strong signals, reflecting the benefit of using strong relevant feedback passage signals in PRF.

Under moderate signal conditions, effectiveness varies by different methods. BM25+Rocchio shows only marginal improvements, mostly limited to R@1000, while other metrics such as MAP and nDCG@1 decline or remain on par. For dense retrievers, the results are more fluctuated: ANCE+VPRF-Rocchio continues to deliver strong improvements, especially in MAP and R@1000, demonstrating robustness to moderate signal degradation. TCTv2+VPRF-Rocchio and DistillBERT+VPRF-Rocchio, however, show large performance drops relative to their strong signal performance. Most metrics for these models under moderate signals fall below their respective non-PRF baselines.

With weak signals, the limitations of PRF methods become clear. BM25+Rocchio experiences dramatic performance degradation across all metrics, with some evaluation scores declined by more than 60% relative to the baseline. Among dense retrievers, only ANCE+VPRF-Rocchio still exhibits limited robustness, by maintaining small improvements for R@1000 in TREC DL 2019 and 2020. TCTv2+VPRF-Rocchio shows minor improvements only for R@1000 at PRF depths 1 and 3, while DistillBERT+VPRF-Rocchio suffers degradations across the board in TREC DL 2020, with all metrics performing worse than the non-PRF baseline and some losses exceeding 40%.

These findings highlight two key insights relevant to RQ2.1. First, feedback signal quality has a direct and significant impact on PRF effectiveness. Gains observed under strong signals often

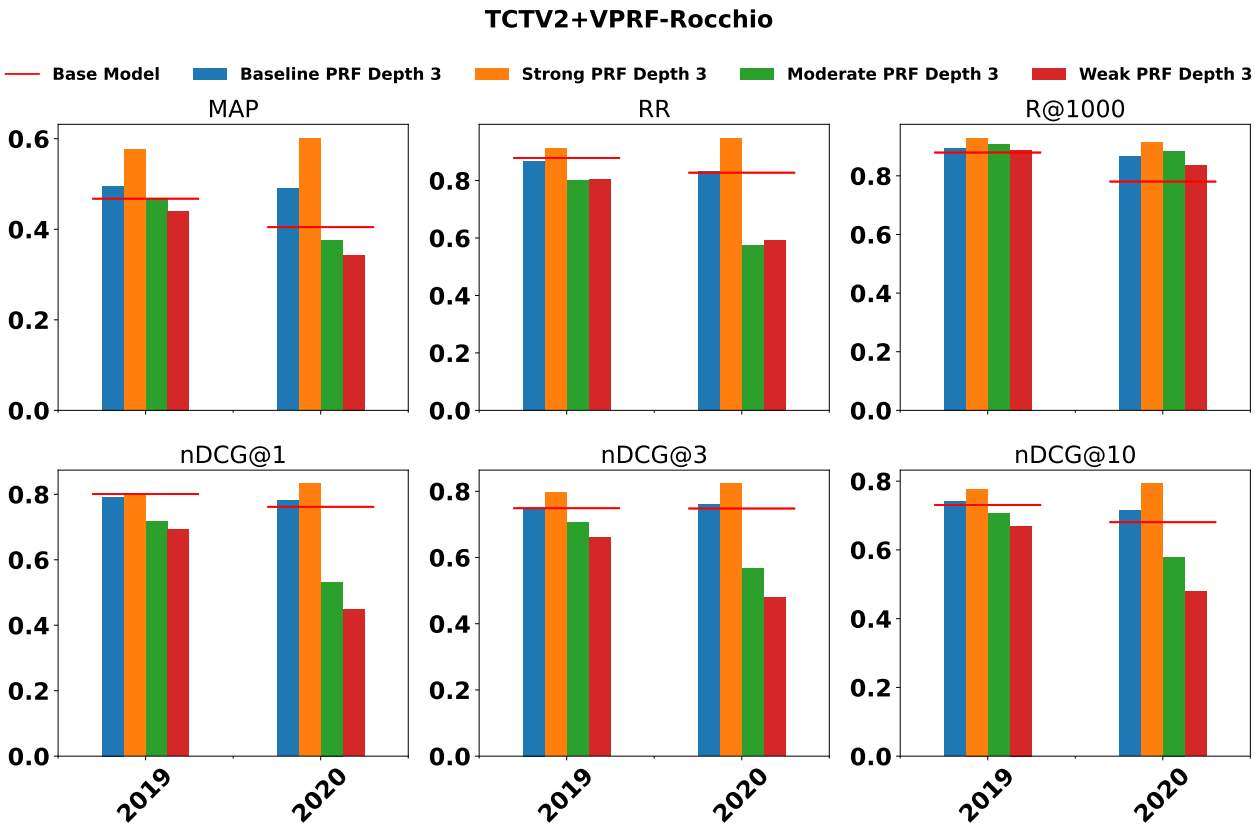


Figure 7.3: The effectiveness of TCTV2+VPRF-Rocchio under different PRF signal qualities. Where 2019 represents the TREC DL 2019 dataset and 2020 represents the TREC DL 2020 dataset. Red line indicates the base model TCTV2 HN+'s performance on each dataset.

disappear, or even reverse, under weaker conditions. Second, the choice of first-stage retriever and its compatibility with the PRF method plays a critical role. ANCE-based configurations are consistently more robust under signal degradation compared to TCTv2 and DistillBERT, indicating that some retriever+PRF combinations are more resilient to noisey feedback signals than others.

Learned PRF Methods

We also examine learned PRF methods: ANCE-PRF [234], TCTv2-PRF [102], and DistillBERT-PRF [102]. These approaches incorporate learned query encoders trained to integrate feedback signals directly, we also present them in Figure 7.5, Figure 7.6, and Figure 7.7 respectively.

Under uncontrolled signal conditions, all three learned methods consistently outperform their respective baseline models without PRF across most metrics on both datasets. This demonstrates the overall strength of learned query adaptation, even when the quality of the feedback is not highly reliable.

When using strong signals, massive improvements on effectiveness across nearly all metrics and models can be observed. Learned PRF models show particularly strong performance in MAP, nDCG@10, and R@1000, indicating their ability to fully exploit high-quality feedback. The only exception is TCTv2-PRF for nDCG@1 on TREC DL 2019, where no improvement is observed.

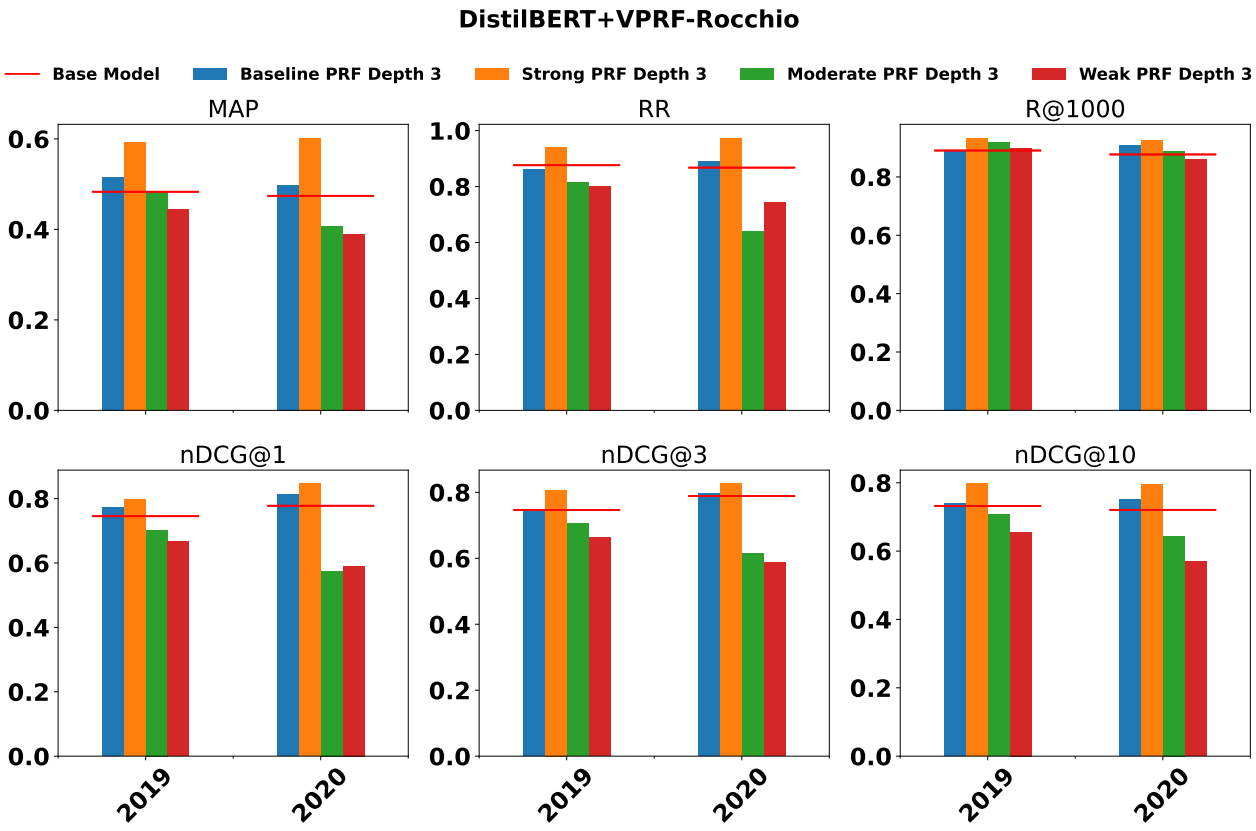


Figure 7.4: The effectiveness of DistilBERT Balanced+VPRF-Rocchio under different PRF signal qualities. Where 2019 represents the TREC DL 2019 dataset and 2020 represents the TREC DL 2020 dataset. Red line indicates the base model DistilBERT Balanced's performance on each dataset.

With moderate signals, a shift in performance behavior is observable. While improvements in deep metrics (e.g., MAP, R@1000) persist, shallow metrics like nDCG@1 and RR often stay on par or decline. For example, DistillBERT-PRF improves MAP and recall on both datasets, but nDCG@1 performance degrades relative to the baseline. This suggests that learned PRF models retain some ability to benefit from moderately relevant feedback, but their precision at high ranks is more vulnerable to quality degradation.

Under weak signals, effectiveness declines across all models and metrics, but the degree of degradation is obviously less severe than with Rocchio-based methods. While some performance losses exceed 20%, most learned PRF models still retain partial gains in deeper metrics like MAP and R@1000. In contrast to classic Rocchio and Vector PRF, no catastrophic failure is observed.

Summary

The experimental results provide a clear answer to **RQ2.1.1**: the effectiveness of PRF methods varies substantially depending on the quality of the feedback signal. When the feedback consists of strongly relevant passages, all PRF methods, both Rocchio-based and learned, yield consistent and significant improvements across all evaluation metrics. In contrast, under moderate signal conditions, effectiveness gains become concentrated in deeper metrics such as MAP and Recall@1000, while shallow metrics like nDCG@1 and RR often stagnate or decline. When weak signals are used, performance drops

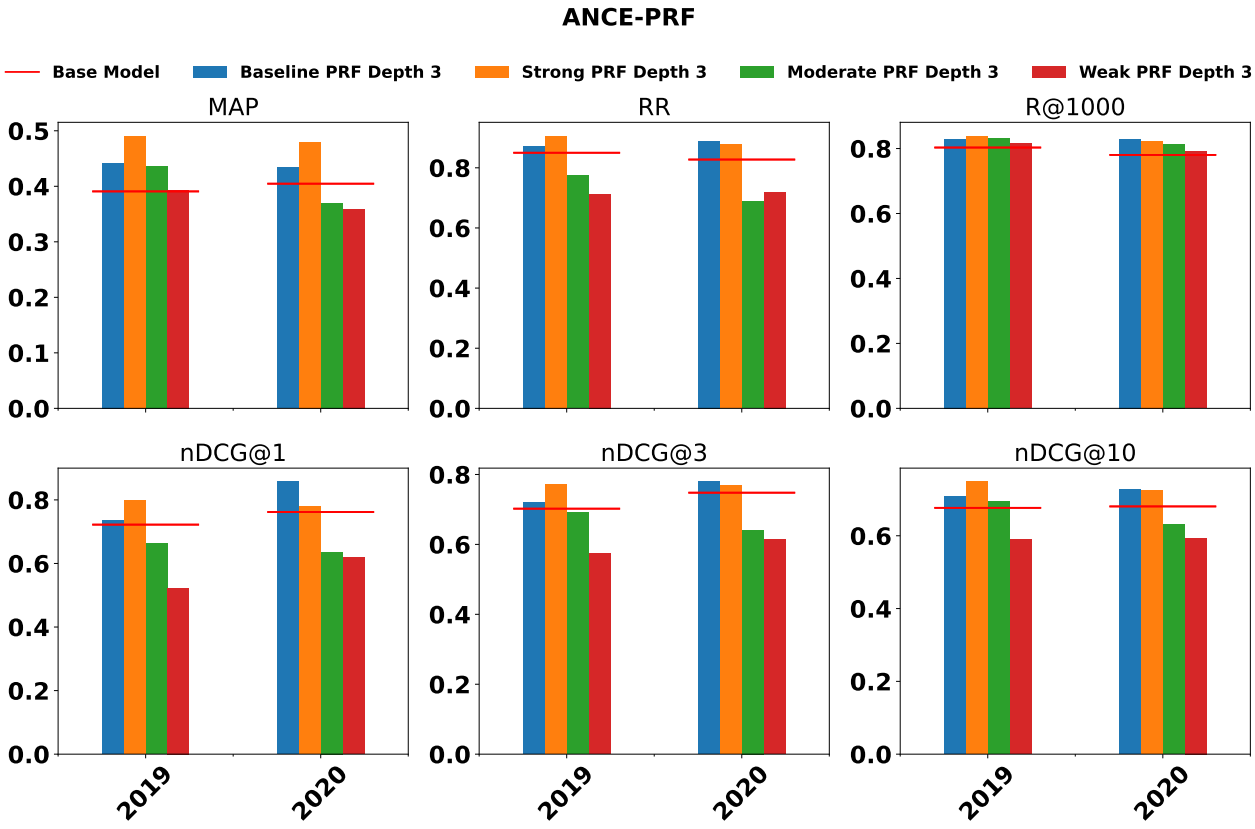


Figure 7.5: The effectiveness of ANCE-PRF under different PRF signal qualities. Where 2019 represents the TREC DL 2019 dataset and 2020 represents the TREC DL 2020 dataset. Red line indicates the base model ANCE's performance on each dataset.

markedly across most methods, with Rocchio-based approaches suffering the most severe degradation, including losses exceeding 50% in some cases.

With respect to **RQ2.1.2**, our findings confirm that the choice of first-stage dense retriever directly influences both the quality of the feedback signal and the resulting effectiveness of PRF. Specifically, ANCE exhibits greater robustness across all signal conditions, maintaining stronger performance under moderate and weak feedback compared to TCTv2 and DistillBERT. This variation indicates that certain retrievers are inherently more compatible with PRF strategies. Furthermore, learned PRF methods consistently demonstrate higher resilience to signal degradation than their Rocchio-based counterparts, particularly in scenarios involving noisy or low-relevance feedback.

7.3.2 Robustness of PRF Across Feedback Signal Qualities and Representation Types

We now turn to evaluating the robustness of different representations under varying feedback signal conditions with PRF, in order to address the **RQ2.2.1** and **RQ2.2.2**.

To isolate the impact of representation type on PRF robustness, we focus our analysis on the Rocchio-based PRF method, referred to as Vector PRF (VPRF-Rocchio) for dense retrievers. This setup enables a direct comparison between bag-of-words and dense vector representations, as Rocchio

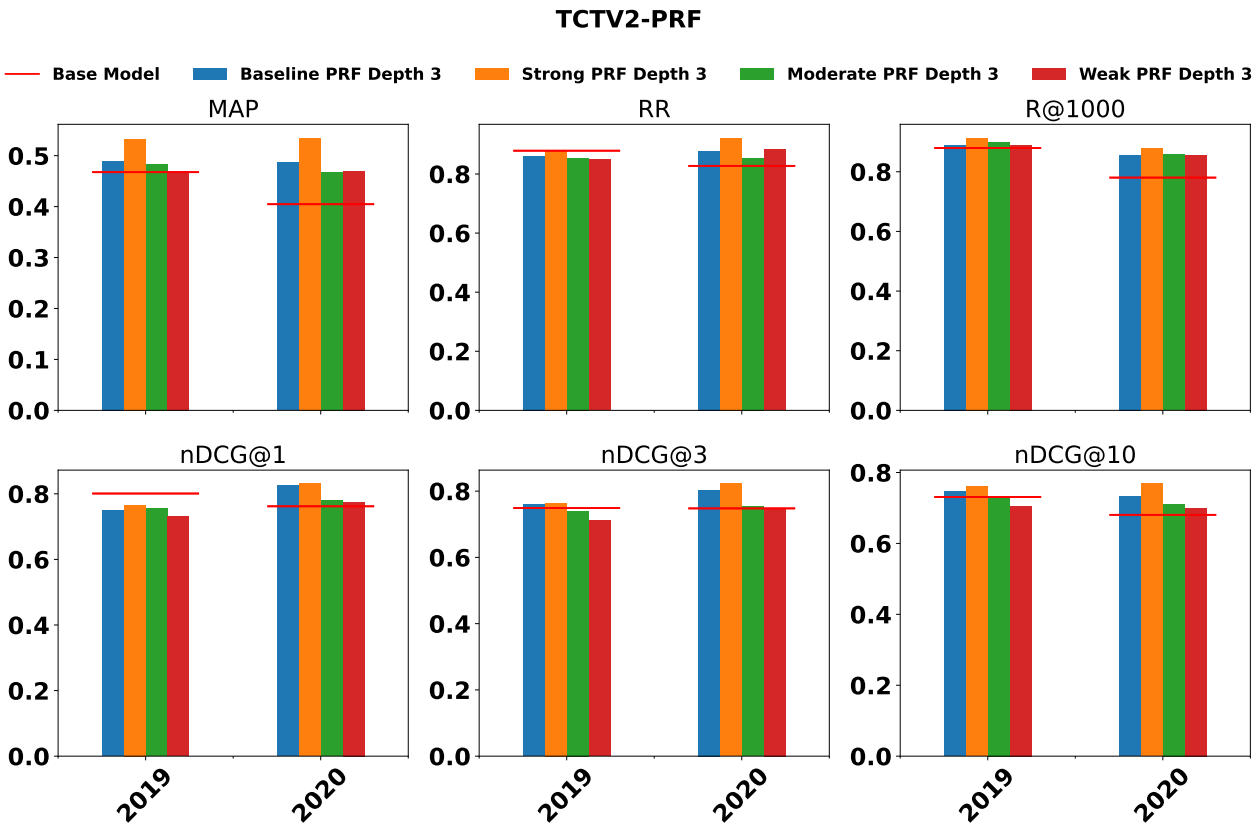


Figure 7.6: The effectiveness of TCTV2-PRF under different PRF signal qualities. Where 2019 represents the TREC DL 2019 dataset and 2020 represents the TREC DL 2020 dataset. Red line indicates the base model TCTV2 HN+'s performance on each dataset.

is the only PRF method implemented uniformly across both dense and sparse representations. The corresponding results are summarized in Tables 7.2 and 7.3.

For the bag-of-words representation (BM25 + Rocchio), performance is highly sensitive to feedback signal quality. While strong signals lead to meaningful improvements, effectiveness degrades sharply as the signal weakens. When moving from strong to weak feedback, relative performance losses exceed 60% on some metrics (over 70% on nDCG@1 for TREC DL 2020). This degradation trend is consistent across both TREC DL 2019 and 2020, and affects all evaluation metrics including MAP, RR, and nDCG@1. These results indicate that the traditional Rocchio method applied to sparse representations is highly vulnerable to noisy feedback and lacks robustness across signal conditions.

In contrast, dense representations demonstrate more stable behavior. Among them, ANCE+VPRF-Rocchio consistently exhibits greater robustness to feedback signal degradations. Even under weak signal qualities, the worst-case losses remain within 30%, a substantial improvement over Rocchio with BM25. This suggests that ANCE's underlying representation space is more tolerant to the inclusion of noisy or non-relevant passages in the feedback loop. Consequently, ANCE+VPRF-Rocchio can be considered the most robust configuration among those examined for this PRF method, effectively addressing RQ2.2.1.

However, not all dense retrievers perform equally under signal degradation. TCTv2+VPRF-Rocchio and DistillBERT+VPRF-Rocchio show significantly less stable patterns. Both configurations

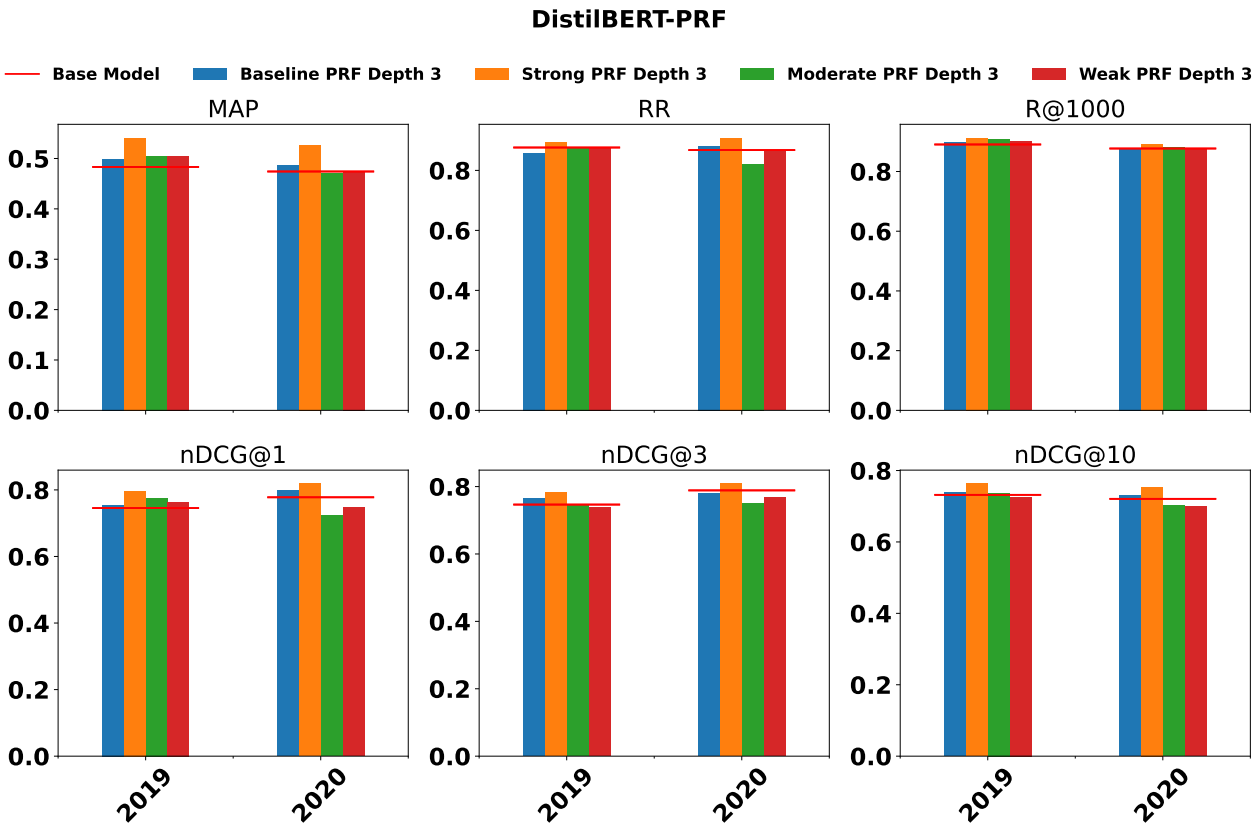


Figure 7.7: The effectiveness of DistilBERT-PRF under different PRF signal qualities. Where 2019 represents the TREC DL 2019 dataset and 2020 represents the TREC DL 2020 dataset. Red line indicates the base model DistilBERT Balanced's performance on each dataset.

experience performance drops of over 50% on several metrics when the feedback signal quality shifts from strong to weak. These results indicate that while dense representations in general are more robust than sparse ones, the degree of robustness varies substantially depending on the specific retriever used.

Taken together, these findings also answer RQ2.2.2. The magnitude of effectiveness degradation from strong to weak feedback signals varies significantly across different PRF approaches, even when the underlying PRF mechanism is held constant (i.e., Rocchio). Bag-of-words representations suffer the greatest declines, while ANCE-based dense representations maintain relatively consistent performance. TCTv2 and DistillBERT fall in between, offering some improvement over sparse methods, but still exhibiting substantial sensitivity to weak signals.

Summary

In conclusion, our analysis shows that both the type of representation and the specific dense retriever used play a critical role in determining the stability of PRF methods under varying feedback signal conditions. PRF applied to ANCE demonstrates the highest robustness, while the same method applied to TCTv2 or DistillBERT shows greater vulnerability to noise. These differences highlight the importance of carefully selecting the retriever-PRF pairing in scenarios where feedback quality cannot be guaranteed.

7.4 Summary

In this chapter, we presented a systematic investigation into how the quality of the feedback signal influences the effectiveness of PRF methods in the context of passage retrieval. We evaluated PRF under varying signal quality conditions (strong, moderate, and weak) and found that feedback signal quality plays a critical role in PRF effectiveness.

Our results demonstrated that strong feedback signals consistently lead to substantial effectiveness gains across all PRF methods and metrics. In contrast, moderate signals yield improvements primarily in deeper metrics such as MAP and Recall@1000, while weak signals often lead to performance degradation, particularly in precision-focused metrics such as nDCG@1 and RR.

We further observed that the stability of PRF performance under less reliable feedback signals varies significantly across PRF approaches. Learned PRF methods (e.g., ANCE-PRF and its variants) exhibit stronger robustness to noise, maintaining acceptable performance even under weak signals. In contrast, not-learned methods, such as classic Rocchio and its vector-based variant (VPRF-Rocchio), are more sensitive to signal quality, with sharp performance drops observed under weak feedback conditions.

Additionally, we showed that dense representations provide more stable performance than bag-of-words representations across the entire spectrum of signal qualities. Among dense retrievers, ANCE combined with VPRF-Rocchio demonstrated better robustness to feedback degradation compared to TCTv2 and DistillBERT, highlighting the importance of retriever architecture in PRF effectiveness.

To further validate these conclusions, we also conducted a complementary analysis using feedback passages sampled directly from official relevance judgments (QRels [2]) with the exact same settings we used for initial retrieved results. By sampling the feedback signals directly from QRels, we effectively remove the influence of first-stage retriever bias. This idealized setup yielded performance trends consistent with those observed in the retrieval-based experiments, confirming that our findings regarding signal quality are intrinsic to the PRF mechanisms rather than artifacts of the retrieval ranking. Therefore, to avoid redundancy and maintain conciseness, we refer readers to the detailed results for this QRels-based validation to Tables A.1 – A.8 in Appendix A.

While this study provides new insights into the relationship between feedback signal quality and PRF performance, it also comes with limitations. Specifically, we did not evaluate mixed-signal scenarios (where the top-k passages vary in relevance) or explore deeper PRF feedback depths beyond $k = 3$[3]. These decisions were made to maintain experimental control and ensure interpretability. Future work can be done to include more diverse feedback compositions and a broader range of PRF strategies (e.g., ColBERT-PRF [214]).

Building on the findings of this chapter, the next chapter introduces TPRF (Transformer-based Pseudo-Relevance Feedback), a new approach designed to address key limitations identified in ANCE-PRF reproduction from Chapter 4, Vector-based PRF from Chapter 5, and the current chapter. First,

[2]The file containing the relevance assessments.

[3]While some dense retrieval based PRF methods are limited in terms of the maximum number of passage they can consider as feedback [102, 234], others are not [104].

while learned PRF methods such as ANCE-PRF have demonstrated strong effectiveness, they rely on large, computationally expensive query encoders, making them unsuitable for deployment in resource-constrained environments. TPRF addresses this by leveraging a lightweight transformer architecture with only a few layers, offering fast inference and a significantly reduced computational footprint.

Second, although vector-based PRF methods like VPRF-Rocchio are inherently efficient and do not require any training, they rely on somewhat tuned parameters (α and β) to interpolate between the original query vector and feedback vectors. As observed in Chapter 5 and further investigated in Chapter 12, the optimal values for these parameters can be varied from query to query. However, tuning these parameters on a per-query basis is infeasible in practice, and even tuning over a full development set does not guarantee optimal performance for every query. Transformer-based PRF (TPRF) is introduced in next chapter to learn to approximate these optimal interpolation weights automatically, by conditioning on the input query and its feedback context, thereby improving retrieval effectiveness without requiring manual parameter tuning.

Third, this chapter has shown that PRF effectiveness is highly sensitive to the quality of the feedback signal. In particular, VPRF-Rocchio performs well under strong signals but degrades significantly under moderate or weak conditions. TPRF is designed to be signal-aware, it learns to identify the quality of the feedback signal and adapt the interpolation parameters accordingly, improving the robustness of PRF across diverse retrieval scenarios.

The following chapter presents the design and training of the TPRF model and demonstrates how it effectively balances efficiency and effectiveness by learning query-specific feedback interpolation strategies using compact transformer layers. TPRF offers a practical solution that combines the strengths of both vector-based and learned PRF approaches, making it well-suited for real-world applications where computational efficiency is critical.

Chapter 8

TPRF – A Transformer-Based Efficient Pseudo-Relevance Feedback Model

While previous chapters in Part 1 have shown that PRF can significantly improve the effectiveness of dense retrievers, many existing approaches rely on large pre-trained language models and computationally intensive architectures. These methods, while highly effective, are not suitable for deployment in real-world environments where memory, latency, and compute constraints are critical, such as on smartphones, smartwatches, or cost-sensitive cloud infrastructure without access to powerful GPUs. In such settings, the high inference cost and large memory footprint of traditional PRF techniques render them impractical for deployment.

This chapter focuses on investigating the following research question:

RQ2.3: Can Pseudo-Relevance Feedback be both efficient and effective?

To this end, we propose a lightweight and efficient transformer-based PRF model, called Transformer-based Pseudo Relevance Feedback (TPRF). Unlike prior methods that require re-encoding long query-passage inputs with large-scale language models [218, 234], TPRF operates directly on dense vector representations. It uses a compact transformer to model the interactions between the original query and its top-ranked feedback passages, learning to generate an enhanced query embedding without text-based inference. As a result, TPRF substantially reduces query latency and model size, enabling its deployment on CPU-only systems and resource-limited devices.

Another key motivation for TPRF arises from a limitation of our previously proposed VPRF method from Chapter 5, which interpolates the original query vector with the mean vector of feedback passages using manually tuned Rocchio parameters (α, β). Although VPRF is inherently lightweight and effective, its reliance on static hyperparameters limits its adaptability. In practice, optimal values of α and β are highly query-dependent. However, tuning them on a per-query basis is infeasible, as it necessitates ground-truth relevance judgments that are unavailable at inference time and imposes a prohibitive computational cost due to the need for multiple retrieval iterations. Furthermore, VPRF typically relies on values tuned over a held-out training set, which can lead to suboptimal performance for individual queries. TPRF addresses this limitation by learning, in a data-driven and query-specific

manner, how to combine the original query and feedback signals, this allows TPRF to effectively learn how to assign adaptive weighting analogous to α and β without requiring manual tuning.

This chapter presents a comprehensive empirical evaluation of TPRF across several configurations and compares its effectiveness, efficiency, and scalability with existing methods, particularly ANCE-PRF (Chapter 4). Experiments on the TREC DL 2019 and 2020 benchmarks, chosen for their high judgment density and alignment with standard dense retrieval evaluations, show that while ANCE-PRF remains the most effective in absolute terms, TPRF consistently outperforms the base dense retriever and achieves competitive results with dramatically lower computational cost. Notably, even the smallest TPRF variant, the one only with a single transformer layer and one attention head, still demonstrates strong performance and extremely low query latency.

Together, these findings demonstrate that PRF can be designed to be both effective and efficient. TPRF provides a general, scalable, and deployment-ready solution to the challenge of incorporating feedback signals in neural retrieval under real-world resource constraints, while also eliminating the need for manual parameter tuning [1].

8.1 Transformer-Based Pseudo-Relevance Feedback Model

We introduce the Transformer-based Pseudo-Relevance Feedback (TPRF) model, designed to overcome the specific limitations of prior dense PRF approaches. By operating directly on dense vector representations, TPRF circumvents the costly tokenization and large-scale encoding required by text-based methods like ANCE-PRF (Chapter 4), thereby preserving the efficiency benefits of our earlier Vector-based PRF (Chapter 5). However, unlike the Vector-based PRF which relies on static, manually tuned Rocchio parameters (α and β), TPRF utilizes a lightweight transformer to learn query-specific feedback integration strategies. This data-driven approach eliminates the need for infeasible per-query parameter tuning while maintaining a minimal computational footprint.

8.1.1 Overview of TPRF Architecture

The TPRF framework consists of three main components: a dense retriever, a lightweight transformer module, and a training strategy optimized for efficiency and generalizability. An overview of the architecture is illustrated in Figure 8.1.

The dense retriever is responsible for generating fixed-size vector representations for passages in the corpus during indexing time. These vectors are stored in an efficient structure for approximate nearest neighbor (ANN) search, such as a FAISS [41] index. At query time, the dense retriever encodes the input query into a dense vector and retrieves the top-k most similar passages based on vector similarity (e.g., dot product). Importantly, TPRF does not assume a specific dense retriever architecture and is fully agnostic to the underlying retrieval model. In this chapter, we demonstrate TPRF in conjunction with multiple dense retrievers, including ANCE [223], TCTv2 [114], and DistillBERT [73].

[1] The training code is available at: `https://github.com/ielab/Transformer-based-PRF`

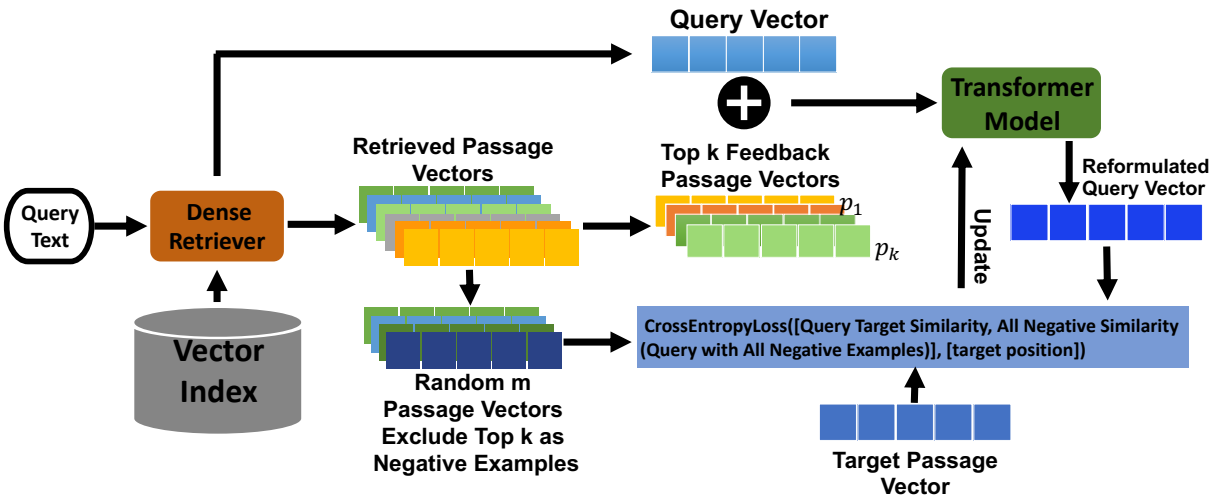


Figure 8.1: The proposed architecture for training TPRF. The initial dense retriever can be any dense retrievers.

8.1.2 PRF Representation via Lightweight Transformer

Following the initial retrieval stage, TPRF collects the dense representations of the query and its top-k retrieved passages, resulting in a matrix of shape $(k+1, 768)$, where each row corresponds to a 768-dimensional embedding (this dimension is determined by the underlying dense retriever). This matrix, consisting of the query embedding followed by feedback passage embeddings, is used as input to a compact transformer encoder. The output of this transformer is a single 768-dimensional vector, representing a refined query embedding that incorporates the feedback signal. This new embedding is then used to perform a second round of retrieval over the original dense index, producing the final ranking.

At the core of TPRF is a stack of transformer encoder layers, implemented following the architecture of the vanilla Transformer proposed by Vaswani et al. [205]. These layers apply multi-head self-attention and feed-forward transformations to the input sequence of embeddings, enabling the model to learn contextual interactions between the original query and the feedback passages. To capture positional information, particularly the rank order of feedback passages, we apply sinusoidal positional encoding as originally described by Vaswani et al. [205] to the input vectors prior to attention computation.

The output of the final transformer layer corresponding to the first position (i.e., the original query) is extracted and used as the refined query embedding. As per standard design, each transformer layer incorporates residual connections and layer normalization to facilitate stable training. We explore different architectural configurations, including variations in the number of attention heads and the number of encoder layers, to evaluate the trade-off between model complexity and retrieval effectiveness.

8.1.3 Training TPRF with Hard Negative Sampling

At training time, after the initial retrieval is performed using the baseline dense retriever, we select the top-k passages as the PRF signal. To construct the training samples, we randomly sample 20 negative passages from among those ranked between positions 10 and 200 in the initial ranking produced by the dense retriever. This negative sampling strategy follows the approach used in prior work on ANCE-PRF (Refer to Chapter 4) [234]. Additionally, we sample one positive passage vector from the set of labelled relevant passages in the MS MARCO training queries (see Section 8.2 for details on datasets).

Inspired by prior studies by Gao et al. [51] and Gao and Callan [48], we also explore an extension involving the use of additional hard negatives. Specifically, we include the two passages ranked immediately after the selected PRF passages as hard negatives, to further challenge the model during training.

Each training instance consists of one positive passage $\mathbf{p}^+$ and a set of negatives $\{\mathbf{p}_1^-, \mathbf{p}_2^-, \ldots, \mathbf{p}_{m-1}^-\}$ sampled from the top retrieved results. These are used to supervise the training of the TPRF transformer model using a cross-entropy loss (Eq. 8.1) over the similarity scores between the refined query representation and the passage vectors. The similarity score $s(q,p)$ is computed as the dot product between the query embedding $\mathbf{q}$ and the passage embedding $\mathbf{p}$. The loss function is defined as:

$$\mathcal{L}_{CE} = -\log \frac{e^{s(\mathbf{q},\mathbf{p}^+)}}{e^{s(\mathbf{q},\mathbf{p}^+)} + \sum_{\mathbf{p}^-} e^{s(\mathbf{q},\mathbf{p}^-)}} \tag{8.1}$$

This formulation encourages the model to assign higher similarity to the positive passage relative to the sampled negatives, thereby learning more discriminative query representations conditioned on the feedback signals.

8.1.4 Theoretical Advantages of TPRF

A key advantage of TPRF lies in its architectural simplicity and flexibility. Unlike methods such as ANCE-PRF (Chapter 4), which rely on pre-trained language models such as RoBERTa and incorporate the fine-tuned ANCE encoder, TPRF employs a vanilla transformer that is trained from scratch specifically for the PRF task. This absence of pre-training allows TPRF to remain compact in terms of parameter count, resulting in lower memory requirements and reduced encoding time during inference.

Another significant distinction is the nature of the input to the PRF model. ANCE-PRF and related approaches rely on textual inputs, typically utilizing concatenated query and feedback passage texts. On the other hand, TPRF operates entirely on dense vector representations. This design removes the dependence on large-scale tokenization and textual encoding, and more importantly, eliminates the input length constraint imposed by the maximum token limit of pre-trained language models (e.g., 512 tokens for RoBERTa). As a result, TPRF is not restricted in the number of feedback passages it can incorporate, enabling scalability to deeper PRF signals.

This vector-based input structure aligns with our earlier Vector-based PRF methods from Chapter 5 [104], which also use dense embeddings in place of text. However, those approaches either assign

equal importance to the query and feedback vectors (Vector-based PRF Average), or require manually tuned interpolation parameters (Vector-based PRF Rocchio). In contrast, TPRF learns to automatically determine the importance of each feedback passage and the original query in a data-driven, query-specific manner. The self-attention mechanism in TPRF allows it to dynamically adjust these weights for each query, without requiring fixed global parameters.

Another strength of TPRF is its model-agnostic design. While ANCE-PRF is tightly coupled with the ANCE encoder, and according to Chapter 4 [102], it does not generalize well to other dense retrievers, TPRF is not tied to any specific retriever. The usage of independently computed dense embeddings allows it to be combined with various dense retrieval backbones. In this chapter, we demonstrate its applicability across ANCE [223], TCTv2 [114], and DistillBERT [73], validating its generalizability.

In terms of efficiency, TPRF provides substantial advantages in query latency. ANCE-PRF incurs significant inference overhead as the number of feedback passages increases, this is due to longer input sequences and more expensive token-level attention computations. For instance, with just three feedback passages, the ANCE-PRF model requires nearly 40 times the inference time compared to ANCE without PRF. In contrast, TPRF's input consists of a fixed-dimensional matrix where each row is a single vector. Increasing the number of feedback passages adds more rows, but does not increase the dimensionality or the complexity of attention computation in the same manner as textual input. Empirical results show that even with 100 feedback passages, TPRF remains faster than ANCE-PRF configured with just 10 passages.

To summarize, TPRF offers both conceptual and practical advantages: it avoids the constraints of textual models, removes the need for manual parameter tuning, generalizes across retrievers, and scales more efficiently to deeper feedback without sacrificing effectiveness.

8.2 Empirical Evaluation

We now describe how to evaluate the performance of our proposed TPRF method and to compare it against established PRF approaches in the literature. The goal of this evaluation is to provide empirical answers to the following research questions, which collectively examine TPRF's effectiveness, efficiency, and scalability across various model configurations and dense retriever backbones:

RQ2.3.1: What is the trade-off between retrieval effectiveness, model size, and query latency across different TPRF configurations?

RQ2.3.2: How does TPRF scale in terms of query latency as PRF depth increases?

RQ2.3.3: How does the model size of TPRF compare to existing PRF approaches, and what are the implications for deployability?

RQ2.3.4: How does the retrieval effectiveness of TPRF compare to ANCE-PRF and other Vector-Based PRF baselines across different dense retrievers?

In the following sections, we describe the datasets, evaluation metrics, and baseline methods used in our experiments, followed by detailed results and analyses corresponding to each of the above research questions.

8.2.1 Datasets and Evaluation Metrics

We evaluate our proposed method on the TREC Deep Learning Track datasets from 2019 [27] and 2020 [26], denoted as TREC DL 2019 and TREC DL 2020, respectively. Both datasets use the same passage corpus from MS MARCO [151]. We exclude MS MARCO development queries from evaluation due to their sparse judgments (averaging one relevant passage per query), which prevent accurate assessment of the recall-oriented metrics essential for PRF. Instead, models are trained using the MS MARCO training set ($\approx$ 1 million queries).

For effectiveness evaluation, we report the following standard information retrieval metrics: Mean Average Precision (MAP), Reciprocal Rank (RR), Recall@1000 (R@1000), and normalized Discounted Cumulative Gain (nDCG) at cutoffs 1, 3, and 10 (nDCG@1, nDCG@3, and nDCG@10). Statistical significance is assessed using paired two-tailed t-tests with Bonferroni correction applied.

In addition to effectiveness, we measure efficiency in terms of query latency and model size. Query latency is recorded in a CPU-only environment using a consumer-grade Apple machine equipped with a 2.4GHz 8-core Intel Core i9 processor and 64GB of 2667MHz DDR4 memory. For each method, we randomly sample 100 queries from the MS MARCO development set, issue them to the system, and report the average latency per query. Finally, we record the size of each model in megabytes (MB) to quantify and compare memory requirements.

8.2.2 Baselines

BOW Retrievers

We report results for the widely used bag-of-words retrieval model BM25 ($k_1 = 0.9, b = 0.4$), both in its standard form and in combination with the RM3 (10 fbTerms, 10 fbDocs, 0.5 original query weight) Pseudo-Relevance Feedback method (BM25+RM3). These models serve as strong traditional baselines and are implemented using the Anserini toolkit [229], with all parameters set to their default values.

Dense Retrievers

Our TPRF model operates on top of the query and passage representations produced by dense retrievers. Therefore, in our experiments, we adopt three representative and publicly available dense retrieval models as base retrievers: ANCE [223], DistilBERT-Balanced [73], and TCTv2-HN+[115]. These models are used to generate the initial query embeddings and perform the first round of retrieval, which provides the feedback passage embeddings for TPRF training. For all three retrievers, we use

the original model checkpoints released by the authors and construct dense vector indexes using the FAISS library[41] to enable efficient nearest neighbor search.

ANCE-PRF and Variants

The original ANCE-PRF method [234] only considered ANCE as the base dense retrieval and to derive the encoder for the feedback signal. In Chapter 4 we extended the method to generalise it to other dense retrievers, following the ANCE-PRF training approach. Along with ANCE-PRF, we then also consider these model variants: we use the DistilBERT-PRF and TCTV2-PRF implementations and checkpoints provided by Li et al. [102]. This allows to compare the three considered dense retrievers and directly compare each to the PRF methods proposed by Yu et al. [234] and our TPRF on each of these dense retrievers. This allows to distinguish the effectiveness contribution of the dense retriever from that of the specific PRF mechanism.

8.2.3 TPRF Implementation and Training

The TPRF model is implemented using PyTorch and PyTorch Lightning, with its transformer architecture constructed using standard modules provided by the HuggingFace Transformers library [221]. To investigate the generalizability of TPRF across different dense retrieval backbones, we integrate it with three representative models: ANCE [223], DistilBERT-Balanced [73], and TCTv2-HN+ [115]. For each of these models, we use the original checkpoints released by the authors and directly extract the query and passage representations using their respective encoders. These dense encoders are not modified during the training of TPRF, as both the query encoder and passage encoder are frozen, hence no gradient updates are applied. TPRF is trained solely to manipulate these static dense representations. TPRF takes the original query and pseudo-relevant passage embeddings as 768-dimensional inputs. The model aggregates these into a single refined query vector of matching dimensionality, which is then used for a second-stage retrieval over the original dense index, ensuring full compatibility with the base infrastructure.

During training, TPRF is optimized using a supervised contrastive learning approach with a cross-entropy loss function as in Eq. 8.1. For each training query, we retrieve the top 200 passages from the fixed dense retriever and construct a set of training instances. Each instance consists of one positive passage and 20 negatives. The positive passage is sampled from the relevance annotations in the MS MARCO training set. The 20 negatives are drawn from the passages ranked between positions 10 and 200 in the initial retrieval list. This choice of sampling range helps to exclude easy non-relevant passages while still presenting the model with competitive distractors. In addition to these randomly sampled negatives, we experiment with incorporating hard negatives by selecting the two passages that are ranked immediately after the PRF depth k. For instance, when $k = 1$, the passages ranked at positions 2 and 3 are treated as additional hard negatives. These passages often exhibit lexical or semantic similarity to the relevant ones, therefore providing a more challenging learning signal. This approach follows the design choices made in Chapter 4 and Yu et al. [234]. We do not use

in-batch negatives, as prior work [234] suggests they may introduce noise under PRF-specific training conditions.

To reduce computational cost and training variance, we pre-compute the dense representations of all training queries and passages using the frozen dense retrievers. The top 200 retrieval results for each query are also generated in advance and stored locally. During training, the TPRF model loads these pre-computed embeddings and relevance labels without performing retrieval or encoding on-the-fly. This approach improves training efficiency and avoids repeated GPU computation of dense representations.

The TPRF model is configured with several architectural variants to explore trade-offs between model complexity and effectiveness. We experiment with transformer depths of 6, 8, 10, and 12 layers, and attention head counts of 4, 6, and 12. Across all configurations, the input and output dimensionality is fixed at 768 to maintain direct compatibility with the base retrievers (e.g., ANCE, DistilBERT), ensuring the refined query remains in the original vector space without requiring additional projection layers. For the feed-forward sub-layer, we set the hidden dimension to 1024. This reduced size was chosen to minimize parameter count and inference latency while retaining sufficient capacity to model non-linear interactions between feedback signals. Finally, we apply a dropout rate of 0.2, slightly higher than the standard 0.1 used in large language models, to provide stronger regularization, mitigating the risk of overfitting to specific query-passage vector patterns during training.

The training is conducted for 50 epochs using the AdamW optimizer with a fixed batch size of 512. We experiment with two learning rates, 1e-6 and 1e-5. Empirical results indicate that the higher learning rate yields more consistent improvements in retrieval effectiveness across configurations, and thus we adopt 1e-5 for all reported results.

All models are trained using a single NVIDIA Tesla SMX2 GPU with 32 GB of memory. Each training run corresponds to one TPRF configuration paired with a base retriever and requires approximately 26 hours to complete the 50 epochs. These training resources and durations are consistent across all three retriever backbones.

For comparison with existing PRF approaches, particularly ANCE-PRF, we evaluate both ANCE-PRF and our ANCE-TPRF implementation at a fixed PRF depth of $k = 3$. This setting aligns with the configuration used in the publicly released ANCE-PRF checkpoint, where it achieves optimal performance. Adopting this specific depth ensures a direct and controlled comparison between the two methods, allowing us to investigate the impact of the feedback architecture under the same operating conditions established for the baseline.

8.3 Results

8.3.1 Overall Effectiveness

Tables 8.1 and 8.2 report the retrieval effectiveness of various baseline and PRF methods on the TREC DL 2019 and TREC DL 2020 datasets. For each dense retriever, we select the best-performing TPRF

Dataset	TREC DL 2019					
Metric	MAP	R@1000	RR	nDCG@1	nDCG@3	nDCG@10
BM25	0.3013	0.7501	0.7036	0.5426	0.5230	0.5058
BM25+RM3	0.3390	0.7998	0.6683	0.5465	0.5195	0.5180
ANCE	0.3710	0.7554	0.8372	0.7209	0.6765	0.6452
ANCE-PRF	**0.4253**	**0.7912**	**0.8492**	**0.7326**	**0.6878**	**0.6807**
ANCE-VPRF-Rocchio	0.4211	0.7825	0.7870	0.6860	0.6622	0.6539
ANCE-TPRF	0.3942	0.7814†	0.8250	0.7209	0.6664	0.6648
DistilBERT	0.4590	0.8406	0.8765	0.7558	0.7494	0.7210
DistilBERT-PRF	0.4874	0.8565	0.8624	0.7636	**0.7717**	**0.7358**
DistilBERT-VPRF-Rocchio	**0.4974**	**0.8775**	0.8361	0.7364	0.7386	0.7231
DistilBERT-TPRF	0.4588	0.7696	**0.9632**†	**0.7791**	0.7535	0.7348
TCTV2	0.4469	0.8261	0.8869	**0.8023**	0.7410	0.7204
TCTV2-PRF	0.4784	0.8519	0.8725	0.7597	**0.7610**	0.7411
TCTV2-VPRF-Rocchio	**0.4883**	**0.8694**	0.8528	0.7248	0.7129	0.7111
TCTV2-TPRF	0.4669	0.7644	**0.9477**†	0.7791	0.7584	**0.7449**

Table 8.1: Retrieval effectiveness on TREC DL 2019. For the proposed TPRF, models are selected based on the best nDCG@10 on the validation set, as this is the primary official metric for the TREC Deep Learning benchmarks. The best results among each backbone dense retriever are marked with **Bold**. Significant improvements of TPRF over corresponding baselines are marked with †.

Dataset	TREC DL 2020					
Metric	MAP	R@1000	RR	nDCG@1	nDCG@3	nDCG@10
BM25	0.2856	0.7863	0.6585	0.5772	0.5021	0.4796
BM25+RM3	0.3019	0.8217	0.6360	0.5648	0.4740	0.4821
ANCE	0.4076	0.7764	0.7907	0.7346	0.7082	0.6458
ANCE-PRF	**0.4452**	**0.8148**	**0.8371**	**0.8025**	**0.7450**	**0.6948**
ANCE-VPRF-Rocchio	0.4341	0.7948	0.8079	0.7407	0.7117	0.6598
ANCE-TPRF	0.4234	0.7911	0.7733	0.7160	0.6953	0.6546
DistilBERT	0.4698	0.8727	0.8350	0.7593	0.7426	0.6854
DistilBERT-PRF	0.4836	0.8658	0.8525	0.7747	0.7459	0.6977
DistilBERT-VPRF-Rocchio	**0.4879**	**0.8926**	**0.8641**	**0.8056**	**0.7564**	**0.7083**
DistilBERT-TPRF	0.4800	0.8522	0.8425	0.7623	0.7513	0.7055
TCTV2	0.4754	0.8429	0.8392	0.7932	0.7199	0.6882
TCTV2-PRF	**0.4932**	0.8400	0.8509	0.8025	**0.7745**	**0.7115**
TCTV2-VPRF-Rocchio	0.4860	**0.8518**	0.8154	0.7685	0.7273	0.6804
TCTV2-TPRF	0.4708	0.8266	**0.8734**†	**0.8210**†	0.7683†	0.6991

Table 8.2: Retrieval effectiveness on TREC DL 2020. For the proposed TPRF, models are selected based on the best nDCG@10 on the validation set, as this is the primary official metric for the TREC Deep Learning benchmarks. The best results among each backbone dense retriever are marked with **Bold**. Significant improvements of TPRF over corresponding baselines are marked with †.

	TREC DL 2019						TREC DL 2020					
	MAP	R@1000	MRR	nDCG@1	nDCG@3	nDCG@10	MAP	R@1000	MRR	nDCG@1	nDCG@3	nDCG@10
[a] ANCE	0.3710	0.7554	0.8372	0.7209	0.6765	0.6452	0.4076	0.7764	0.7907	0.7346	0.7082	0.6458
[p] ANCE-PRF	**0.4253**$^{+a}$	**0.7912**$^{+a}$	**0.8492**	**0.7326**	**0.6878**	**0.6807**	**0.4452**$^{+a}$	**0.8148**$^{+a}$	**0.8371**$^{+a}$	**0.8025**$^{+a}$	**0.7450**$^{+a}$	**0.6948**$^{+a}$
[t] ANCE-TPRF	0.3942^{-p}	0.7814^{+a}	0.8250	0.7209	0.6664	0.6648	0.4234^{-p}	0.7911	0.7733^{-p}	0.7160^{-p}	0.6953^{-p}	0.6546^{-p}

Table 8.3: Effectiveness of dense PRF methods based on ANCE; bold values are the best for that metric and dataset. Statistical significant differences ($p < 0.05$) are indicated using the superscript character corresponding to each method. + means significantly better, − means significantly worse.

configuration based on validation nDCG@10. Unlike the Vector-based PRF methods in Chapter 5 which required analyzing sensitivity to α and β parameters via extensive plots, TPRF learns these weights automatically. Therefore, we report the best checkpoint performance here. A detailed analysis of the trade-offs between model size and effectiveness is provided in Section 8.3.2 and Figure 8.2.

TPRF demonstrates competitive performance on specific metrics, particularly at higher ranks (RR, nDCG@1, and nDCG@3). On TREC DL 2020 (Table 8.2), TCTV2-TPRF achieves the highest nDCG@1 (0.8210) among all TCTV2 variants, significantly outperforming the TCTV2 baseline (marked with †). It also achieves the highest Reciprocal Rank (0.8734), which is a statistically significant improvement over the baseline. Although its nDCG@10 (0.6991) is slightly lower than that of TCTV2-PRF (0.7115), it remains higher than the VPRF-Rocchio-based variant (0.6804). Across most configurations, TPRF outperforms the original dense retriever baselines, reflecting consistent improvements across different backbone models.

On TREC DL 2019 (Table 8.1), TCTV2-TPRF obtains the best nDCG@10 (0.7449) among all TCTV2-based methods, numerically outperforming both TCTV2-PRF (0.7411) and TCTV2-VPRF-Rocchio (0.7111). It provides a competitive nDCG@3 (0.7584), which is comparable to TCTV2-PRF (0.7610), and achieves a strong nDCG@1 of 0.7791. Notably, on this dataset, TCTV2-TPRF achieves a statistically significant improvement in Reciprocal Rank (0.9477) compared to the baseline (0.8869).

For DistilBERT, TPRF delivers competitive performance, particularly on TREC DL 2019 where DistilBERT-TPRF achieves the highest nDCG@1 (0.7791) and Reciprocal Rank (0.9632) among all DistilBERT variants. These improvements in RR and nDCG@1 are statistically significant compared to the DistilBERT baseline. However, it does not achieve the highest nDCG@10; its score of 0.7348 is slightly below DistilBERT-PRF (0.7358). On DL 2020, it records an nDCG@10 of 0.7055, which is marginally lower than DistilBERT-VPRF-Rocchio (0.7083). These results indicate that DistilBERT-TPRF is particularly effective at promoting the single most relevant document to the top rank, even if the overall top-10 ranking density is slightly lower than other PRF methods.

To provide a more detailed comparison, Table 8.3 focuses on the ANCE-based configurations. For ANCE-TPRF, the results are obtained using the configuration with 6 transformer layers and 12 attention heads. Across most metrics, ANCE-TPRF improves over the base ANCE model on TREC DL 2019. Specifically, it outperforms ANCE in MAP (+0.0232), R@1000 (+0.0260), and nDCG@10 (+0.0196). Notably, the improvement in R@1000 is statistically significant. In early precision metrics, it performs comparably to the baseline, matching nDCG@1 (0.7209) while achieving a slightly lower RR (0.8250 vs 0.8372).

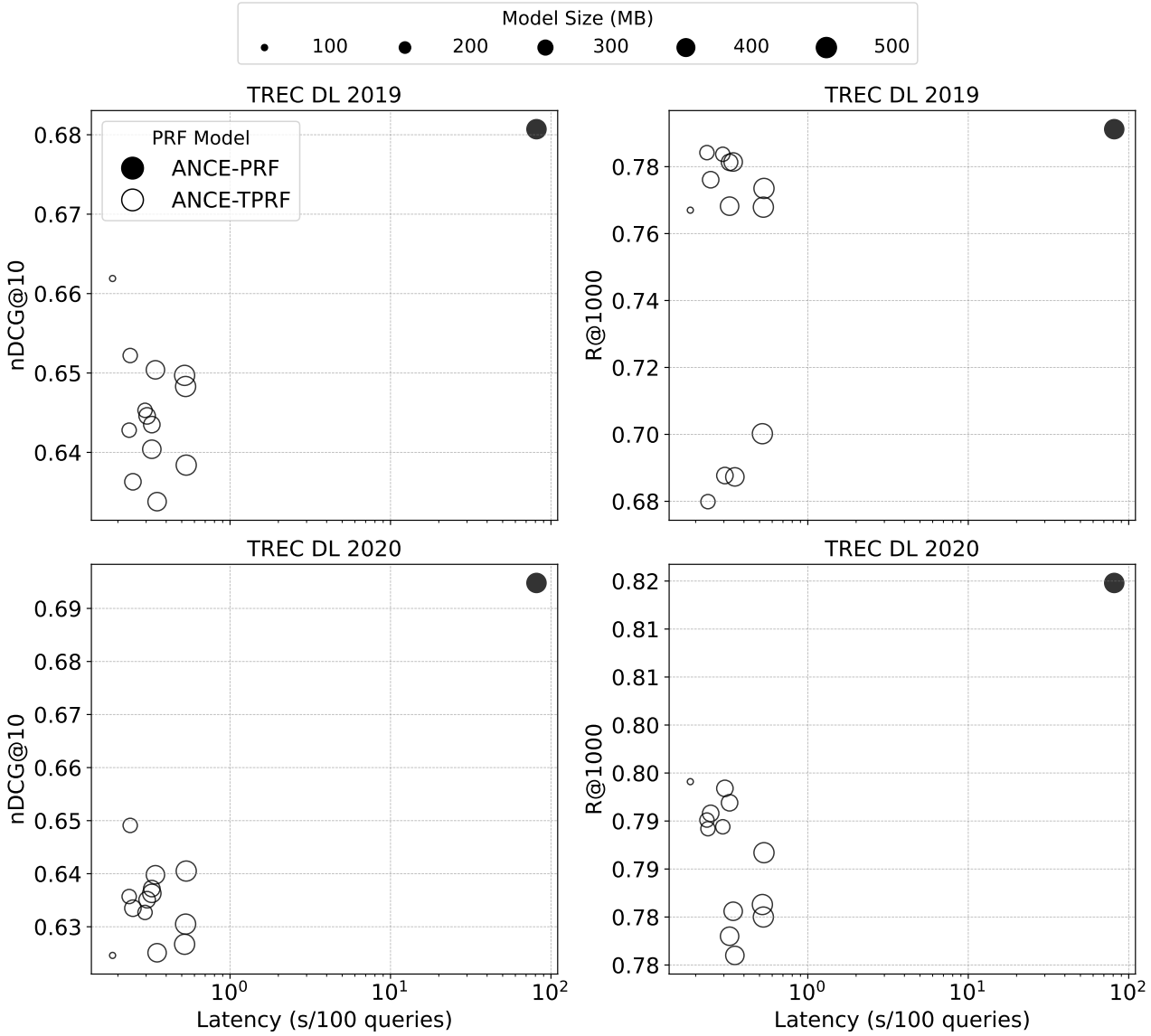


Figure 8.2: Relationship between effectiveness (measured as nDCG@10 and R@1000), query latency and model size for PRF methods, across datasets. For ANCE-TPRF, we display different configurations (w.r.t. number of layers and attention heads).

On a more challenging dataset TREC DL 2020, ANCE-TPRF again shows numerical improvements over ANCE in MAP (+0.0158), R@1000 (+0.0147), and nDCG@10 (+0.0088). However, it performs slightly worse than the baseline in early precision metrics such as RR (0.7733 vs 0.7907) and nDCG@1 (0.7160 vs 0.7346). Despite the lack of statistical significance in these specific ANCE comparisons, ANCE-TPRF still demonstrates consistent numerical benefits in recall-oriented metrics such as R@1000 and MAP without the overhead of large pre-trained PRF encoders. This positions TPRF as a competitive and more efficient alternative, particularly when computational resources are limited.

Model	Configuration	Size	Latency (ms/q)
ANCE-PRF	12 layers, 12 heads	503.4MB	937.21
ANCE-TPRF	12 layers, 12 heads	582.9MB	5.33
ANCE-TPRF	10 layers, 12 heads	488.6MB	4.37
ANCE-TPRF	8 layers, 12 heads	393.8MB	3.47
ANCE-TPRF	6 layers, 12 heads	299.2MB	2.35
ANCE-TPRF	1 layer, 1 head	62.7MB	1.85

Table 8.4: Latency comparison between ANCE-PRF and different configurations of ANCE-TPRF on CPU-only machine.

8.3.2 Evaluating the Trade-off Between Effectiveness, Model Size, and Query Latency

The motivation for proposing TPRF was to achieve a better balance between the effectiveness gains enabled by pseudo-relevance feedback and the efficiency demands required in resource-constrained environments, such as those lacking GPU access or limited in memory capacity. This subsection evaluates that trade-off in terms of retrieval effectiveness, model size, and query latency across different TPRF configurations. The analysis, addressing **RQ2.3.1: What is the trade-off between retrieval effectiveness, model size, and query latency across different TPRF configurations?** is presented in Figure 8.2, using the ANCE backbone as a case study. For each combination of transformer layers and attention heads, we report the best-performing checkpoint selected based on nDCG@10.

Regarding effectiveness, the variation across ANCE-TPRF model configurations is relatively small in terms of nDCG@10 on both DL 2019 and DL 2020. The differences between the best and worst performing models are not clearly separated for nDCG@10. However, the variance is more observable when measured using Recall@1000, particularly on DL 2019. In this case, the models exhibit a clear separation into two clusters, and the performance gap between the strongest and weakest models is more evident.

When compared with ANCE-PRF, ANCE-TPRF generally performs less effectively across evaluation metrics and datasets. However, over several metrics on both DL19 and DL20 datasets, ANCE-TPRF can approach ANCE-PRF's performances, suggesting that in some cases TPRF can approximate the effectiveness of ANCE-PRF.

These effectiveness differences come with a substantial efficiency gain. ANCE-TPRF exhibits extremely low query latency on a commodity CPU, with 100 queries taking less than one second to process. In contrast, ANCE-PRF requires nearly 100 seconds to process the same number of queries under the same conditions. This efficiency gain positions TPRF as a practical option for real-time or low-resource deployment scenarios.

Model size analysis further supports this trade-off. There is no clear correlation between model size and effectiveness among ANCE-TPRF configurations. The most effective models are not necessarily the largest. For instance, the best-performing model on DL 2020 in terms of nDCG@10 (0.6546) has a size of 299.2 MB, whereas the largest model (582.9 MB) yields a substantially lower nDCG@10

(0.5581). This observation suggests that increasing model complexity does not necessarily yield effectiveness gains in the TPRF setting.

Compared to ANCE-PRF, which has a model size of 503.4 MB, several ANCE-TPRF models achieve comparable or better recall and precision while using less memory. The most effective ANCE-TPRF model on DL 2019, based on both nDCG@10 and Recall@1000, is only 299.2 MB. Similarly, the strongest performing configuration for DL 2020 achieves 0.6546 nDCG@10 with the same model size and a Recall@1000 of 0.7911 using a slightly larger configuration of 393.8 MB.

The model size in TPRF is primarily determined by the number of transformer layers rather than the number of attention heads. All models with 6 layers, regardless of the number of attention heads (4, 6, or 12), result in the same size of 299.2 MB. Additionally, these smaller models also exhibit the lowest latency. For example, models of 299.2 MB require only 0.235 seconds to process 100 queries, while the largest model takes 0.533 seconds. Although this difference is measurable, it is negligible when compared to the latency of ANCE-PRF, as shown in Table 8.4.

An additional configuration with a minimal architecture, which only has 1 transformer layer and 1 attention head, was not included in the main comparison plot but still deserves discussion. This model is the smallest among all evaluated (62.7 MB) and has the fastest latency (0.185 seconds per 100 queries). Despite its simplicity, it is still able to improve upon the base ANCE retriever in terms of MAP and Recall@1000 on both datasets.

Overall, ANCE-PRF continues to provide the highest effectiveness across most metrics. However, TPRF offers a flexible and efficient alternative, enabling substantial reductions in query latency and model size while still delivering consistent improvements over the base dense retrievers. This trade-off makes TPRF particularly well-suited for deployment in memory- and latency-constrained environments.

8.3.3 Scalability of TPRF Latency with Respect to PRF Depth

This subsection addresses **RQ2.3.2: How does TPRF scale in terms of query latency as PRF depth increases?** We investigate how the inference time of TPRF varies as the number of feedback passages grows, and compare it with several neural PRF baselines including ANCE-PRF, DistilBERT-PRF, and TCTV2-PRF (Chapter 4).

It is important to note that this latency analysis is a stress test of the architecture's computational throughput. For this scalability experiment, we simulate inputs of varying depths (k) to measure inference speed. This differs from the effectiveness evaluation in Section 8.3.1, where models were trained and evaluated at a fixed depth of $k = 3$ to ensure a controlled comparison with the ANCE-PRF baseline. While training distinct TPRF models optimized for larger PRF depths is prohibitive due to training resource constraints (26 GPU-hours per model per k), this latency sweep demonstrates that the architecture itself remains efficient even at extreme feedback depths.

In methods like ANCE-PRF, DistilBERT-PRF, and TCTV2-PRF, the query encoder takes as input a sequence formed by concatenating the query text with the full text of the top-k retrieved feedback

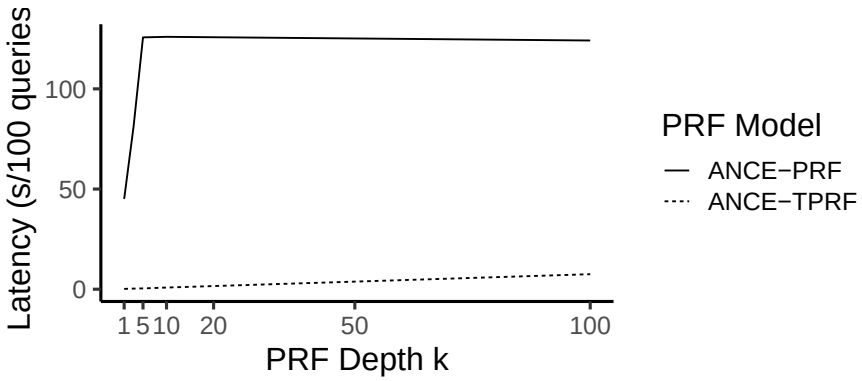


Figure 8.3: Query latency for ANCE-PRF and ANCE-TPRF as a function of the PRF depth k.

passages. As k increases, input length grows accordingly, leading to substantial increases in inference time due to the quadratic time complexity of self-attention in transformer models [34, 205]. This growth in latency continues until the model's maximum input token limit is reached. For RoBERTa-based encoders, this limit is 512 tokens, which is typically reached at around $k = 5$. Beyond this point, the additional feedback content is truncated, and latency plateaus. In our CPU-based experiments, ANCE-PRF reaches peak latency at approximately 125 seconds per 100 queries at $k = 5$, as shown in Figure 8.3.

By contrast, TPRF does not process raw text as input. Instead, it operates on fixed-size dense vectors produced by the dense retriever, and stacks the query vector with the top-k feedback passage vectors to form an input matrix. As PRF depth increases, the number of rows in this matrix increases, but its dimensionality remains fixed. Since transformer operations in TPRF are applied over vector representations, not token sequences, the self-attention computation remains constant in depth and structure. The only additional overhead comes from assembling the input matrix, which grows with k but incurs negligible cost.

This difference in architecture leads to significantly improved scalability for TPRF. As visualized in Figure 8.3, TPRF maintains low and predictable latency even when PRF depth increases to $k = 100$. In fact, the latency of TPRF at $k = 100$ remains lower than that of ANCE-PRF with as few as $k = 10$ feedback passages.

This latency advantage holds consistently across dense retriever backbones. Since TPRF does not reuse or re-encode the original query using a large pre-trained encoder, and its architecture is independent of the retriever's own transformer layers, its efficiency is unaffected by the underlying model. This makes TPRF a portable solution with uniform latency characteristics across multiple retriever types.

While TPRF may sacrifice a small amount of effectiveness relative to the most optimized PRF variants, this is balanced by substantial efficiency gains in both inference time and memory usage. Its ability to scale across PRF depth without incurring heavy latency penalties makes it a strong candidate for retrieval systems deployed in real-time or resource-constrained settings.

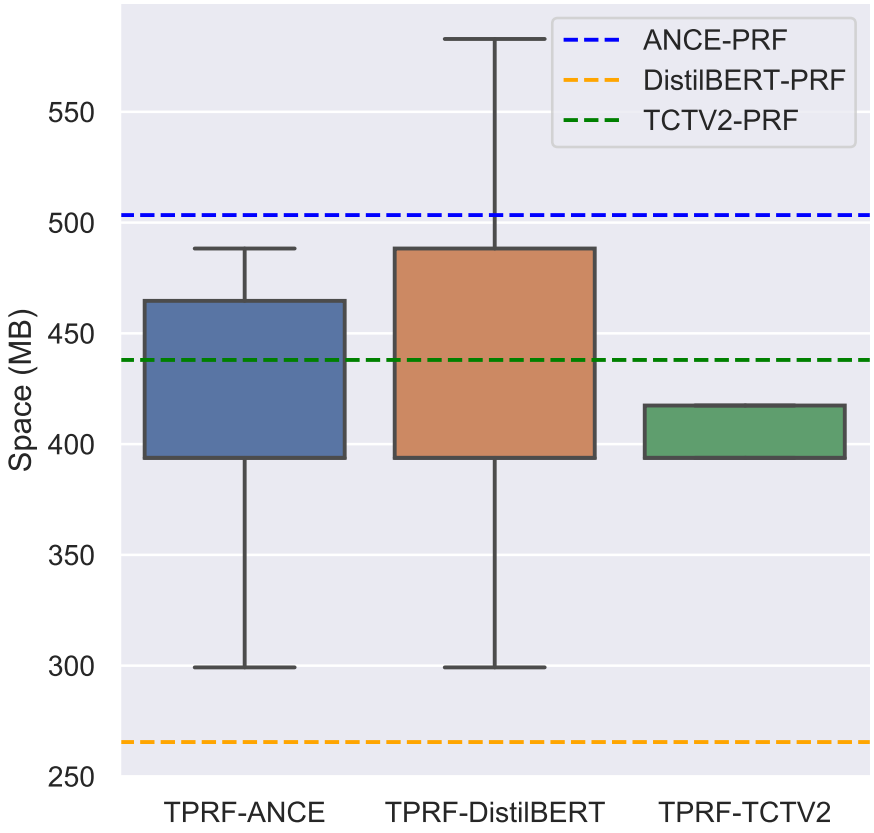


Figure 8.4: Model sizes for TPRF with different backbone models and associated trained PRF variations.

8.3.4 Comparison of Model Size Across PRF Methods and Its Impact on Deployability

This subsection addresses **RQ2.3.3: How does the model size of TPRF compare to existing PRF approaches, and what are the implications for deployability?** To answer this question, we compare the model sizes of TPRF with those of existing PRF methods trained on the same dense retriever backbones. Figure 8.4 provides a summary of the size distributions of TPRF models instantiated on ANCE, DistilBERT, and TCTV2, along with horizontal lines indicating the corresponding model sizes of ANCE-PRF, DistilBERT-PRF, and TCTV2-PRF, respectively. All models utilise a hidden dimension of 1024 and 12 attention heads, varying only in depth {6, 8, 10, or 12} layers.

All TPRF models represented in the figure are trained transformer-based query encoders, where model size is determined by architectural parameters, specifically the number of transformer layers and attention heads. Unlike prior work such as ANCE-PRF (Chapter 4) [234], which fine-tunes a dense retriever's original query encoder and therefore inherits its full architecture and pre-trained parameters, TPRF uses a lightweight transformer initialized from scratch. This design allows TPRF to scale model size flexibly according to computational and memory constraints, independent of the underlying dense retriever.

The flexibility to scale TPRF according to hardware requirements is critical for deployability, though the specific memory advantages depend on the baseline architecture. As shown in Figure 8.4,

for standard BERT-based backbones like ANCE and TCTV2, TPRF configurations are consistently smaller than their corresponding neural PRF baselines. Specifically, the majority of ANCE-TPRF configurations fall well below the ANCE-PRF baseline ($\approx$ 503 MB), with the smallest models nearing 300 MB, and TCTV2-TPRF remains consistently below the TCTV2-PRF threshold ($\approx$ 450 MB). In contrast, for the highly compressed DistilBERT model, all evaluated TPRF configurations exceed the size of the lightweight DistilBERT-PRF baseline ($\approx$ 265 MB). This outcome highlights a structural trade-off: since TPRF operates by adding a transformer stack on top of fixed embeddings, it inherently introduces parameter overhead. While this overhead is negligible compared to full BERT-based encoders, it outweighs the minimal footprint of an already distilled model like DistilBERT. Therefore, while TPRF offers architectural tunability as evidenced by the vertical spread of the boxplots, its primary memory advantage is achieved when replacing heavy feedback encoders rather than those based on already distilled architectures.

Importantly, smaller TPRF models do not mean weaker retrieval performance. Even the smallest configurations, such as the 299.2 MB models, consistently outperform the base dense retrievers over multiple metrics across datasets. These configurations trade a marginal decrease in effectiveness relative to ANCE-PRF for significant gains in model compactness and lower latency, offering practical benefits in constrained environments.

In response to **RQ2.3.3**, we conclude that TPRF achieves a favorable trade-off between model size and retrieval effectiveness. It offers clear memory savings over standard PRF approaches like ANCE-PRF and TCTV2-PRF, with configurable architectures that adapt to deployment requirements. While this trade-off may result in slight effectiveness loss compared to fully fine-tuned PRF encoders, the reduction in model size and corresponding gains in efficiency make TPRF highly suitable for memory-constrained or latency-sensitive settings.

8.3.5 Effectiveness Comparison of TPRF Against Neural and Vector-Based PRF Methods

This subsection addresses **RQ2.3.4: How does the retrieval effectiveness of TPRF compare to ANCE-PRF and other Vector-Based PRF baselines across different dense retrievers?** We analyze this by comparing the effectiveness of TPRF against multiple baselines, including dense retrievers without feedback, their corresponding neural PRF variants (e.g., ANCE-PRF, DistilBERT-PRF, TCTV2-PRF), and traditional vector-based feedback approaches such as VPRF-Rocchio and BM25+RM3. Figure 8.5 and Figure 8.6 present boxplots that capture the distribution of performance for all TPRF configurations across evaluation metrics on TREC DL 2019 and 2020, respectively. Each box summarizes the range, interquartile spread, and peak performance (triangle marker) for TPRF models trained with various parameter settings (i.e., different combinations of transformer layers and attention heads).

Across both datasets, TPRF shows a consistent ability to outperform the base dense retrievers across nearly all metrics. This is reflected in the fact that the median and maximum values of most TPRF configurations lie above the corresponding dense retriever baselines. For instance, on DL 2020,

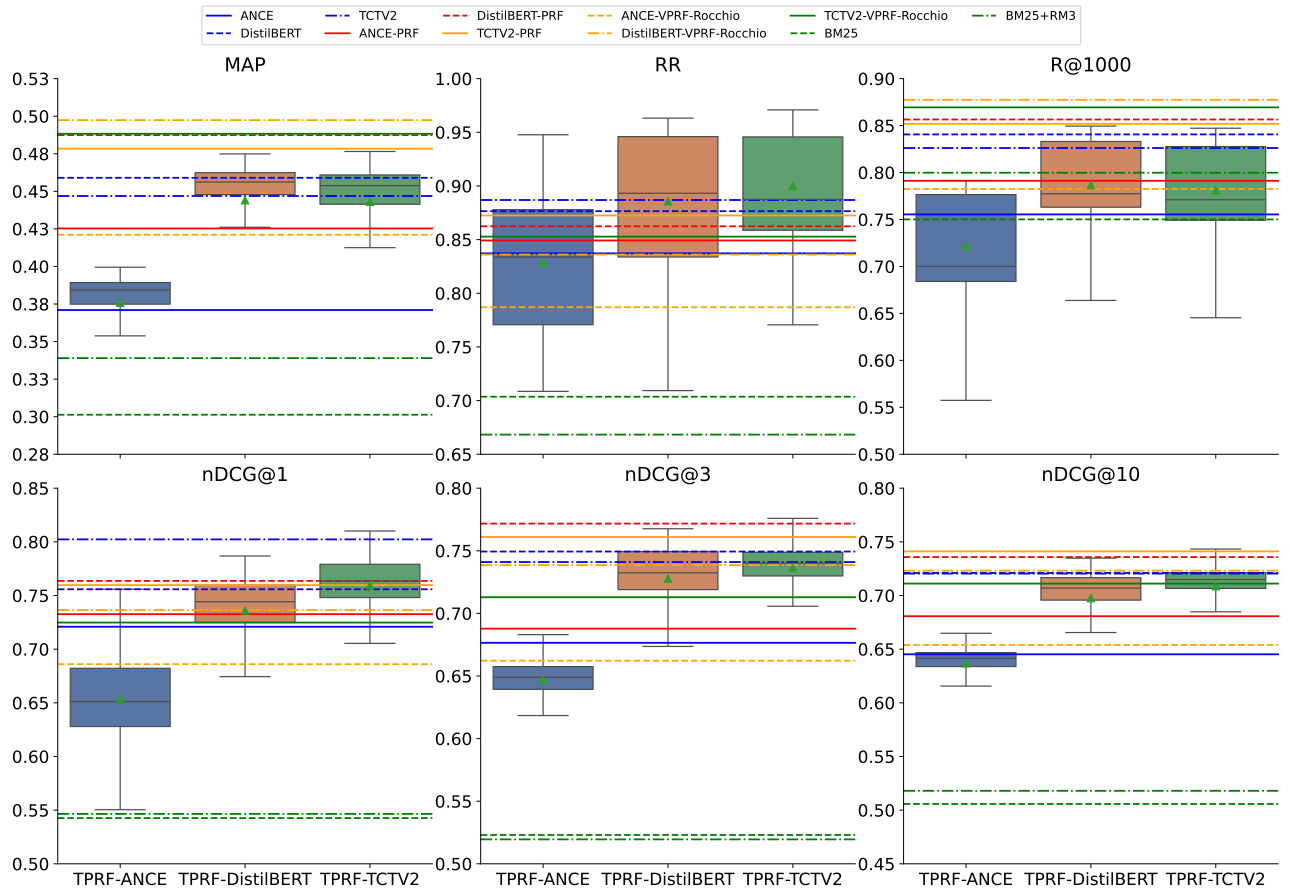


Figure 8.5: Variance in effective of TPRF models based on training with different parameter settings on TREC DL 2019. Each boxplot represents the distribution of the maximum result per evaluation metric across all the training parameters.

TCTV2-TPRF models outperform the original TCTV2 retriever in terms of MAP, nDCG@10, and nDCG@3, while also matching or exceeding TCTV2-PRF and VPRF-Rocchio variants. Similar patterns are observed with DistilBERT-TPRF, which achieves higher nDCG@10 and RR values than DistilBERT-PRF and its VPRF-Rocchio-based counterpart. These trends are particularly strong for nDCG metrics and RR, suggesting that TPRF is effective at improving early precision.

ANCE-TPRF is comparatively less competitive than its counterparts based on DistilBERT and TCTV2. On DL 2019, ANCE-TPRF shows wider variability in performance, especially in RR and Recall@1000, where some configurations fall below the base retriever or baseline PRF models. However, ANCE-TPRF still reaches competitive performance at its best configurations, especially for Recall@1000, where it outperforms ANCE-PRF. On DL 2020, ANCE-TPRF exhibits a narrower range but generally falls below ANCE-PRF on most nDCG and MAP metrics, suggesting that ANCE-PRF maintains a stronger advantage in that setting.

Importantly, while neural PRF models like ANCE-PRF and DistilBERT-PRF are designed around pre-trained encoders and benefit from textual re-encoding, TPRF operates entirely on dense vector representations. Despite this, TPRF still performs competitively with these approaches and often surpasses them in several ranking metrics. For example, DistilBERT-TPRF and TCTV2-TPRF show nDCG@10 distributions that exceed those of their DistilBERT-PRF and TCTV2-PRF variants,

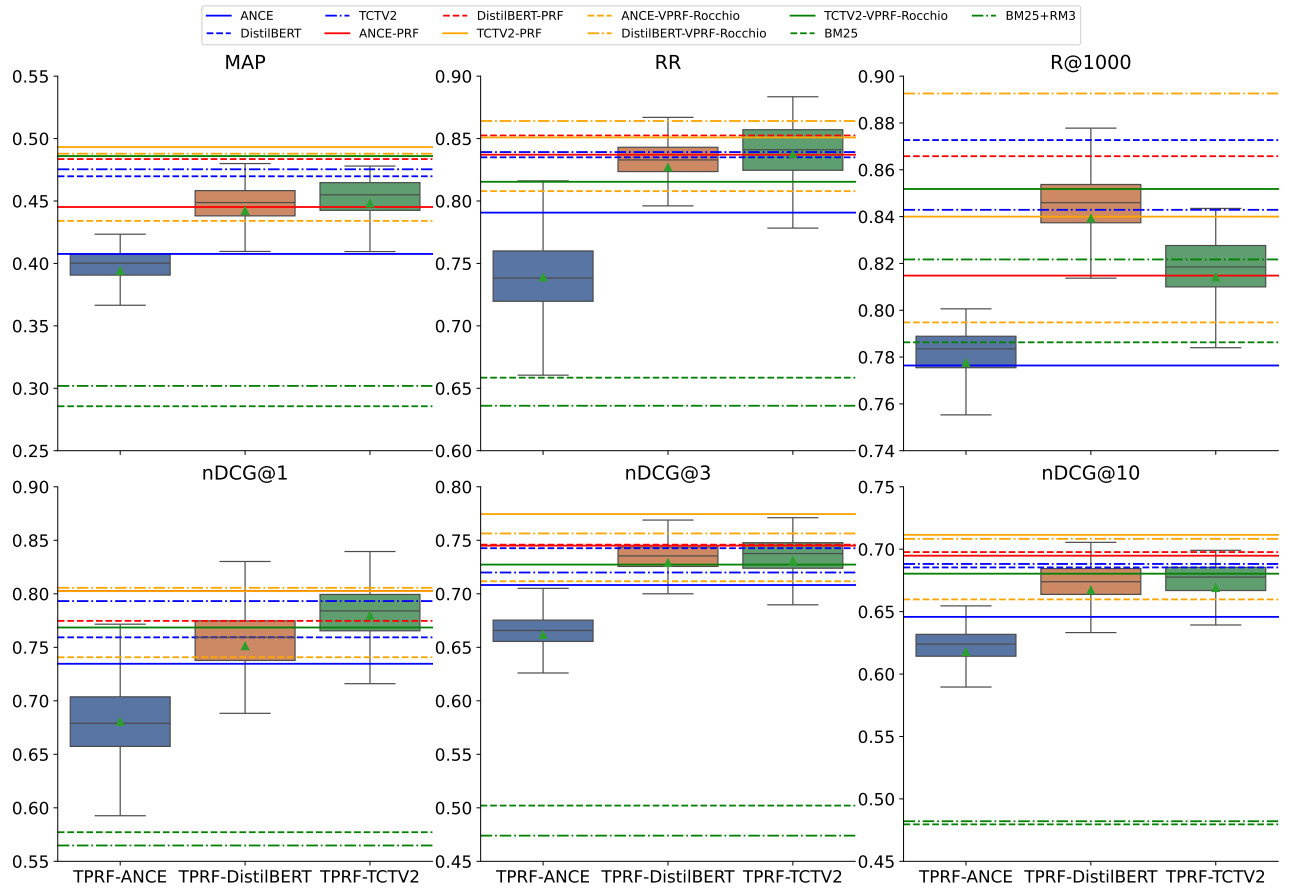


Figure 8.6: Variance in effective of TPRF models based on training with different parameter settings on TREC DL 2020. Each boxplot represents the distribution of the maximum result per evaluation metric across all the training parameters.

particularly in DL 2020, where their performance ranges are tightly concentrated around or above the highest baselines.

Traditional sparse PRF baselines such as BM25+RM3 serve as additional comparison points. While BM25+RM3 is strong on recall, particularly for TREC DL 2019, TPRF methods consistently outperform it in all other metrics, including MAP, RR, and nDCG-based measures. This reinforces the observation that even compact TPRF models provide a superior balance of precision and ranking effectiveness.

Lastly, TPRF with DistilBERT and TCTV2 consistently achieves performance on par with or better than their respective neural PRF and VPRF baselines in early precision metrics (RR, nDCG@1) as shown in Tables 8.1 and 8.2. However, for deeper metrics such as MAP and Recall@1000, VPRF-Rocchio often retains a slight advantage, particularly on TREC DL 2019. While ANCE-TPRF does not consistently surpass ANCE-PRF in absolute terms, it remains competitive in recall and consistently outperforms the dense retriever baseline.

In summary, TPRF achieves strong retrieval effectiveness across multiple retrievers and datasets. It consistently improves over base dense retrievers and frequently matches or exceeds neural PRF baselines on high-precision ranking metrics. While vector-based methods like VPRF-Rocchio remain highly competitive for recall-oriented tasks, TPRF outperforms traditional feedback methods such

as BM25+RM3 and offers a distinct advantage in early ranking precision. These results validate the effectiveness of TPRF as a model-agnostic feedback mechanism that operates efficiently while delivering competitive or superior ranking performance.

8.4 Summary

This chapter explored the feasibility of designing a Pseudo-Relevance Feedback (PRF) method that is both effective and efficient, addressing the high level research question **RQ2.3: Can Pseudo-Relevance Feedback be both efficient and effective?** To answer this, we proposed TPRF, a lightweight feedback model based on a vanilla transformer architecture, and systematically evaluated its performance in terms of retrieval effectiveness, inference latency, scalability to PRF depth, and model size.

Our results show that TPRF improves retrieval performance over baseline dense retrievers across various configurations and datasets. With models such as DistilBERT and TCTV2, TPRF achieves comparable effectiveness to existing neural PRF approaches while using a smaller model footprint and incurring substantially lower query latency. When compared with vector-based PRF methods like VPRF-Rocchio, TPRF demonstrates the ability to adaptively weigh feedback signals without relying on manually tuned parameters, leading to consistent gains, especially in early precision metrics such as RR, nDCG@1, and nDCG@3.

On the efficiency aspect, TPRF exhibits highly predictable and scalable latency characteristics. Its inference time remains stable even as the number of feedback passages increases to $k = 100$, a depth at which traditional models like ANCE-PRF become prohibitively slow or hit input size limits. TPRF also offers significant flexibility in model size through configurable layer and head parameters, allowing users to make informed trade-offs between speed, memory usage, and effectiveness based on deployment constraints.

These findings affirm that PRF can be both efficient and effective. TPRF represents a step forward in achieving this balance, offering a practical and adaptable solution suitable for diverse retrieval settings, particularly those with constrained computational resources.

Building on this, the next chapter shifts focus toward the practical engineering required to deploy these innovations. We introduce a modular PRF framework integrated into the Pyserini toolkit. Using our Vector-based PRF (VPRF) strategies (Chapter 5) as a foundational case study, we demonstrate how to seamlessly incorporate dense feedback mechanisms into a production-grade retrieval system. More importantly, this implementation serves as a generalized blueprint for the broader family of feedback methods explored in this thesis: by establishing a unified interface for vector manipulation, we pave the way for the similar integration of advanced architectures like TPRF, as well as LLM-driven approaches like LLM-VPRF (Chapter 10) and PromptPRF (Chapter 11). This framework enables streamlined experimentation and reproducibility across the entire spectrum of dense feedback techniques, allowing researchers to better understand their trade-offs within large-scale retrieval systems.

Chapter 9

Pseudo-Relevance Feedback Framework Development and Evaluation

This chapter addresses the final high-level research question of Part 2:

RQ2.4: How to design an extensible Pseudo-Relevance Feedback framework?

While previous chapters in this part examined the impact of signal quality (Chapter 7) and efficient lightweight model design for PRF (Chapter 8), the focus here shifts to infrastructure: specifically, the design of a scalable software environment capable of supporting diverse feedback strategies.

To this end, we introduce a modular PRF framework implemented within the Pyserini IR toolkit [112], a widely adopted Python interface built on top of Anserini [229]. Pyserini provides support for multiple dense retrievers and facilitates dense vector search via a standardized API using FAISS [41] indexes. Leveraging this modularity, we design a PRF framework that is agnostic to both the underlying dense retriever and the specific feedback logic.

More importantly, we present this framework not merely as an implementation of a single method, but as a generalized blueprint for integrating dense feedback mechanisms. While we utilize the Vector-based PRF (VPRF) strategies from Chapter 5 as a foundational case study to demonstrate the integration process, the architecture is intended to support the broader spectrum of methods explored in this thesis. By establishing a unified interface for vector manipulation and feedback injection, this framework paves the way for the seamless integration of advanced architectures like TPRF (Chapter 8), generative approaches such as LLM-VPRF (Chapter 10), and prompt-based strategies like PromptPRF (Chapter 11).

To demonstrate and validate the utility of this framework, we detail the integration of VPRF-Average and VPRF-Rocchio as a concrete working example. We verify that our implementation replicates the effectiveness reported in the original work and further extend the analysis to additional retrievers and datasets available through Pyserini. Our experiments show that the module consistently improves the retrieval effectiveness of all tested dense retrievers, confirming the robustness of the implementation.

The modular design ensures extensibility: additional PRF techniques can be implemented following the VPRF example, and new retrievers integrated into Pyserini immediately benefit from the existing feedback capabilities. In this chapter, we describe the architecture of the framework, its components, and usage, followed by a systematic empirical evaluation. Through this contribution, we aim to provide the community with a shared and extensible platform for advancing research on PRF in neural IR.

9.1 Integrating Vector-Based Feedback Logic

To demonstrate the framework's capability to host diverse feedback mechanisms, we integrate the two foundational VPRF strategies established in Chapter 5: **VPRF-Average** and **VPRF-Rocchio**. These methods serve as ideal test cases for the infrastructure: VPRF-Average validates the basic vector aggregation pipeline, while VPRF-Rocchio tests the framework's ability to handle user-configurable hyperparameters during the query refinement process.

VPRF-Average serves as the baseline implementation for the framework. By treating the original query and the top-k feedback passages as equally informative, this method validates the core system logic: retrieving feedback vectors and aggregating them with the query, without the complexity of hyperparameter tuning. Readers are referred to Section 5.1 in Chapter 5 for the formal mathematical definition.

VPRF-Rocchio is integrated to demonstrate the framework's support for user-configurable feedback logic. Unlike the simple averaging approach, this implementation exposes two tunable scalars, α and β, allowing users to control the weighting of the original query versus the feedback signal directly through the Pyserini API. Implementing this method confirms that the framework can flexibly accommodate strategies that require external parameter configuration, paving the way for the integration of more complex mechanisms like TPRF, LLM-VPRF, or PromptPRF in future extensions.

Both PRF methods are implemented as reusable modules in Pyserini [112] and can be directly applied to any dense retriever model whose passage vectors are accessible through the framework. For full evaluation details of the VPRF methods, readers are referred to Chapter 5.

9.2 Dense Retriever-Based Pseudo-Relevance Feedback in Pyserini

Figure 9.1 illustrates the architecture of our proposed PRF framework for Dense Retrievers implemented in Pyserini. The framework follows a two-stage retrieval process: the initial stage retrieves candidate passages using the original query, and the second stage refines the query representation based on these top-ranked results to perform a final retrieval. This design enables flexible and modular integration of PRF techniques into dense retrieval pipelines.

The framework is composed of three core components: `QueryEncoder`, `SimpleDenseSearcher`, and `DenseVectorPrf`. The `QueryEncoder` encodes the raw query text into a dense embedding using a

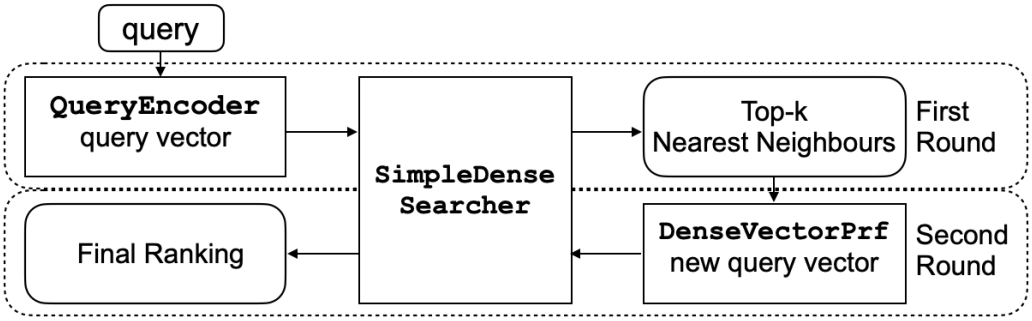


Figure 9.1: The proposed Pyserini Dense Retriever-based Pseudo-Relevance Feedback framework.

specified Dense Retriever model. The `SimpleDenseSearcher` then retrieves the top-*k* nearest passages from the dense index using vector similarity. These top-*k* results are passed to the `DenseVectorPrf` module, which constructs an updated query vector using one of the supported vector-based PRF methods. The revised query vector is then used for a second round of retrieval with the same searcher, providing a final ranked list.

All three components are implemented as abstract classes, allowing seamless extension to additional dense retrievers or PRF strategies. This modularity makes the framework broadly applicable and easy to adapt for future developments in dense retrieval. In the following sections, we describe the implementation details of each component and provide examples of how Dense Retriever-based PRF can be executed using Pyserini.

9.2.1 Modules and Configuration

In the following, we provide detailed descriptions of the implementation of each core component in the PRF framework, outlining their roles, design structure, and how they interact within the Pyserini pipeline.

`QueryEncoder`. The `QueryEncoder` module is an abstract class in Pyserini, which is responsible for encoding the raw query text into vectors for later retrieval of PRF candidates and generation of the new query vector. Currently the `QueryEncoder` module supports encoding with ANCE [223], DPR [83], TCT-ColBERT V1 [114], TCT-ColBERT V2 [115], DistillBERT KD [72], DistillBERT Balanced [73], and SBERT [173]. Different dense retrievers can be selected either programmatically or via command line arguments.

`SimpleDenseSearcher`. The `SimpleDenseSearcher` module is also an abstract class in Pyserini, which, in the first round of retrieval, is responsible for retrieving the top-*k* (PRF depth) candidates to be used as PRF signal. To do this, it uses the query encoded by `QueryEncoder`. In the second round of retrieval, this module is responsible for retrieving the final results using the new query generated by the `DenseVectorPrf` module. `SimpleDenseSearcher` requires a FAISS [41] index as one of the inputs and it performs the pairwise dot product between the query and document representations, for each passage in the index. This module can also work with any of the dense retrievers already supported by Pyserini.

Code Sample 9.1: VPRF-Average implementation in Pyserini.

```
# get all the PRF candidate vectors
all_candidate_embs = [item.vectors for item in prf_candidates]

# stack query vector and PRF candidate vectors
# then get mean as new query vector
new_emb_qs = np.mean(
np.vstack((emb_qs[0], all_candidate_embs)), axis=0)

# return new query vector as numpy array
new_emb_qs = np.array([new_emb_qs]).astype('float32')
```

Code Sample 9.2: VPRF-Rocchio implementation in Pyserini.

```
# get all the PRF candidate vectors
all_candidate_embs = [item.vectors for item in prf_candidates]

# get weighted mean of candidate vectors
weighted_mean_doc_embs = rocchio_beta * np.mean(
all_candidate_embs, axis=0)

# get weighted query vector
weighted_query_embs = rocchio_alpha * emb_qs[0]

# get sum of the weighted vectors
new_emb_q = np.sum(
np.vstack((weighted_query_embs, weighted_mean_doc_embs)), axis=0)

# return new query vector as numpy array
new_emb_q = np.array([new_emb_q]).astype('float32')
```

`DenseVectorPrf`**.** The `DenseVectorPrf` module, also an abstract class, is responsible for performing the actual PRF process and generating the new query vector representation. Currently, the VPRF-Average and VPRF-Rocchio approaches from Chapter 5 [104] are supported by the `DenseVectorPrf` module. For the Average approach (Eq. 5.1), we first stack the query vector and the PRF candidate vectors. Then we compute the average of the stacked vectors and obtain the final vector representing the new PRF query, as shown in Code Sample 9.1. For Rocchio, we first stack the PRF candidate vectors and compute their mean. Then we linearly interpolate the query vector and the average PRF candidate vector, using the interpolation parameters α and β (Eq. 5.2), as shown in Code Sample 9.2.

9.2.2 Usage

Our PRF module is designed for ease of use and integration. Code Sample 9.3 demonstrates how to run the VPRF-Average method on top of ANCE using batch mode, while Code Sample 9.4 illustrates the VPRF-Rocchio method under the same setup. Both methods can be executed via the `dsearch` command, which accepts the following key arguments:

- `--topics`: the path to the raw query text file;
- `--index`: the path to the pre-built Faiss index;

Code Sample 9.3: Usage of Average PRF in Pyserini with batch processing.

```
$ python -m pyserini.dsearch --topics dl19-passage \
    --index msmarco-passage-ance-bf \
    --encoder castorini/ance-msmarco-passage \
    --output test_ance_avg_prf3_batch.res \
    --batch-size 64 \
    --threads 12 \
    --prf-depth 3 \
    --prf-method avg
```

- `--encoder`: the dense retriever's name, or the path to a local model checkpoint;
- `--output`: the path to where the result file should be saved;
- `--batch-size`: the number of queries to be batched for group processing;
- `--threads`: the maximum number of threads to use;
- `--prf-depth`: the number of candidate vectors for PRF;
- `--prf-method`: specifies the PRF method, which currently can either be `avg` or `rocchio`;
- `--rocchio-alpha`: the value of the parameter α in VPRF-Rocchio (default: $\alpha = 0.9$);
- `--rocchio-beta`: the value of the parameter β in VPRF-Rocchio (default: $\beta = 0.1$);

The execution mode is controlled by the values assigned to `--batch-size` and `--threads`. When both parameters are set to 1, the system processes queries sequentially, which is suitable for small-scale evaluation or debugging. When `--batch-size` is greater than 1, the PRF module switches to batch mode, enabling multiple queries to be processed concurrently in a single inference pass. This setting is particularly useful for improving efficiency in large-scale retrieval scenarios, especially when running on GPU-enabled environments. The `--threads` option controls the number of parallel CPU threads for preprocessing and file I/O operations.

For the VPRF-Average method (see Code Sample 9.3), the key parameters are `--prf-depth 3` and `--prf-method avg`. The former controls how many top-ranked passages from the first retrieval stage are used to compute the new query vector, and the latter sets the PRF strategy to `avg`. This method is lightweight and requires minimal configuration.

In contrast, the VPRF-Rocchio method (see Code Sample 9.4) requires two additional hyperparameters: `--rocchio-alpha 0.9` and `--rocchio-beta 0.1`, which define the interpolation weights for the original query vector and the aggregated feedback vectors, respectively. The default values are $\alpha = 0.9$ and $\beta = 0.1$, reflecting a conservative adjustment toward the feedback signal while preserving most of the original query semantics. These parameters can be tuned based on task-specific requirements or dataset characteristics.

A key strength of our PRF integration is the seamless support for popular datasets. Pyserini operates with pre-built dense indexes, raw queries, and encoded queries for datasets such as MS MARCO,

Code Sample 9.4: Usage of Rocchio PRF in Pyserini with batch processing.

```
$ python -m pyserini.dsearch --topics dl19-passage \
    --index msmarco-passage-ance-bf \
    --encoder castorini/ance-msmarco-passage \
    --output test_ance_rocchio_prf5_batch.res \
    --batch-size 64 \
    --threads 12 \
    --prf-depth 5 \
    --prf-method rocchio \
    --rocchio-alpha 0.4 \
    --rocchio-beta 0.6
```

TREC DL, and more. When using these datasets, Pyserini automatically downloads and configures the necessary resources, requiring only the user to specify the dataset identifier via command-line arguments. This significantly reduces the overhead of setting up retrieval pipelines and facilitates quick and consistent reproduction of prior experiments.

In summary, the PRF module enables straightforward experimentation with vector-based feedback methods by exposing a minimal yet flexible interface. Once the appropriate parameters are configured, PRF-enhanced retrieval can be launched with a single terminal command, as demonstrated in Code Samples 9.3 and 9.4, without requiring manual index construction or additional setup.

9.3 Empirical Evaluation

As a demonstration of PRF in Pyserini and to address the **RQ2.4: How to design an extensible Pseudo-Relevance Feedback framework?** we conduct experiments to reproduce the results from Chapter 5 to show that our new feature in Pyserini can easily reproduce the results and can be applied to all existing dense retrievers already in Pyserini. Therefore, in this section, we first replicate the VPRF experiments using our Pyserini setup, confirming the results and presenting extra results with four additional models provided by Pyserini on the same datasets.

9.3.1 Experimental Setup

To verify the correct implementation of dense retriever-based PRF methods in Pyserini, we replicate the VPRF experiments from Chapter 5 on the TREC DL 2019 [27] and TREC DL 2020 [26] collections. Both benchmarks are derived from the MS MARCO Passage Ranking dataset [151]. Additionally, for completeness purpose, we also evaluate the PRF methods on the MS MARCO dev queries and report the results in Table 9.2. As expected, the PRF methods fail to show effectiveness gains in this setting and frequently lead to measurable losses, in contrast to the improvements observed on the TREC DL collections (Table 9.1). The TREC DL benchmarks use the same MS MARCO passage corpus and similarly sourced queries [26, 27], but benefit from deeper and more comprehensive relevance judgments, making them more suitable for PRF evaluation. Statistical significance is assessed using a two-tailed paired t-test with Bonferroni correction.

Our PRF methods do not require any model training and can be applied directly to a wide range of dense retrievers without modification. In our experiments, we consider several dense retrieval models readily supported by Pyserini: ANCE [223], TCT-ColBERT V1 [114], TCT-ColBERT V2 [115], DistillBERT KD [72], DistillBERT Balanced [73], SBERT [173], and ADORE [239]. Among these, only ANCE was evaluated in the original study in Chapter 5. Our framework extends the applicability of their proposed PRF methods to a broader range of retrievers under a unified implementation.

All models are integrated using Pyserini's `SimpleDenseSearcher` and `DenseVectorPrf` classes. At inference time, each retriever relies on its corresponding `QueryEncoder`, embedded within `SimpleDenseSearcher`, to encode queries on-the-fly. No PRF-specific parameters are tuned for individual models; the same PRF configuration is applied uniformly across all retrievers to ensure consistency in evaluation.

As additional baselines, we include results from BM25, BM25+RM3, and BM25 with BERT-based reranking [157], alongside the corresponding dense retrievers without PRF.

Evaluation is conducted using standard TREC DL metrics: Mean Average Precision (MAP), nDCG@{10,100}, and Recall@1000. All experiments are run on a 2019 MacBook Pro equipped with a 2.4 GHz 8-Core Intel Core i9 CPU and 64GB of RAM. No GPU is used during inference. The batch size is set to 64, with 12 threads enabled for parallel processing. Dense indexes for all models are pre-built offline using FAISS [41].

9.3.2 Results

To validate the extensibility and correctness of our PRF framework, we conduct comprehensive experiments on TREC DL 2019 and 2020, as well as the MS MARCO dev set. The results are shown in Table 9.1 and Table 9.2. Dense retriever-based PRF runs can be executed using Code Samples 9.3 and 9.4 by substituting the appropriate parameters for `--topics` and `--encoder`. All experiments were conducted using untuned PRF parameters across all models to ensure fair comparison and reproducibility.

We evaluated five dense retrievers (ANCE, TCT-ColBERT V1 and V2, DistillBERT KD, DistillBERT Balanced, SBERT, and ADORE) alongside traditional baselines (BM25, BM25+RM3, and BM25 + BERT-base). All models and datasets are directly supported by Pyserini, and all results reported are fully reproducible with minimal configuration changes[1].

We begin by validating our implementation against the ANCE + VPRF setting originally studied in Chapter 5 [104]. As shown in Table 9.1, we successfully reproduce the observed gains from PRF: both the Average and Rocchio variants improve over the original ANCE baseline on all metrics for TREC DL 2019 and 2020. We then extend the evaluation to the other dense retrievers. Despite using the same default PRF settings across all models, we observe consistent improvements in nDCG@100 and Recall@1000 across most models and datasets. For example, TCT-ColBERT V2 achieves its highest

[1] Full instructions and additional experiments are available at: `https://github.com/castorini/pyserini/blob/master/docs/experiments-vector-prf.md`

Table 9.1: Comparison between dense retrievers with and without PRF on two popular TREC DL datasets. All PRF parameters are not tuned: PRF depth $k = 3$ for VPRF-Average; for VPRF-Rocchio $\alpha = 0.4$, $\beta = 0.6$, PRF depth $k = 5$. **Bold** indicates the best results w.r.t. each base model. Significant improvements over corresponding baselines are marked with †.

		TREC DL 2019				TREC DL 2020			
Model	PRF	MAP	nDCG@10	nDCG@100	Recall@1000	MAP	nDCG@10	nDCG@100	Recall@1000
BM25	-	0.3773	0.5058	0.5018	0.7389	0.2856	0.4796	0.4902	0.7863
BM25	RM3	0.4270	0.5180	0.5286	**0.7882**	0.3019	0.4821	0.5077	**0.8217**
BM25 + BERT-base	-	**0.4827**	**0.7061**	**0.6426**	0.7389	**0.4926**	**0.7064**	**0.6473**	0.7863
ANCE [223]	Original	0.3710	0.6452	0.5540	0.7554	0.4076	0.6458	0.5679	0.7764
	Average PRF	**0.4247**†	0.6532	**0.5937**†	0.7739	**0.4325**†	**0.6573**	0.5793	0.7909
	Rocchio	0.4211	**0.6539**	0.5928†	**0.7825**†	0.4315	0.6471	**0.5800**	**0.7957**
TCT-ColBERT V1 [114]	Original	0.3906	0.6700	0.5730	0.7916	0.4290	0.6678	0.5826	0.8181
	Average PRF	0.4336	0.6639	0.6119†	0.8230	**0.4725**†	**0.6957**	**0.6101**	**0.8667**†
	Rocchio	**0.4463**†	**0.6875**	**0.6143**†	**0.8393**†	0.4625†	0.6945	0.6056	0.8576†
TCT-ColBERT V2 [115]	Original	0.4269	0.7204	0.6129	0.8342	0.4717	**0.6882**	0.6200	0.8407
	Average PRF	**0.4766**†	**0.7312**	**0.6487**†	**0.8574**	0.4701	0.6836	0.6209	0.8739†
	Rocchio	0.4709†	0.7111	0.6435	0.8496	**0.4819**	0.6804	**0.6324**	**0.8760**†
DistillBERT KD [72]	Original	0.3759	0.6994	0.5765	0.6853	0.3909	**0.6447**	0.5728	0.6893
	Average PRF	0.4362†	**0.7096**	**0.6217**†	0.7180	0.3955	0.6316	0.5755	**0.7279**
	Rocchio	**0.4378**†	0.7052	0.6189	**0.7291**†	**0.3990**	0.6289	**0.5760**	0.7222
DistillBERT Balanced [73]	Original	0.4761	0.7210	0.6360	0.7826	0.4755	0.6854	0.6346	0.8009
	Average PRF	0.5057	0.7190	0.6526	0.8054	**0.4873**	**0.7086**†	0.6449	**0.8392**†
	Rocchio	**0.5249**†	**0.7231**	**0.6684**	**0.8352**†	0.4846	0.7083†	**0.6470**	0.8262†
SBERT [173]	Original	0.4060	0.6930	0.5985	0.7872	0.4124	0.6344	0.5734	0.7937
	Average PRF	0.4354†	**0.7001**	**0.6149**	0.7937	0.4258	0.6412	0.5781	0.8169
	Rocchio	**0.4371**†	0.6952	**0.6149**	**0.7941**	**0.4342**†	**0.6559**	**0.5851**	**0.8226**†
ADORE [239]	Original	0.4188	0.6832	0.5946	0.7759	0.4418	0.6655	0.5949	0.8151
	Average PRF	0.4672†	0.6958	**0.6263**	0.7890	0.4706†	**0.7086**†	0.6176	**0.8323**
	Rocchio PRF	**0.4760**†	**0.7021**	0.6193	**0.8251**†	**0.4760**†	0.7019†	**0.6193**	0.8251

Recall@1000 on TREC DL 2020 with VPRF-Rocchio (0.8760), outperforming even its strong original baseline.

From the results, it is observable that VPRF substantially boosts Recall@1000, which is critical in multi-stage retrieval pipelines where dense retrievers serve as the first-stage ranker. This trend is consistent across all models, with TCT-ColBERT V2 achieving Recall@1000 of 0.8760 on TREC DL 2020 when using VPRF-Rocchio, outperforms all traditional and dense retriever baselines.

We also have two additional observations: first, these improvements are achieved without any parameter tuning, we use the parameters from the previous study on VPRF, this demonstrates the robustness and transferability of the PRF module across models and datasets. Second, PRF enhances even strong retrievers like DistillBERT Balanced, enabling it to outperform BM25 + BERT-base on all metrics on TREC DL 2019, and on TREC DL 2020 except MAP and nDCG@100.

Latency is also evaluated to assess the practical feasibility of the framework. For example, with ANCE, average latency increases from 209ms (without PRF) to 395ms (with PRF) per query, staying below twice the original inference time. This is because the second-stage PRF retrieval uses a pre-computed vector and skips query encoding. We note this fairly low latency (comparable, if not lower than BM25+RM3 for long feedback inputs) makes this PRF framework efficient and effective – and it can serve as a "free boost" to all dense retrievers.

To further validate the framework, we also tested on the MS MARCO dev set (Table 9.2). As discussed in Chapter 5, this dataset is suboptimal for PRF evaluation due to its limited and sparse

Model	Method	MAP	nDCG@10	nDCG@100	Recall@1000
ANCE	Original	0.3362	0.4457	0.9587	0.3302
	Average PRF	0.3133	0.4247	0.9490	0.3073
	Rocchio PRF	0.3115	0.4250	0.9545	0.3048
TCT-ColBERT V1	Original	0.3416	0.4514	0.9640	0.3350
	Average PRF	0.2882	0.4014	0.9452	0.2816
	Rocchio PRF	0.2809	0.3988	0.9543	0.2740
TCT-ColBERT V2 HN+	Original	0.3644	0.4750	0.9695	0.3590
	Average PRF	0.3183	0.4325	0.9585	0.2995
	Rocchio PRF	0.3190	0.4360	0.9659	0.2933
DistillBERT KD	Original	0.3309	0.4391	0.9553	0.3250
	Average PRF	0.2830	0.3940	0.9325	0.2470
	Rocchio PRF	0.2787	0.3937	0.9432	0.2716
DistillBERT Balanced	Original	0.3515	0.4651	0.9771	0.3443
	Average PRF	0.2979	0.4151	0.9613	0.2630
	Rocchio PRF	0.2969	0.4178	0.9702	0.2897
SBERT	Original	0.3373	0.4453	0.9558	0.3314
	Average PRF	0.3094	0.4183	0.9446	0.3035
	Rocchio PRF	0.3034	0.4157	0.9529	0.2974
ADORE	Original	0.3523	0.4637	0.9688	0.3466
	Average PRF	0.3188	0.4330	0.9583	0.3127
	Rocchio PRF	0.3209	0.4376	0.9669	0.3145

Table 9.2: Comparison between dense retrievers with and without PRF on the dev queries set of the MS MARCO dataset. All PRF parameters are not tuned: PRF depth $k = 3$ for average; for Rocchio $\alpha = 0.4$, $\beta = 0.6$, PRF depth $k = 5$. The **Original** method gives the best results over all metrics consistently.

relevance judgments: on average, only one relevant passage per query is available, which often overlaps with PRF candidates. As expected, both PRF methods tend to underperform on MS MARCO, often yielding effectiveness drops. Nevertheless, these experiments confirm that the framework executes correctly and consistently across datasets.

Overall, these results provide a strong empirical answer to RQ2.4: How to design an extensible Pseudo-Relevance Feedback framework? Our implementation in Pyserini is modular, retriever-agnostic, and evaluation-friendly. It requires no model re-training, supports a broad range of dense retrievers out of the box, and introduces only modest computational overhead. It enables researchers to evaluate and integrate PRF methods across retrieval models with minimal effort, simply by altering command-line arguments. Moreover, it offers a practical starting point for further research on PRF strategies, including parameter tuning, alternative aggregation methods, or integration with rerankers.

9.4 Summary

Reproducing results from prior research and integrating new techniques into existing infrastructures remain challenging in the IR community. In this chapter, we addressed **RQ2.4: How to design an extensible Pseudo-Relevance Feedback framework?** by presenting a modular PRF architecture built into the Pyserini toolkit. While we utilized the Vector-based PRF (VPRF) strategies from Chapter 5 as the foundational case study, this implementation serves as a generalized blueprint designed to support the broader spectrum of feedback methods explored throughout this thesis, including the learned adaptivity of TPRF (Chapter 8 and future generative approaches like PromptPRF (Chapter 11).

By implementing VPRF-Average and VPRF-Rocchio, we demonstrated that our framework correctly replicates established effectiveness improvements on TREC DL benchmarks. Beyond replication, we extended the evaluation to a diverse suite of dense retrievers supported by Pyserini, including TCT-ColBERT, DistillBERT, SBERT, and ADORE. Our findings reveal that the framework consistently enhances retrieval effectiveness across models, especially in terms of Recall@1000, achieving these gains with untuned parameters. This validates the robustness of the infrastructure and confirms that the vector manipulation logic can be decoupled from the specific retriever backbone.

From a systems perspective, the framework lowers the barrier for both reproducing existing work and exploring new PRF directions. Its modular design allows researchers to integrate new retrievers by simply referencing model checkpoints, or to implement novel feedback strategies—such as the transformer-based logic of TPRF, by inheriting the abstract base classes. It is also worth noting that this modular approach to dense feedback is not exclusive to Pyserini; the VPRF strategies established here have also been integrated into other powerful toolkits such as PyTerrier[2] [136], further underscoring the universality and practical utility of these vector-based feedback mechanisms.

This chapter concludes Part 2 of the thesis, which focused on developing adaptive and efficient feedback mechanisms for neural retrieval. We began in Chapter 7 by systematically quantifying the impact of feedback signal quality, identifying the sensitivity of PRF pipelines to noise and motivating the need for robustness. In Chapter 8, we introduced TPRF, a lightweight transformer-based model that addresses these sensitivity issues by learning to weight feedback signals dynamically, balancing efficiency with effectiveness. Finally, Chapter 9 provided the necessary software infrastructure to deploy these innovations, ensuring they are accessible for reproducibility and rapid prototyping.

While the methods explored in Part 2 are lightweight and effective, they rely primarily on manipulating existing vector representations. The recent emergence of Large Language Models (LLMs) introduces new opportunities to generate entirely new feedback signals through semantic understanding and reasoning.

In Part 3, we move beyond discriminative neural PRF models to explore the integration of generative AI into the feedback loop. The next chapter introduces LLM-VPRF, which generalizes the vector-based concepts established in this part to the realm of LLMs, leveraging their generative capabilities to construct powerful feedback-aware representations without requiring retriever-specific training.

[2] `https://pyterrier.readthedocs.io/en/latest/ext/pyterrier-dr/prf.html`

Part 3

Large Language Model-Based Feedback and Emerging Directions

Introduction to Part 3

In the previous parts of this thesis, we have developed and evaluated a range of PRF methods for BERT-based rerankers [155] and encoder-based dense retrievers [72, 73, 114, 115, 223, 239], with the focus mainly on their practicality, adaptability, and computational efficiency. However, the emergence of decoder-style Large Language Models (LLMs) introduces a new retrieval paradigm. In this part of the thesis, we explore whether PRF remains applicable and beneficial to these generative LLMs. We aim to address the final research question:

RQ3: Are Pseudo-Relevance Feedback models still applicable to decoder-style Large Language Models?

We begin in Chapter 10 by generalizing VPRF mechanisms (Chapter 5) to LLM-based dense retrievers [13, 128, 250]. We investigate whether the VPRF approach, which uses query embeddings and top-ranked passage embeddings, can be effectively applied to retrieval systems that rely on LLM-based dense representations. Our findings show that, even in zero-shot scenarios, VPRF substantially improves the performance across dense retrievers with different LLM backbones, all with negligible inference cost and no additional model complexity.

Building upon these insights and the generative nature of decoder-style LLMs, Chapter 11 introduces PromptPRF [106], a novel generative PRF method designed specifically for decoder-style LLMs based on the PromptReps dense retrieval framework [250]. PromptPRF leverages the generative capabilities of LLMs to generate query-independent features from passages, then reformulate the query representations with feedback passage features through carefully designed prompts. Rather than using the original passage content or modifying the retriever or performing additional re-ranking, PromptPRF operates entirely at the prompt level, making it highly suitable for zero-shot retrieval. We systematically study PromptPRF across multiple LLM-based dense retrievers and datasets, demonstrating consistent gains in retrieval effectiveness, even under tight latency and hardware constraints. PromptPRF proves particularly effective for smaller-scale models, where PromptPRF enables them to approach or exceed the performance of larger dense retrieval models.

Together, these chapters demonstrate that PRF remains an effective strategy even in the context of generative LLM-based dense retrieval models. By shifting from embedding manipulation feedback mechanisms to prompt-based reformulation, we preserve the core benefits of PRF while embracing the flexibility of LLMs. These contributions offer a pathway for integrating traditional retrieval concepts into emerging generative retrieval frameworks and highlight the evolving role of PRF in the era of large language models.

Chapter 10

LLM-VPRF: Generalizing Vector-Based Pseudo-Relevance Feedback to Large Language Models

Building on previous work with encoder-based dense retrievers, this chapter shifts focus to dense retrieval models built on decoder-style Large Language Models (LLMs). Unlike BERT-style encoders, LLM-based retrievers generate semantically rich embeddings without contrastive training and have shown strong performance across tasks [57, 108, 109]. However, the application of PRF to such models remains largely unexplored.

Therefore, this chapter investigates the following research question:

RQ3.1: Can Vector-based Pseudo-Relevance Feedback generalize to decoder-style Large Language Models?

To address this question, we introduce LLM-VPRF, a method that extends the established VPRF framework from Chapter 5 to operate with representative LLM-based dense retrievers [13, 128, 250]. This work evaluates the compatibility and effectiveness of VPRF when applied to decoder-based architectures, including PromptReps [250], RepLLaMA [128], and LLM2Vec [13].

Through comprehensive evaluations on the BEIR and TREC DL datasets, we observe that integrating VPRF with LLM-based dense retrievers provides consistent improvements in retrieval performance, particularly for nDCG@10 and Recall@100. While supervised models such as RepLLaMa demonstrate stable and reliable gains, zero-shot models like PromptReps exhibit higher variance, with effectiveness heavily influenced by the choice of feedback passages. Despite these variations, the overall efficiencies of the VPRF method remains competitive, introducing minimal computational overhead and preserving its suitability for real-world deployment. Furthermore, oracle analyses reveal substantial space for further improvement, re-highlighting the necessity for more refined or adaptive feedback selection strategies.

These results confirm that VPRF is not limited to encoder-based architectures and can be effectively generalized to LLM-based dense retrievers. This chapter marks a key step in extending classical neural

feedback techniques to modern LLM frameworks and provides the foundation for the next chapter (Chapter 11), which explores prompt-driven and generative approaches to LLM-based feedback.

10.1 Vector-Based Pseudo Relevance Feedback with Large Language Models

This section presents the methodology of LLM-VPRF, which extends the VPRF framework from Chapter 5 to LLM based dense retrievers. The goal is to investigate whether feedback-driven query refinement in vector space remains effective when applied to semantically rich embeddings generated by modern decoder-style LLMs.

LLM-VPRF modifies the original query embeddings using vectors from top-ranked feedback passages, operating entirely in the embedding space without altering the query text. This design enables the method to remain model-agnostic and efficient while leveraging the enhanced representation capacity of LLMs. The following subsections describe the implementation of LLM-VPRF, including the generation of embeddings from different LLM-based retrievers, the application of vector-based feedback methods such as Average and Rocchio, and the retrieval configurations used for evaluation.

10.1.1 LLM-VPRF Implementation

To evaluate the effectiveness of VPRF with LLM-based dense retrievers, we build upon the framework introduced in Chapter 5 and adapt it to operate with decoder-style LLM embeddings. This implementation consists three key steps: (1) generating query and passage embeddings using representative LLM retrievers, (2) applying vector-based feedback methods to refine query representations, and (3) configuring retrieval and evaluation pipelines for consistent comparison. Each component is described in detail below.

10.1.2 Embedding Generation

Our experiments utilize three representative LLM-based retrievers to generate dense query and passage embeddings: PromptReps [1] [250], RepLLaMA[2] [128], and LLM2Vec[3] [13]. These models were selected to cover a range of retrieval scenarios: PromptReps represents an unsupervised zero-shot approach using Meta-LLaMA-3-8B-Instruct; RepLLaMA is a supervised model fine-tuned from LLaMA-2-7B with retrieval-specific objectives; and LLM2Vec employs a multi-objective training setup (bidirectional attention, masked next token prediction, and SimCSE) to enhance embedding quality from a decoder-only architecture.

For PromptReps and RepLLaMA, we use the Tevatron framework [54] for passage and query encoding. Tevatron enables flexible encoder-side inference with Huggingface Transformers and is

[1] `https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct`
[2] `https://huggingface.co/castorini/repllama-v1-7b-lora-passage`
[3] `https://huggingface.co/McGill-NLP/LLM2Vec-Mistral-7B-Instruct-v2-mntp-unsup-simcse`

compatible with custom LLM adapters. For LLM2Vec, we adapt the original authors' public encoding scripts to align with our retrieval pipeline while preserving all core pre-processing steps and tokenizer configurations.

All embeddings are generated with two NVIDIA H100 GPUs. We use a batch size of 128 across all models to maintain consistent GPU memory utilization and throughput. Input texts are truncated to 512 tokens, and embeddings are extracted from the final hidden layer of the decoder output corresponding to the last token in the input. The resulting representations are L2-normalized to facilitate similarity-based retrieval.

To ensure reproducibility, we release our full experimental infrastructure, including embedding scripts, index construction routines, and evaluation scripts at `https://github.com/ielab/LLM-VPRF`.

10.1.3 Recap of VPRF Methods

To evaluate the compatibility of VPRF with LLM-based dense retrievers, we adopt the methodology established in Chapter 5. We implement two fundamental strategies to refine query representations using the richer embeddings generated by LLMs (refer to Section 5.1 for theoretical details):

- **VPRF-Average:** This parameter-free method computes the mean of the original query vector and the top-k feedback passage vectors, treating both sources of information as equally important.
- **VPRF-Rocchio:** This method introduces tunable parameters α and β to control the weighting between the original query and the feedback signal, allowing for a calibrated balance between query retention and expansion.

For the formal mathematical definitions (Eq. 5.1 and Eq. 5.2) and detailed theoretical justifications of these vector operations, we refer readers to Section 5.1 in Chapter 5.

10.1.4 Retrieval and VPRF Parameters

To assess the impact of VPRF on LLM-based dense retrievers, we apply it to three representative decoder-only dense retrievers: PromptReps, RepLLaMa, and LLM2Vec. The initial retrieval outputs from these models serve as the baselines for comparison.

In the Rocchio variant of VPRF, we investigate two parameterization strategies to control the influence of original query embeddings and feedback signals. In the first strategy, we vary both α and β independently across the range $[0.1, 0.9]$ with a step size of 0.1 to explore the interaction space. In the second, we fix $\alpha = 1.0$ and vary β over the same range to isolate the effect of feedback magnitude while preserving the full weight of the original query. These settings allow us to understand how the balance between original and feedback information affects retrieval outcomes. For both strategies, we evaluate feedback depths $k \in \{1, 2, 3, 5, 10\}$, which represent common values used in pseudo-relevance feedback literature and in earlier chapters of this thesis.

All VPRF updates are performed directly in the embedding space, modifying query vectors without altering input text or model architecture. This approach is entirely model-agnostic and applies uniformly to decoder-only retrievers, making it suitable for integration into a wide range of dense retrieval systems. The feedback-enhanced queries are used in a second retrieval stage with the full passage corpus using the updated representations.

For retrieval, we employ the FAISS library [41] for exact nearest neighbor search. All retrieval experiments are conducted on 2 NVIDIA H100 GPUs. We use a batch size of 128 for all datasets and models to maintain consistent throughput and ensure reproducibility. To minimize runtime overhead, all passage embeddings are cached in GPU memory, and query vectors are processed in batches. This setup enables fair and efficient comparison across models and feedback configurations.

10.2 Empirical Evaluation

This section presents a comprehensive empirical evaluation of the LLM-VPRF framework. In particular, we aim to assess how the semantic richness of LLM-generated embeddings interacts with VPRF techniques and whether this interaction leads to measurable gains in retrieval effectiveness.

To guide this evaluation, we formulate three sub research questions under the high-level research question **RQ3.1: Can Vector-based Pseudo-Relevance Feedback generalize to decoder-style Large Language Models?**:

RQ3.1.1 : Can VPRF consistently improve retrieval performance when applied to LLM-based dense retrievers?

RQ3.1.2 : How do the semantic characteristics of LLM embeddings influence the effectiveness of the feedback mechanism?

RQ3.1.3 : What are the practical implications of applying VPRF to LLM-based retrievers in terms of efficiency and scalability?

By addressing these questions, we aim to extend the understanding of how VPRF interacts with LLM architectures and to evaluate the feasibility of incorporating feedback-driven refinement into emerging LLM retrieval pipelines.

10.2.1 Dataset and Evaluation

To evaluate the retrieval effectiveness of LLM-VPRF, we conduct experiments using three LLM-based dense retrievers as mentioned in Section 10.1.2: PromptReps [250], RepLLaMA [128], and LLM2Vec [13]. All models are evaluated across two widely adopted benchmarks: the BEIR benchmark [200] and the TREC Deep Learning Tracks from 2019 and 2020 [26, 27].

For the BEIR benchmark, we select 13 datasets out of the 19 available. This subset is consistent with prior retrieval studies and offers a diverse range of information retrieval tasks, including fact

verification (FEVER, SciFact), question answering (NQ, HotpotQA), argument retrieval (ArguAna), and others. These datasets vary in domain, document structure, and query type, enabling a broad evaluation of model generalizability and robustness. Each dataset provides a corpus of passages, a set of queries, and relevance judgments based on human annotations or weak supervision.

For the TREC Deep Learning (TREC DL) benchmarks, we use the passage ranking tracks from TREC DL 2019 and TREC DL 2020. They serve as high-quality test collections for evaluating fine-grained ranking performance in information retrieval tasks, and are widely used to evaluate dense retrieval models.

For all datasets, we report two standard retrieval metrics: nDCG@10 and Recall@100. nDCG@10 measures ranking quality at a depth of 10 with graded relevance, while Recall@100 quantifies the model's ability to retrieve relevant documents within the top 100 results. These metrics allow us to make comparative analysis across models and configurations.

10.2.2 Comparison Methods

PromptReps [250] is an unsupervised zero-shot dense retriever built on Meta-Llama-3-8B-Instruct[1]. It employs a prompting strategy to generate both dense and sparse representations without requiring any task-specific fine-tuning. In this study, we use only the dense representations produced by PromptReps to ensure a controlled comparison across models that rely exclusively on dense embeddings for retrieval. Despite its zero-shot nature, PromptReps has demonstrated competitive or superior performance to traditional baselines across a range of datasets.

RepLLaMA [128] is a supervised LLM-based retriever built upon LLaMA-2-7B[4] [202]. It employs a multi-stage training pipeline including synthetic data generation, supervised contrastive learning, and cross-encoder distillation. This process fine-tunes the model to produce passage embeddings optimized for semantic similarity in retrieval tasks. We follow the methodology outlined in the original work and independently implement the supervised fine-tuning using the same LLaMA-2-7B foundation model. RepLLaMA serves as a strong supervised baseline representing the current state of LLM-based dense retrieval with task-specific training.

LLM2Vec [13] is a supervised text encoding framework based on Mistral-7B-Instruct[3], but trained using a multi-objective learning setup. It incorporates bidirectional attention, masked next token prediction, and contrastive learning via SimCSE[55] to produce semantically rich and context-aware embeddings from decoder-only models. This training strategy allows LLM2Vec to better capture sentence-level semantics, enables the models to capture contextual information more effectively, achieving competitive performance across various retrieval tasks. In our experiments, we use the official model checkpoint and follow the original authors' encoding protocol to ensure consistent evaluation.

[4] `https://huggingface.co/meta-llama/Llama-2-7b-hf`

10.2.3 Evaluation Protocol

To report aggregated performance, we adopt two evaluation strategies. **Best Average** refers to the best-performing configuration (i.e., combination of VPRF parameters) after effectiveness scores are averaged across all datasets in a dataset collection (BEIR or TREC DL). **Optimal Best**, on the other hand, selects the best-performing configuration individually for each dataset and then computes the average of these per-dataset maximum scores. This distinction allows us to assess both the general robustness of a single parameter setting (Best Average) and the upper bound of VPRF effectiveness under ideal tuning conditions (Optimal Best). Statistical significance is assessed using a two-tailed paired t-test with Bonferroni correction.

10.3 Results

This section presents the empirical results of applying VPRF to LLM-based dense retrievers, followed by analysis across different evaluation aspects. We begin by assessing overall performance improvements in ranking effectiveness across the BEIR and TREC DL datasets, then examine the influence of LLM semantic representations on feedback behavior, and conclude with an evaluation of the efficiency and practical implications of the proposed method.

10.3.1 Performance Improvements with LLM-based Dense Retrievers

To investigate whether VPRF remains effective when applied to LLM-based dense retrievers and address RQ3.1.1, we evaluate performance across a diverse set of retrieval models and benchmark datasets. Table 10.1 reports retrieval results for PromptReps, RepLLaMA, and LLM2Vec after applying VPRF with the best-performing configuration for each dataset.

Across both TREC DL and BEIR datasets, we observe consistent improvements in retrieval effectiveness, particularly for the supervised model RepLLaMA. On average, RepLLaMA achieves the highest overall scores: 0.7312 in Recall@100 and 0.7548 in nDCG@10 on TREC DL, and 0.7022 and 0.5458 respectively on BEIR. These results confirm that VPRF can reliably enhance performance when applied to a fine-tuned LLM retriever. Moreover, RepLLaMA outperforms the other two models across nearly all datasets, suggesting that supervised training facilitates better integration of feedback signals during query reformulation.

For LLM2Vec, although generally less competitive on BEIR, shows substantial gains on TREC DL, reaching 0.5793 Recall@100 and 0.5131 nDCG@10. These results suggest that VPRF is particularly beneficial when applied to LLMs trained for fine-grained representation learning tasks. The stronger performance on TREC DL indicates that VPRF may be especially effective in domains with more structured or semantically rich queries, such as web search.

For PromptReps, which is a zero-shot unsupervised model, demonstrates more variable performance. While it achieves reasonable scores on average, obtaining 0.5041 for nDCG@10 on TREC and 0.3613 on BEIR, its Recall@100 drops slightly on BEIR (0.5649) compared to its baseline (0.5651).

Datasets	PromptReps				RepLLaMA				LLM2Vec			
Metric	R@100		nDCG@10		R@100		nDCG@10		R@100		nDCG@10	
TREC DL 2019	0.4778	**0.5017**†	0.5062	**0.5431**†	0.6753	**0.6938**	0.7319	**0.7596**†	0.4931	**0.5258**	0.4011	**0.4947**†
TREC DL 2020	**0.5101**	0.5061	0.4381	**0.4651**	0.7537	**0.7685**	0.7335	**0.7499**	0.5906	**0.6327**†	0.4690	**0.5314**†
TREC Average	0.4940	**0.5039**	0.4722	**0.5041**	0.7145	**0.7312**	0.7327	**0.7548**	0.5419	**0.5793**	0.4351	**0.5131**
ArguAna	0.9047	**0.9061**	0.2970	**0.2981**	0.9367	**0.9523**†	0.4612	**0.4664**	0.9836	**0.9879**	**0.5119**	0.5103
Climate-FEVER	**0.5195**	0.5187	0.1992	**0.2016**	0.5805	**0.6312**†	0.2321	**0.3041**†	0.5636	**0.5826**	0.2122	**0.2464**†
DBPedia	0.3553	**0.3557**	**0.3153**	0.3142	0.5745	**0.6100**†	0.4653	**0.4763**	0.3744	**0.3836**	0.2420	**0.2439**
FEVER	**0.7913**	0.7741	**0.5628**	0.5560	0.9501	**0.9550**	0.7623	**0.7690**	**0.8386**	0.8359	**0.4415**	0.4409
FiQA-2018	0.6164	**0.6214**	**0.2707**	0.2704	0.7200	**0.7295**	0.4145	**0.4160**	0.6478	**0.6538**	0.2659	**0.2713**†
HotpotQA	**0.3731**	0.3535	**0.1964**	0.1884	0.8442	**0.8538**†	**0.6978**	0.6961	**0.7561**	0.7503	**0.5455**	0.5345
NFCorpus	0.2879	**0.2930**	0.2956	**0.3010**	0.3309	**0.3537**†	0.3647	**0.3848**	0.2869	**0.3077**†	0.2769	**0.3027**†
NQ	**0.7618**	0.7581	**0.3443**	0.3438	0.9654	**0.9713**	0.6185	**0.6225**	0.8527	**0.8626**	0.3312	**0.3446**
Quora	**0.9562**	0.9546	**0.7255**	0.7224	0.9953	**0.9957**	0.8716	**0.8741**	**0.9914**	0.9908	**0.8608**	0.8589
SCIDOCS	0.4312	**0.4368**	0.1850	**0.1868**	0.4182	**0.4579**†	0.1847	**0.1992**	0.3999	**0.4075**	0.1518	**0.1569**
SciFact	0.8767	**0.8800**	**0.5262**	0.5245	0.9567	**0.9633**	0.7138	**0.7212**	0.9593	**0.9609**	**0.6886**	0.6849
Touche-2020	0.3539	**0.3634**	0.1485	**0.1614**	0.4908	**0.4946**	0.3478	**0.3499**	0.2665	**0.3199**†	0.0759	**0.0983**
TREC-COVID	0.1180	**0.1277**	0.5951	**0.6281**†	0.1540	**0.1605**	0.8013	**0.8164**	0.0783	**0.0812**	0.5161	**0.5252**
BEIR Average	**0.5651**	0.5649	0.3586	**0.3613**	0.6860	**0.7022**	0.5335	**0.5458**	0.6153	**0.6250**	0.3939	**0.4014**

Table 10.1: The evaluation results for each dataset and each model with best results among all VPRF parameters, left and right sub-columns under each evaluation metric represent Baseline and with VPRF, respectively. Significant improvements over corresponding baselines are marked with †.

This suggests that without task-specific supervision, the effectiveness of VPRF is highly sensitive to the quality of initial retrieval and the feedback passages it selects. However, PromptReps still delivers strong results on semantically focused datasets such as ArguAna, Quora, and SciFact, where query and passage alignment is less ambiguous.

Overall, the findings support that VPRF generalizes well across different LLM retrievers. The degree of improvement varies depending on model architecture and training approaches: supervised models benefit most consistently, while zero-shot models exhibit higher variance. These results provide strong evidence that VPRF is not only compatible with decoder-style LLM retrievers but can also offer meaningful effectiveness gains when integrated into LLM-driven retrieval pipelines.

10.3.2 Impact of LLM Semantic Capabilities on Feedback Mechanism

To address RQ3.1.2, we analyze how the semantic properties of different LLM embeddings affect the ability of VPRF to improve retrieval performance. Specifically, we examine the gap between the Best Average and Optimal Best scores, which reflects how effectively a model can benefit from feedback given ideal parameter settings. A gap from previous chapters suggests that the model has latent potential to exploit semantic signals more effectively, but may be limited by variability in feedback quality or sensitivity to hyperparameter selection.

PromptReps exhibits the largest discrepancy between Best Average [5] and Optimal Best across both benchmarks. On BEIR (Table 10.2, its nDCG@10 improves from 0.3318 to 0.3613, and R@100 from 0.5226 to 0.5649. This significant difference implies that PromptReps, while effective under ideal conditions, is more sensitive to feedback passage selection and parameter tuning. As a zero-shot

[5]Averaging effectiveness scores across datasets is generally discouraged [181], it remains a common practice in BEIR evaluations.

unsupervised model, its embeddings may be more contextually diverse but less grounded in retrieval-optimized structure, making consistent signal extraction from feedback passages more difficult without fine-grained control.

RepLLaMA, in contrast, shows smaller but more stable gains between Best Average and Optimal Best. On BEIR, the difference is only +0.8% for R@100 and +1.3% for nDCG@10 compared to baseline, indicating that the model is already near its potential with minimal tuning. This suggests that supervised fine-tuning produces more retrieval-aligned embeddings that are inherently easier to update with feedback, making them more robust across settings. Similar trends hold on TREC DL, with stable improvements (+1.3% for R@100 and +0.9% for nDCG@10 compare to baseline), re-highlighting that RepLLaMA embeddings are semantically consistent and more effectively leverage PRF without overfitting to parameter sensitivity.

LLM2Vec sits between the two extremes. On TREC DL, it benefits strongly from optimal settings, improving R@100 from 0.5419 to 0.5793 (+6.9%) and nDCG@10 from 0.4351 to 0.5131 (+17.9%) compared to baseline. This large increase in nDCG@10 highlights LLM2Vec's capacity for semantic understanding and fine-grained relevance, potentially due to its multi-objective training involving SimCSE and bidirectional attention. However, on BEIR, the gains are more modest, suggesting its effectiveness may vary depending on domain complexity and query formulation style.

These results demonstrate that the effectiveness of VPRF is closely tied to the semantic structure of the underlying LLM embeddings. Models with more stable or fine-tuned semantic representations (e.g., RepLLaMa) are better positioned to integrate feedback consistently, while models with higher flexibility or unsupervised objectives (e.g., PromptReps and LLM2Vec) offer greater potential under optimal configurations but require more careful control of feedback dynamics. This insight highlights the importance of embedding stability and alignment with retrieval objectives when designing or selecting LLMs for PRF-based pipelines.

10.3.3 Practical Implications for Retrieval Systems

To address RQ3.1.3, we evaluate the computational efficiency of applying VPRF to LLM-based dense retrievers and consider its scalability in practical settings. Table 10.3 reports the average query latency (in ms) for each retriever on the TREC DL 2019 dataset, using 43 queries and a feedback depth of $k = 3$. Latency is measured per query, and VPRF times exclude the first-stage retrieval since it is common to both baseline and feedback-enhanced pipelines.

Across all three retrievers, the baseline query latency is already low, ranging from 5.4ms (LLM2Vec) to 6.1ms (PromptReps). These values reflect the lightweight nature of embedding-based retrieval using precomputed passage vectors and indicate that all three models are suitable for responsive applications.

Applying VPRF introduces a modest increase in latency. The VPRF-Average method adds approximately 4.6–5.4ms per query, while VPRF-Rocchio incurs a slightly higher cost of 5.2–5.6ms. The consistency of these values across models suggests that the overhead introduced by vector aggregation is both small and predictable, with no heavy dependency on the underlying retriever

Models	Metric	Method	PromptReps	RepLLaMA	LLM2Vec
BEIR	R@100	Baseline	0.5247	0.6859	0.6153
		Best Average	0.5226(-0.4%)	0.6972(1.6%)	0.6193(0.7%)
		Optimal Best	**0.5649†‡(7.7%)**	**0.7022†(2.4%)**	**0.6250‡(1.6%)**
	nDCG@10	Baseline	0.3330	0.5335	0.3939
		Best Average	0.3318(-0.4%)	0.5390(1.0%)	0.3947(0.2%)
		Optimal Best	**0.3613†‡(8.5%)**	**0.5458‡(2.3%)**	**0.4014‡(1.9%)**
TREC DL	R@100	Baseline	0.4940	0.7145	0.5419
		Best Average	0.4926(-0.3%)	0.7290(2.0%)	0.5756†(6.2%)
		Optimal Best	**0.5039‡(2.0%)**	**0.7312†(2.3%)**	**0.5793†(6.9%)**
	nDCG@10	Baseline	0.4722	0.7327	0.4351
		Best Average	0.5017†(6.3%)	0.7484†(2.1%)	0.4935†(13.4%)
		Optimal Best	**0.5041†(6.8%)**	**0.7548(3.0%)**	**0.5131†‡(17.9%)**

Table 10.2: The results for the averaged results on BEIR 13 datasets and TREC DL 2019/2020. Where means just average of the baseline results with no PRF involved; Best Average represents the best results after averaged all 13 BEIR datasets or TREC DL 2019/2020 together; Optimal Best represents select the best results from each dataset before calculate the average. **Bold** texts means the best results for each metric, the percentage after results shows the increase/decrease regarding Baseline. Significant improvements over corresponding baselines are marked with †, the significant improvements between Optimal Best and Best Average are marked with ‡.

Models	PromptReps	RepLLaMa	LLM2Vec
Baseline	6.1ms (6.0ms)	6.0ms (6.2ms)	5.4ms (5.4ms)
VPRF-Average	4.6ms (4.6ms)	4.6ms (4.5ms)	5.4ms (5.3ms)
VPRF-Rocchio	5.4ms (5.5ms)	5.2ms (5.2ms)	5.6ms (5.6ms)

Table 10.3: Efficiency evaluation for each model with VPRF on TREC DL 2019 (2020), we use a moderate depth $k = 3$ for VPRF evaluation. The time measured is per query time consumption in milliseconds. VPRF time consumption is measured without first stage.

architecture. This is particularly encouraging for scalability, as it implies that VPRF can be deployed in parallelized or batched environments without introducing performance bottlenecks.

In practical scenarios with the full two-stage pipeline, both VPRF variants approximately double the query processing time compared to the baseline. However, since the baseline latencies are already below 10 milliseconds per query, the total inference time remains well within the constraints of real-time systems. Importantly, these measurements were obtained using a fixed feedback depth ($k = 3$), which reflects a moderate and commonly adopted setting in PRF literature. Further increases in k would marginally scale the feedback cost but remain computationally manageable for most deployments according to the latency analysis of VPRF in Chapter 5.

Overall, the efficiency analysis confirms that VPRF introduces minimal computational overhead relative to baseline retrieval, while offering substantial improvements in effectiveness. Therefore, the method is practical for deployment in high-throughput or latency-sensitive environments. These

findings reaffirm the feasibility of incorporating VPRF into modern LLM-based retrieval pipelines without sacrificing system responsiveness or scalability.

10.4 Summary

This chapter investigated whether VPRF can be effectively generalized to LLM-based dense retrievers, addressing the high-level research question **RQ3.1: Can Vector-based Pseudo-Relevance Feedback generalize to decoder-style Large Language Models?**Through extensive empirical analysis across BEIR and TREC DL datasets, we examined the effectiveness, impact of semantic capabilities, and practical deployment considerations of LLM-VPRF.

In response to **RQ3.1.1**, our results demonstrate that VPRF can consistently enhance retrieval performance across different LLM-based retrievers. Improvements were particularly stable for supervised models such as RepLLaMAA, while unsupervised models like PromptReps and LLM2Vec exhibited larger variance but also showed strong gains under optimal settings. This confirms that the VPRF mechanism remains effective even when extended to decoder-style LLM embeddings.

For **RQ3.1.2**, we found that the semantic characteristics of the underlying embeddings is critical. Models with more stable and fine-grained embeddings (e.g., RepLLaMA) show improvements with PRF consistently with less sensitivity to hyperparameters, whereas models trained with unsupervised or multi-objective strategies showed greater dependence on careful feedback selection. The observed gap between Best Average and Optimal Best results highlights the importance of semantic robustness in determining PRF effectiveness.

Regarding **RQ3.1.3**, our efficiency analysis showed that both Average and Rocchio variants of VPRF introduce only minimal latency overhead, only adds a few milliseconds per query, across all evaluated LLMs. These findings reaffirm the practicality of LLM-VPRF in latency-sensitive applications, and demonstrate its scalability for real-world retrieval scenarios.

Overall, our study establishes that VPRF generalizes beyond encoder-only architectures and can be effectively integrated into stronger LLM-based retrievers with minimal computational cost. Building on these insights, the next chapter introduces an alternative approach for combining PRF with LLMs that moves beyond embedding-level modification. We shift focus to PromptPRF, a method that leverages LLMs as generative agents to synthesize feedback signals directly through prompt-based generation. This approach offers a new perspective on integrating LLMs into the feedback loop, enabling zero-shot and rank-aware query reformulation without explicit vector operations like VPRF.

Chapter 11

PromptPRF – Pseudo-Relevance Feedback in Zero-shot Large Language Model Retrieval

In the previous chapter of this part, we explored how Large Language Models (LLMs) can be leveraged for retrieval and feedback. In Chapter 10, we established that the VPRF technique remains effective when applied to the high-dimensional embeddings produced by LLM-based retrievers. However, this approach utilizes the LLM primarily as a static encoder, leaving its generative reasoning capabilities – such as summarization, expansion, or explanation – untapped. Furthermore, the inherent computational cost of deploying large-scale LLM backbones for online query encoding remains a significant barrier to real-time deployment. In this chapter, we advance this line of investigation by exploring how to explicitly harness the generative nature of LLMs to construct richer feedback signals, and whether this can be achieved efficiently without incurring the prohibitive latency typically associated with large-scale inference.

LLM-based dense retrievers have demonstrated strong performance in zero-shot and prompt-based retrieval settings [128, 250], particularly with LLaMA series of models [42, 202]. However, such gains often come with substantial inference costs in terms of latency, memory, and hardware requirements [38]. These constraints limit the applicability of large retrievers in scenarios where responsiveness or resource efficiency is essential, such as mobile applications and resource constrained systems. This leads us to a central question: *Can smaller LLM-based dense retrievers match the effectiveness of their larger counterparts without increasing retrieval cost?*

Building on insights from earlier chapters on feedback effectiveness, this chapter investigates this question through a novel feature-based and prompt-based PRF framework: **PromptPRF**. PromptPRF is a lightweight, zero-shot feedback mechanism that enhances query representations using pre-computed features extracted from top-ranked passages. In contrast to embedding-based or online expansion methods, PromptPRF uses decoder-style LLMs to generate structured and unstructured feedback signals (e.g. `Summary`, `Keywords`, `Entities`, `Facts`, `News Articles`, etc.) offline. These features are integrated into the query through prompting at inference time, avoiding retriever fine-tuning and minimizing online computational cost.

To implement and validate PromptPRF, we extended the zero-shot PromptReps [250] dense retriever inherenting its zero-shot capabilities and retriever-agnostic characteristics. PromptPRF supports a wide range of feature types and LLM extractors, allowing for modular integration across retrieval pipelines. Throughout this chapter, we evaluate PromptPRF with all models from QWEN2.5 [199], QWEN3 [228], and GEMMA3 [198] model families with various model sizes on the TREC DL 2019/2020 [26, 27] and BEIR [200] datasets, systematically analyzing the impact of feature type, feedback depth, extractor scale, and rank-aware prompting.

This chapter seeks to answer the research question:

RQ3.2: Can we make LLM-based Pseudo-Relevance Feedback more effective and efficient?

Our empirical results demonstrate that PromptPRF significantly improves the performance of smaller dense retrievers, enabling them to approach or exceed the retrieval effectiveness of much larger models without PRF. This reframes the design space for LLM-based retrieval: rather than scaling the retriever itself, effectiveness gains can be achieved through intelligent and cost-efficient feedback mechanisms.

The remainder of this chapter details the PromptPRF methodology, experimental validation, results, and a comprehensive analysis of its behavior across different conditions.

11.1 Preliminary Efficiency Analysis

Deploying LLM-based dense retrievers typically entails substantial computational and infrastructural costs, regardless of the specific retriever architecture. To assess these practical constraints, we begin by analyzing the minimum computational demands for deploying an existing dense retrieval system, exemplified by PromptReps [250], which will be the backbone considered throughout this chapter. For this, we examine the hardware requirements for deployment, along with query encoding latency and computational efficiency during inference.

11.1.1 Hardware Requirements

The data listed in Table 11.1 illustrate the substantial growth in hardware requirements as the size of LLM backbones increases. Smaller-scale models such as QWEN2.5 0.5-1.5B Instruct, QWEN3 0.6-1.7B Instruct, and GEMMA3 1-4B Instruct are relatively lightweight and can be deployed using consumer-grade GPUs with moderate memory (e.g., NVIDIA RTX 3060 or NVIDIA RTX 3090) and minimal system RAM. In contrast, larger models like QWEN2.5 32-72B Instruct, QWEN3 32B, or GEMMA3 27B require specialized, high-memory GPU configurations such as dual NVIDIA H100 or NVIDIA A100 setups, along with significantly higher system RAM (32–140 GB) and disk space ($\geq$ 60 GB).

This escalation in deployment cost and complexity has direct implications for real-world applications. For instance, while the QWEN2.5 7B, QWEN2.5 14B, even QWEN2.5 32B models can be reasonably deployed on single-GPU servers, the QWEN2.5 72B-scale model require access to

Table 11.1: Minimum hardware requirements for different LLM backbones for dense retrieval. (All models used are instruct models.)

Model	Disk	RAM	Minimum GPU VRAM
QWEN2.5 0.5B	∼1.1 GB	2 GB	2 GB (e.g., RTX 3050, RTX 1650)
QWEN2.5 1.5B	∼3.3 GB	6 GB	4 GB (e.g., RTX 3060)
QWEN2.5 3B	∼6.5 GB	12 GB	8 GB (e.g., RTX 3060)
QWEN2.5 7B	∼13.4 GB	16 GB	14 GB (e.g., RTX 3090)
QWEN2.5 14B	∼26.8 GB	32 GB	28 GB (e.g., A100, RTX 4060 Ada)
QWEN2.5 32B	∼64 GB	96 GB	64 GB (e.g., A100/H100 80GB)
QWEN2.5 72B	∼145 GB	140 GB	165 GB ((e.g., 3×H100 or 3×A100)
QWEN3 0.6B	∼1.3 GB	4 GB	4 GB (e.g., RTX 3050, RTX 3060)
QWEN3 1.7B	∼3.9 GB	8 GB	8 GB (e.g., RTX 3060, RTX 4060)
QWEN3 4B	∼9.2 GB	16 GB	16 GB (e.g., RTX 4090)
QWEN3 8B	∼18 GB	32 GB	24 GB (e.g., RTX 4090, A100 40GB)
QWEN3 14B	∼30 GB	48 GB	48 GB (e.g., A100 40GB, H100)
QWEN3 32B	∼68 GB	96 GB	80 GB (e.g., A100/H100 80GB)
GEMMA3 1B	∼0.6 GB	2 GB	2 GB (e.g., GTX 1650)
GEMMA3 4B	∼16 GB	16 GB	6.4 GB (e.g., RTX 4060)
GEMMA3 12B	∼24 GB	32 GB	28 GB (e.g., RTX 4090)
GEMMA3 27B	∼60 GB	32 GB	60 GB (e.g. H100)

high-end multi-GPU infrastructure, which may not be feasible in cost-constrained environments. Moreover, the RAM requirements of the QWEN2.5 72B models (140 GB) pose an additional bottleneck, particularly when scaling to multiple concurrent users or queries, other model series like QWEN3 and GEMMA3 also share similar hardware requirements with larger models. These constraints re-highlight the need for more efficient retrieval strategies that can leverage smaller models while still maintaining competitive retrieval quality. Therefore, PromptPRF is introduced precisely to address this gap.

11.1.2 Query Latency and Computational Efficiency

In dense retrieval applications, the utilization of LLM backbones not only increases memory and storage requirements but also introduces substantially higher inference-time latency. Total query latency is primarily driven by two factors: (i) the encoding of the query into a dense vector using the LLM backbone, and (ii) the subsequent vector search over the dense index, typically performed via k-nearest neighbor (k-NN) or brute-force similarity computation. Among these, the query encoding step represents the primary computational bottleneck, particularly as the model size increases.

Table 11.2 presents the per-query encoding latency measured on NVIDIA H100 GPUs across the QWEN2.5, QWEN3, and GEMMA3 model families. While batching strategies may offer throughput improvements, which still benefit smaller models over larger ones, the baseline latency for single-

Table 11.2: Latency due to query encoding across LLM backbones of increasing size; experiments executed using Nvidia H100. (All models used are instruct models.)

Model	GPU	Query Encoding Latency
QWEN2.5 0.5B	1x H100	11.76 ms/query
QWEN2.5 1.5B	1x H100	15.38 ms/query (+31%)
QWEN2.5 3B	1x H100	20.41 ms/query (+74%)
QWEN2.5 7B	1x H100	29.81 ms/query (+153%)
QWEN2.5 14B	1x H100	41.57 ms/query (+253%)
QWEN2.5 32B	2x H100	117.65 ms/query (+900%)
QWEN2.5 72B	4x H100	281.29 ms/query (+2,292%)
QWEN3 0.6B	1x H100	13.20 ms/query
QWEN3 1.7B	1x H100	19.87 ms/query (+51%)
QWEN3 4B	1x H100	27.45 ms/query (+108%)
QWEN3 8B	1x H100	39.90 ms/query (+202%)
QWEN3 14B	1x H100	55.80 ms/query (+323%)
QWEN3 32B	2x H100	125.67 ms/query (+852%)
GEMMA3 1B	1x H100	44.45 ms/query
GEMMA3 4B	1x H100	74.63 ms/query (+68%)
GEMMA3 12B	1x H100	116.28 ms/query (+162%)
GEMMA3 27B	2x H100	165.63 ms/query (+273%)

stream query encoding scales non-linearly with model size. For example, within the QWEN2.5 family, the 7B Instruct model requires an average of 29.81 milliseconds per query, representing a 153% increase over the 0.5B variant's 11.76 milliseconds. This disparity becomes prohibitive at the largest scales: the QWEN2.5 72B Instruct model incurs a latency of 281.29 milliseconds per query, shows an increase of over 22 times compared to the 0.5B baseline, even when distributing the inference load across four H100 GPUs. Similar trends are observed in the GEMMA3 family, where the 27B model requires 165.63 milliseconds, nearly quadrupling the latency of the 1B variant.

These latency metrics highlight the critical trade-offs inherent in scaling LLM-based retrievers. While larger parameter models typically offer improved semantic understanding, their computational cost renders them impractical for applications demanding real-time or interactive response limits. Conversely, smaller backbones, such as the QWEN2.5 0.5B or QWEN3 0.6B, provide significantly lower latency (under 14 ms), making them highly suitable for deployment in resource-constrained or latency-critical environments. This performance gap motivates the development of strategies such as PromptPRF, which aims to bridge the effectiveness gap without incurring the computational overhead associated with massive-scale retrievers.

11.1.3 Implications for Dense Retrieval

Different dense retrieval architectures integrate LLMs of varying scale; yet, all face the same fundamental trade-off between computational efficiency and retrieval effectiveness. Small and medium-sized LLMs offer lower latency and hardware cost but may underperform in terms of retrieval quality, while large models tend to yield stronger effectiveness but impose substantial computational demands, including increased GPU memory requirements, longer inference time, and higher energy consumption. These constraints are particularly critical in real-world deployments, where cost, responsiveness, and scalability are often tightly coupled.

From a system design perspective, the following considerations typically guide model selection and deployment strategy:

1. Smaller models, such as the QWEN2.5 0.5B or QWEN3 0.6B, should be prioritized for efficiency-critical applications, including conversational search, edge computing, and mobile deployments, where low latency and limited resource availability are defining constraints.

2. Larger parameter models (e.g., QWEN2.5 7B, GEMMA3 27B, and above) are suitable primarily when the anticipated improvement in retrieval quality justifies the significant increase in computational cost and latency, such as in offline batch processing or high-precision expert systems.

3. The selection of PRF strategies must be aligned with deployment constraints, emphasizing the offline generation of feedback signals to mitigate online inference latency and reduce real-time system load.

Motivated by these constraints, we seek to bridge the performance gap between large-scale LLM-based retrievers and lightweight, resource-efficient alternatives. To this end, we propose PromptPRF, a pseudo-relevance feedback strategy that enhances retrieval effectiveness while minimizing latency and inference-time computation. Unlike existing LLM-based PRF methods that rely on query-dependent, on-the-fly feature generation and prompt encoding, PromptPRF shifts the feedback generation stage entirely offline. This design decouples the most computationally intensive components of PRF from the online retrieval process, enabling integration without incurring substantial latency overhead. PromptPRF also allows smaller LLM-based dense retrievers to approach or surpass the performance of much larger models without PRF, all while preserving the efficiency advantages critical to real-time and resource-constrained deployments. As detailed in the next section, this approach enables scalable, effective retrieval across a wide range of infrastructure settings.

11.2 PromptPRF - Pseudo Relevance Feedback in Zero-shot Large Language Model Retrieval

Our goal is to develop a PRF method that improves retrieval effectiveness while maintaining computational efficiency. Specifically, we seek to reduce the effectiveness gap between small and large dense retrievers without introducing additional hardware (GPU) burdens in the online phase or increasing query-time latency. This is achieved by offloading the generative component of the PRF pipeline, i.e. the creation of feedback features, from the online to the offline stage, thereby eliminating the need for query-time generation.

While showing similarities in terms of the features that are used, PromptPRF is different from prior LLM-based PRF approaches such as GRF [139], in that these previous methods require online, query-dependent generation of feedback content. Such approaches are impractical in latency-sensitive or resource-constrained environments. Instead, PromptPRF generates query-independent features from passages offline, enabling efficient integration during inference without incurring runtime generation costs.

PromptPRF is zero-shot and it is developed within the PromptReps [250] framework, a zero-shot dense retrieval approach that uses prompt-based query encoding without the need for supervised fine-tuning. In our experiments, PromptReps is used for both the initial and second-stage retrieval steps, with PromptPRF introducing a refinement stage in between. The extracted feedback features, which are augmented with rank information, are then injected into a prompt that guides the re-encoding of the query, producing a refined dense representation for the second retrieval stage.

PromptPRF comprises four modular components:

1. **Offline Feature Extraction (Section 11.2.1):** Structured and unstructured features are generated from passages using decoder-style LLMs at passage indexing time.

2. **Initial Retrieval (Section 11.2.2):** An initial ranked list is obtained using the PromptReps framework.

3. **Query Refinement (Section 11.2.3):** Feedback features and explicit rank metadata are incorporated into a modified prompt to re-encode the query.

4. **Second-Stage Retrieval (Section 11.2.4):** The refined query embedding is used to retrieve a final ranking from the same dense index.

While our experimental validation is conducted within the PromptReps dense retrieval framework, the architectural design of PromptPRF suggests potential adaptivity to other dense retrievers that support prompt-based representation generation. The following subsections describe each component in detail.

Table 11.3: Prompts used for passage-based feature generation tasks. Each template uses `{passage}` (replace with the actual text) followed by a task instruction (e.g., summary, keywords/entities, or document generation). A per-feature token limit controls cost, runtime, and verbosity (higher for generation, lower for extraction). Newlines (`\n`) separate the passage from the instruction. COT stands for Chain-of-Thought.

Feature	Prompt	Max Tokens
Document	`Passage: {passage}\n\nBased on the passage, generate a similar passage:`	512
Essay	`Passage: {passage}\n\nBased on the passage, write an essay:`	512
News Article	`Passage: {passage}\n\nBased on the passage, write a news article:`	512
Summary	`Passage: {passage}\n\nBased on the passage, write a summary:`	256
Facts	`Passage: {passage}\n\nBased on the passage, generate a bullet-point list of relevant facts:`	256
Keywords-COT	`Passage: {passage}\n\nBased on the passage, generate a bullet-point list of relevant keywords. Next to each point, briefly explain why:`	256
Entities-COT	`Passage: {passage}\n\nBased on the passage, generate a bullet-point list of relevant entities. Next to each point, briefly explain why:`	256
Query Keyword	`Passage: {passage}\n\nBased on the passage, generate a bullet-point list of diverse keyword queries that will find this passage:`	256
Keywords	`Passage: {passage}\n\nBased on the passage, generate a bullet-point list of relevant keywords:`	64
Entities	`Passage: {passage}\n\nBased on the passage, generate a bullet-point list of relevant entities:`	64

11.2.1 Stage 1: Offline Feature Extraction

In PromptPRF, we generate passage-level features from the original passages. To generate features we prompt an LLM backbone with an instruction and the passage – we refer to this backbone as the *feature extractor*. Prompts for feature generation are provided in Table 11.3. Inspired by GRF [139], we instruct the feature generator to produce features belonging to the feature types $T = \{$Keywords, Entities, Summaries, Essays, Keywords-COT, Entities-COT, News-Articles, Facts, Query-Keywords, and Documents$\}$[1]. For each feature type $t \in T$, let I_t denote the instruction or prompt template designed to guide the LLM in generating feature t. The feature generation for passage p, i.e. $f_t^{(p)}$ can be described as $f_t^{(p)} = \text{LLM}(text(p), I_t)$.

[1] COT refers to Chain-of-Thoughts.

Then the feature collection for the passage p_j is obtained as:

$$\mathcal{F}_j = \bigcup_{t \in T} \left\{ f_t^{(j)} \right\} = \left\{ f_t^{(j)} \mid t \in T \right\} \tag{11.1}$$

Given that feature generation is independent of the query, this process can be executed entirely offline. Consequently, it introduces no additional latency during query time and incurs no computational overhead within the online retrieval pipeline. In our experiments, we analyze the impact of different LLM sizes on feature extraction quality, comparing models ranging from compact architectures (e.g., QWEN2.5 0.5B) to large-scale backbones (e.g., QWEN2.5 72B).

Although the feature generation is inspired by the GRF [139], we slightly altered the original prompts provided by GRF. Our feature generation prompts differ from the original ones by necessity, not by design preference. GRF utilise query-dependent prompts (e.g., "write an esssay based on the query"), which requires expansive online generation at retrieval time. To achieve offline efficiency, PromptPRF must utlise query-independent prompts for feature generation (e.g., "write an essay based on the passage"). This structural modification allows us to pre-compute all features during indexing. While this contributes to the effectiveness gap compared to GRF, as the generated features are not tailored to the specific user query, it is the critical component for the efficiency gains of PromptPRF introduced in this chapter.

11.2.2 Stage 2: Initial Retrieval

We employ the dense retrieval component of PromptReps [250], which has a bi-encoder structure consisting of E_D and E_Q and operates in a zero-shot manner without additional training. Given a query q PromptReps generates a dense representation $\mathbf{q}$ by prompting an LLM-based E_Q to summarize the query into a single word and extract its hidden representation. The same process is used to generate passage embeddings $\mathbf{p_i}$ using E_D; typically E_Q and E_D are based on the same backbone and they differ in the prompt. Relevance is estimated using cosine similarity: $s(q, p_i) = \frac{\mathbf{q} \cdot \mathbf{p_i}}{|\mathbf{q}||\mathbf{p_i}|}$, where $\mathbf{p_i}$ is the dense representation of the ith passage p_i. After applying the similarity function to each passage in the collection in response to a query, we obtain a ranked list $R = \{p_1, p_2, ..., p_k\}$ according to decreasing relevance score $s(q, p_k)$, where k is the rank of each passage.

11.2.3 Stage 3: Query Refinement with PRF

To implement PromptPRF, we modify the PromptReps query encoder E_Q to obtain a new encoder E_F which is responsible for encoding the query along with the extracted passage features. For this, we modify the original PromptReps prompt template, which encodes a query in isolation. We modify this prompt template to explicitly inject the top-ranked passages (or their offline extracted features) and rank information (as detailed in Figure 11.1), enabling the LLM to attend to the feedback signals during the query encoding step, which allows for enhanced contextualization of the query representation.

```
[
  {"role": "system", "content":
    "You are an AI assistant that can understand human language."},
  {"role": "user", "content":
    'Query: "{QUERY}".\n
     {FEATURE NAME} for top {RANK} Retrieved Passage: "{PASSAGE OR FEATURE CONTENT}".
     ...
     {FEATURE NAME} for top {RANK} Retrieved Passage: "{PASSAGE OR FEATURE CONTENT}".\n
     Use one word to represent the query and the top passages in a retrieval task.
     Make sure your word is in lowercase."
  },
  {"role": "assistant", "content": 'The word is: "'}
]
```

Figure 11.1: Prompt format with rank information.

Formally, the refined query representation $\mathbf{q}'$ is obtained as:

$$\mathbf{q}' = \mathrm{E}_F\left(\mathrm{Template}(q) \oplus \left(\bigoplus_{j=1}^{k} \bigoplus_{f \in \mathcal{F}_j} \text{“Rank } j : \beta_i f_t^{(j)}\text{”}\right)\right) \tag{11.2}$$

where β_i is a hyper-parameter that controls the contribution of each passage's extracted feature and $\mathrm{Template}(q)$ refers to the query-specific prompt template used in this work, and reported in the Table 11.3. Along with features f, we also include information about the rank of the passage from the first round of retrieval (Rank j).

In this chapter, we set $\beta_i = 1$ and evaluate feature types in isolation to disentangle the specific contribution of each semantic signal (e.g., Keywords vs. Facts) to the retrieval process. Combining multiple feature types introduces confounding variables that would obscure the analysis of which specific feedback signals drive performance gains. We reserve the exploration of feature fusion for future work once the efficacy of individual signals is established.

11.2.4 Stage 4: Second Retrieval

The refined query representation $\mathbf{q}'$, which incorporates feedback signals from the top-ranked passages, is then used to perform a second stage of retrieval over the same dense index. The updated query representation reflects both the original query intent and the contextual signals derived from pseudo-relevant passages. The relevance score $s(q', p)$ is computed as the cosine similarity between the refined query embedding $\mathbf{q}'$ and the dense representation of each passage $\mathbf{p}$:

$$s(q', p) = \frac{\mathbf{q}' \cdot \mathbf{p}}{|\mathbf{q}'||\mathbf{p}|} \tag{11.3}$$

This scoring function maintains consistency with the initial retrieval stage. By reusing the same dense retrieval infrastructure, PromptPRF introduces no additional index-side complexity.

11.2.5 Relationship with Other Approaches

PromptPRF is conceptually related to a line of work that uses LLMs for PRF or query expansion, but differs in several key aspects.

GenPRF [217] prompts a FlanT5 [23] model to generate expansion terms directly from the query. While both GenPRF and PromptPRF utilize LLMs to generate feedback features, GenPRF is restricted to short expansion phrases and does not incorporate information from retrieved passages. In contrast, PromptPRF generates a diverse set of structured and unstructured features from retrieved passages, capturing richer contextual signals. Finally, GenPRF requires LLM inferences at query time.

Query2doc [209] is another query-dependent approach that generates synthetic pseudo-documents from the input query, thus requiring LLM inferences at query time. PromptPRF also includes document-style features, but instead of generating from the query, it prompts the LLM to generate from retrieved passages, making it passage-driven and query-independent.

GRF [139] is the most similar to our method in spirit, as it uses LLMs to produce structured features like Summaries and Entities. In fact, we rely on modifications of GRF's feature sets and original prompts. However, there are three key differences: (1) GRF performs query-time, query-dependent feature generation, whereas PromptPRF performs offline, passage-based generation; (2) GRF integrates features into the query for BM25-based sparse retrieval, while PromptPRF is designed for dense retrieval with learned or prompt-based embeddings; and (3) although PromptPRF borrows feature categories and feature generation prompts from GRF, we use our own prompts and incorporate passage rank metadata into the query refinement process.

11.2.6 Retriever-Agnostic Feedback Generation

PromptPRF operates independently of any specific retrieval architecture and is not tightly coupled with PromptReps [250] or any particular embedding strategies. Although we instantiate it within a PromptReps-style retriever for experimental consistency and to exploit its zero-shot nature, thus rendering a fully zero-shot pipeline, the method is broadly applicable to any dense retrieval framework capable of processing prompt-augmented queries.

11.3 Empirical Evaluation

We conducted a series of comprehensive empirical evaluations of PromptPRF to assess its effectiveness, efficiency, and generalisability across dense retrieval tasks. The goal is to investigate the high level research question of this chapter:

RQ3.2: Can we make LLM-based Pseudo-Relevance Feedback more effective and efficient?

To address this question in a more structured way, we define the following sub-research questions, each aligned with a specific experimental focus:

- **RQ3.2.1:** Does PromptPRF improve retrieval effectiveness compared to no feedback and passage-based feedback?

- **RQ3.2.2:** How does the choice of feature extractor affect PromptPRF's performance?

- **RQ3.2.3:** What is the impact of feature type and feedback depth on retrieval effectiveness, and how can they be tuned to balance quality and stability?
- **RQ3.2.4:** Does PromptPRF generalize to out-of-domain datasets?
- **RQ3.2.5:** Does incorporating rank information into the feedback prompt contribute to more effective query refinement?
- **RQ3.2.6:** What are the computational costs and efficiency trade-offs of PromptPRF compared to baseline methods?

The remainder of this section presents the datasets, evaluation metrics, baselines, and feature extraction models used. In the next section, we present the results and analyses structured according to the sub-questions above. Each subsection is grounded with experimental outcomes and aims to quantify both the benefits and limitations of PromptPRF under varying retrieval conditions.

11.3.1 Datasets

We evaluate PromptPRF on two popular benchmark families same as in Chapter 10: the TREC Deep Learning (TREC DL) tracks from 2019 and 2020, and the BEIR benchmark. These datasets enable a comprehensive evaluation for both in-domain and out-of-domain retrieval scenarios to validate the generalisability of our PromptPRF. For BEIR benchmark, we focus on a representative subset of 8 datasets commonly adopted in dense and generative retrieval research: ArguAna, FiQA-2018, NFCorpus, Quora, SCIDOCS, SciFact, TREC-COVID, and Touche-2020. While some of these datasets (e.g., TREC-COVID, FiQA) provide curated relevance judgments, others rely on distant supervision or heuristic signals. This diversity makes BEIR a suitable testbed for evaluating generalization under zero-shot and out-of-domain retrieval settings. Unlike domain-specific evaluations such as TREC DL, BEIR enables us to assess the robustness of PromptPRF across a broad range of retrieval scenarios without requiring task-specific fine-tuning.

11.3.2 Evaluation Metrics

We evaluate retrieval effectiveness using nDCG@10, the standard metric for both the TREC DL and BEIR benchmarks. This choice aligns with the protocol established in our primary baseline, the PromptReps framework. nDCG@10 captures both the relevance and position of retrieved passages within the top-10 results; by rewarding systems that rank highly relevant passages closer to the top, it is well-suited for evaluating high-precision ranking systems.

In the case of TREC DL 2019 and 2020, relevance judgments are graded, enabling metrics like nDCG@10 to capture fine-grained distinctions between highly relevant and marginally relevant passages. In contrast, most datasets in BEIR adopt binary relevance labels, where documents are either relevant or not. Despite this difference, we use nDCG@10 across both benchmarks to evaluate retrieval effectiveness for consistency.

nDCG@10 is particularly suitable for evaluating PRF methods, as it reflects the quality of the top-ranked results, which are directly affected by query reformulation. All reported scores are computed using the official evaluation scripts provided by BEIR and TREC, ensuring comparability with prior work. Statistical significance is assessed using a two-tailed paired t-test with Bonferroni correction applied where appropriate. Results are considered significant at $p < 0.05$.

11.3.3 Baselines

We evaluate PromptPRF within the PromptReps [250] framework using LLMs from several model families with varying size and capacity. Our primary goal is to determine whether PromptPRF can enable smaller retrievers to achieve retrieval quality comparable to larger models, without introducing additional training or query-time computational overhead (beyond a second step of retrieval / re-ranking).

QWEN and GEMMA Models

We utilize a comprehensive set of instruction-tuned models from the QWEN and GEMMA families as our primary PromptReps base retrievers. Specifically, we evaluate the QWEN2.5 series[2] [199] (ranging from 0.5B to 72B parameters), the QWEN3 series[3] [228] (0.6B to 32B), and the GEMMA3 series[4] [198] (1B to 27B). These models represent a diverse spectrum of dense retrieval backbones with varying parameter counts and resource demands. The smaller variants (e.g., QWEN2.5 0.5B, GEMMA3 1B) serve as key baselines to investigate whether PRF-enhanced lightweight models can approximate or surpass the effectiveness of significantly larger-scale models (e.g., QWEN2.5 72B, GEMMA3 27B) without PRF.

Comparative Baselines

In addition to No PRF baselines, we compare PromptPRF against a Passage PRF baseline, which directly concatenates the content of top-ranked passages into the query prompt. This baseline serves as a strong reference point for evaluating whether structured feedback signals generated offline by LLMs (as in PromptPRF) are more effective than simple content-level feedback.

Together, these baselines enable a detailed analysis of PromptPRF's capacity to improve retrieval performance across various model sizes and architectures while preserving or improving computational efficiency.

11.3.4 Feature Extraction Models

In PromptPRF, LLMs are employed to generate ten distinct feedback features from top-ranked passages: `Summary`, `Keywords`, `Keywords-COT`, `Entities`, `Entities-COT`, `News-Article`, `Facts`,

[2] `https://huggingface.co/collections/Qwen/qwen25`
[3] `https://huggingface.co/collections/Qwen/qwen3`
[4] `https://huggingface.co/collections/google/gemma-3-release`

`Query-Keywords`, `Document`, and `Essay`. These features are generated offline and incorporated into the query via specific prompting strategies during inference. Although the same LLM architectures are utilized as dense retrievers within the PromptReps framework, their role in feature extraction is distinct: here, they serve purely as generative models conditioned on passage content, independent of the input query.

We evaluate a broad set of decoder-style LLMs as feature extractors on both TREC DL datasets, utilizing the same QWEN2.5, QWEN3, and GEMMA3 model families described in the previous section. This diverse selection allows us to assess feature generation quality across a wide range of parameter scales, from sub-billion parameter models to large-scale 72B backbones.

By disentangling the retrieval and feature generation roles, we can systematically examine how PromptPRF scales with model size and determine the extent to which smaller extractors produce high-quality feedback suitable for dense retrieval. Utilizing compact extractors during the offline passage indexing phase offers significant advantages, including reduced computational costs, faster indexing throughput, and lower infrastructure requirements compared to relying on larger foundation models.

11.4 Results

This section presents empirical results evaluating PromptPRF across a range of retrieval conditions and configurations. We examine its impact on retrieval effectiveness, model scaling, feature design, and cost-efficiency. Results are organized to address the sub-research questions outlined earlier in Section 11.3, beginning with the effectiveness of PromptPRF on TREC DL datasets and followed by analyses on feature extractors, feedback configurations, effectiveness with BEIR benchmark, and efficiency trade-offs.

11.4.1 PromptPRF Effectiveness on TREC DL

To evaluate whether PromptPRF improves retrieval effectiveness over baselines and traditional feedback strategies in order to address **RQ3.2.1: Does PromptPRF improve retrieval effectiveness compared to no feedback and passage-based feedback?**, we compare it against two configurations: (i) No PRF, i.e., standard PromptReps without feedback, and (ii) Passage PRF, where top-ranked passages are directly appended to the prompt. We report the most effective configurations of PromptPRF for dense retriever model families on the TREC DL19 and DL20 benchmarks, using nDCG@10 as the evaluation metric (see Tables 11.4 and 11.5).

Effectiveness Gains for Small Retrievers

Across both datasets, PromptPRF provides significant improvements over both No PRF and Passage PRF baselines, with the most substantial gains observed for smaller retriever architectures. On TREC DL19, the QWEN2.5 7B retriever using `Facts` features (extracted by the smaller QWEN2.5 1.5B

Table 11.4: Retrieval effectiveness (nDCG@10) on TREC DL 2019 with best PromptPRF configurations. We compare the base retriever (No PRF) against Passage PRF and PromptPRF. The best result per model row is **bolded**. Statistical significance ($p < 0.05$) over No PRF is denoted by † (for Passage PRF) and ‡ (for PromptPRF).

Model	**Size**	**No PRF**	**Passage PRF**	**PromptPRF**	**Best Config.** (Ext. / Feat. / Depth)
Traditional Baselines					
BM25	–	0.4973	–	–	–
BM25 + RM3	–	–	0.5156	–	– / – / 3
ANCE-PRF	–	–	0.6807	–	– / – / 3
Vector-PRF	–	–	0.6629	–	– / – / 3
ColBERT-PRF	–	–	0.7276	–	– / – 3
QWEN2.5 Family					
QWEN2.5	0.5B	0.2835	**0.3366**	0.3345	Q2.5 7B / Facts / 5
QWEN2.5	1.5B	0.4491	0.4607	**0.4764**	Q2.5 1.5B / Keywords / 3
QWEN2.5	3B	0.3202	0.3261	**0.4103**‡	Q2.5 0.5B / Facts / 20
QWEN2.5	7B	0.3292	0.4269†	**0.4584**‡	Q2.5 1.5B / Facts / 5
QWEN2.5	14B	0.4296	0.5484†	**0.6057**‡	Q2.5 0.5B / Keywords / 20
QWEN2.5	32B	**0.4363**	0.3083	0.4002	Q2.5 1.5B / Keywords / 3
QWEN2.5	72B	0.5931	**0.5992**	0.5912	Q2.5 3B / Entities / 10
QWEN3 Family					
QWEN3	0.6B	**0.3053**	0.2123	0.2838	Q3 1.7B / Keywords-COT / 1
QWEN3	1.7B	0.2871	0.3215	**0.3553**‡	Q3 4B / Essay / 1
QWEN3	4B	0.5343	0.5505	**0.5963**‡	Q3 32B / Keywords / 3
QWEN3	8B	0.5422	0.5811†	**0.5941**‡	Q3 1.7B / Facts / 5
QWEN3	14B	0.5430	0.5609	**0.5743**	Q3 32B / Facts / 5
QWEN3	32B	0.5202	0.5457	**0.5631**	Q3 32B / Essay / 5
GEMMA3 Family					
GEMMA3	270M	0.2386	0.2562	**0.2784**‡	G3 4B / Keywords / 1
GEMMA3	1B	0.1881	0.2374†	**0.2624**‡	G3 1B / Facts / 5
GEMMA3	4B	0.4267	0.4254	**0.5025**‡	G3 270M / Entities / 1
GEMMA3	12B	0.5331	0.5567	**0.5810**‡	G3 270M / Keywords / 1
GEMMA3	27B	**0.5652**	0.5380	0.5589	G3 12B / Keywords / 1

model) at PRF depth 5 achieves an nDCG@10 of 0.4584, significantly outperforming its No PRF baseline (0.3292). This represents a remarkable absolute gain of +12.9 points. Impressively, this performance surpasses the significantly larger QWEN2.5 32B No PRF baseline (0.4363), indicating that PromptPRF can enable a compact retriever to exceed the effectiveness of a model with more than four times the parameter count. Similarly, the very lightweight GEMMA3 1B model sees a massive relative improvement of roughly 39%, jumping from 0.1881 (No PRF) to 0.2624 (PromptPRF).

On TREC DL20, a similar pattern is observed. When applying `Keywords-COT` features generated by the QWEN3 1.7B extractor at PRF depth 5, the QWEN3 4B retriever's effectiveness increases to 0.5088, significantly outperforming both the No PRF (0.4028) and Passage PRF (0.4459) baselines.

This configuration not only narrows the gap with larger models but explicitly outperforms the QWEN3 32B No PRF baseline (0.4456) by over 6 points. These results illustrate the generalizability of PromptPRF across test collections and model families, demonstrating that even features generated by smaller or mid-sized extractors can yield transformative improvements.

It is also worth mentioning that these effectiveness gains are achieved without modifying the retriever architecture or requiring additional online compute. All PromptPRF features are generated offline and injected at query time with minimal runtime cost. Thus, PromptPRF offers a viable strategy to strengthen small retrievers in latency- or resource-constrained environments, allowing the deployment of models that combine high retrieval quality with lower infrastructure demands.

These findings show that PromptPRF effectively unlocks the latent potential of lightweight dense retrievers. With suitable feedback features and carefully chosen depth settings, smaller models like QWEN2.5 7B or QWEN3 4B can deliver competitive performance, challenging the assumption that only large-scale retrievers can achieve high effectiveness in zero-shot settings.

Effectiveness Gains for Medium Retrievers

PromptPRF also yields substantial benefits for medium-scale retrievers, although the relative improvements are naturally more moderate compared to smaller models due to the higher initial baselines. On TREC DL19, the QWEN2.5 14B retriever achieves its best performance using `Keywords` features generated by the compact QWEN2.5 0.5B extractor at PRF depth 20. This configuration reaches an nDCG@10 of 0.6057, dramatically outperforming its No PRF baseline (0.4296). More specifically, this score surpasses the performance of the massive QWEN2.5 72B retriever without feedback (0.5931). The ability of PromptPRF to push a 14B parameter model beyond the capability of a 72B parameter backbone demonstrates its effectiveness in reinforcing semantic content and enriching query representations, even when the base retriever already possesses considerable representational capacity.

Similarly, on TREC DL20, the QWEN2.5 14B retriever again demonstrates this capability. When equipped with `Entities` features generated by the QWEN2.5 7B extractor at PRF depth 20, it achieves an nDCG@10 of 0.4960. This not only significantly improves over its own baseline (0.3822) but also exceeds the QWEN2.5 72B No PRF baseline (0.4826). We observe consistent trends in the GEMMA3 family as well; on TREC DL19, the GEMMA3 12B model with PromptPRF (0.5810) outperforms the larger GEMMA3 27B baseline (0.5652). The depth settings in these cases (often depth 20) illustrate that deeper feedback can be beneficial for higher-capacity retrievers, provided the features remain compact and highly aligned with the query intent.

These findings suggest that PromptPRF enhances retrieval even when applied to strong, medium-sized retrievers, with improvements driven not solely by scale but by the quality and format of the feedback signal. The results highlight PromptPRF's ability to extract additional retrieval utility from models like QWEN2.5 14B and GEMMA3 12B, pushing their performance into a range that approaches or exceeds much larger models without requiring additional online inference cost.

Table 11.5: Retrieval effectiveness (nDCG@10) on TREC DL 2020 with best PromptPRF configurations. We compare the base retriever (No PRF) against Passage PRF and PromptPRF. The best result per model row is **bolded**. Statistical significance ($p < 0.05$) over No PRF is denoted by † (for Passage PRF) and ‡ (for PromptPRF).

Model	**Size**	**No PRF**	**Passage PRF**	**PromptPRF**	**Best Config.** (Ext. / Feat. / Depth)
			Traditional Baselines		
BM25	–	0.4769	–	–	–
BM25 + RM3	–	–	0.5043	–	– / – / 3
ANCE-PRF	–	–	0.6948	–	– / – / 3
Vector-PRF	–	–	0.6598	–	– / – / 3
ColBERT-PRF	–	–	0.6958	–	– / – / 3
			QWEN2.5 Family		
QWEN2.5	0.5B	0.2067	0.2398	**0.2688**‡	Q2.5 14B / Facts / 1
QWEN2.5	1.5B	0.2857	0.3427†	**0.3541**‡	Q2.5 32B / Facts / 20
QWEN2.5	3B	0.2819	0.3071	**0.3390**‡	Q2.5 3B / Summary / 5
QWEN2.5	7B	0.2147	0.3290†	**0.3439**‡	Q2.5 7B / Summary / 20
QWEN2.5	14B	0.3822	0.4670†	**0.4960**‡	Q2.5 7B / Entities / 20
QWEN2.5	32B	**0.3680**	0.2653	0.3599	Q2.5 72B / Entities / 3
QWEN2.5	72B	0.4826	0.5180	**0.5215**	Q2.5 7B / Entities / 3
			QWEN3 Family		
QWEN3	0.6B	**0.2249**	0.1793	0.2194	Q3 4B / Key-COT / 1
QWEN3	1.7B	0.2574	0.2782†	**0.3379**‡	Q3 4B / Summary / 3
QWEN3	4B	0.4028	0.4459†	**0.5088**‡	Q3 1.7B / Key-COT / 5
QWEN3	8B	0.4960	0.5210	**0.5265**‡	Q3 4B / Summary / 10
QWEN3	14B	0.4699	0.4887	**0.5067**	Q3 0.6B / Keywords / 3
QWEN3	32B	0.4456	0.4788	**0.5036**‡	Q3 0.6B / Key-COT / 1
			GEMMA3 Family		
GEMMA3	270M	0.1138	0.1172	**0.1226**	G3 270M / Summary / 3
GEMMA3	1B	0.1132	0.1473	**0.1713**‡	G3 12B / Keywords / 3
GEMMA3	4B	0.2717	0.3224†	**0.3731**‡	G3 270M / Q-Key / 3
GEMMA3	12B	0.4354	**0.4875**	0.4792	G3 12B / Entities / 3
GEMMA3	27B	0.4849	0.4961	**0.5409**‡	G3 12B / Entities / 1

Implications

These findings indicate that PromptPRF enhances effectiveness across a wide set of retriever sizes and feedback feature variants. The performance improvements are statistically significant in the majority of cases and robust across both TREC DL benchmarks. Notably, significant relative gains are observed for lightweight retrievers, such as the GEMMA3 1B and QWEN2.5 3B, for which PromptPRF improves performance to levels comparable with, and in several instances exceeding, significantly larger models. For example, on TREC DL19, the QWEN3 4B model with PromptPRF (0.5963) clearly outperforms

the QWEN3 32B model without feedback (0.5202). This demonstrates that PromptPRF is not merely an incremental enhancement but a framework that challenges the strict dependence on parameter scaling for retrieval effectiveness.

Furthermore, the ability of PromptPRF to predominantly outperform the Passage PRF baseline illustrates the limitations of injecting unstructured, verbose raw passages into the query prompt. While Passage PRF provides a strong baseline, which occasionally outperforming PromptPRF in isolated cases like the QWEN2.5 0.5B on DL19, PromptPRF typically provides superior results by leveraging semantically distilled and query-independent features. These structured signals integrate more effectively into query reformulation, resulting in cleaner and more targeted query embeddings that are less prone to noise.

Together, these results highlight the practical utility of PromptPRF: it enables scalable, modular, and cost-aware retrieval pipelines that maintain competitive performance while preserving efficiency. This makes PromptPRF especially well-suited for production use in constrained environments, bridging the effectiveness gap between compact and large-scale models without the associated computational penalty.

Summary

Overall, the empirical evidence presented in this section provides an affirmative answer to **RQ3.2.1**. PromptPRF consistently outperforms both the No PRF and Passage PRF baselines on TREC DL19 and DL20 across a diverse range of retriever architectures, including the QWEN2.5, QWEN3, and GEMMA3 families. These improvements are not only statistically significant in the majority of configurations but also practically meaningful, demonstrating the capacity of PromptPRF to bridge the performance gap between lightweight and large-scale retrievers. The results validate PromptPRF as an effective and efficient feedback strategy that enhances dense retrieval quality without incurring the query-time latency costs associated with online generation. By pairing compact retrievers with concise, semantically focused feedback features, PromptPRF offers a robust solution adaptable to various model capacities, making it a compelling choice for both research and deployment in real-world retrieval systems.

11.4.2 Impact of Feature Extractor on TREC DL

To investigate how the choice of feature extraction model influences retrieval quality and address **RQ3.2.2: How does the choice of feature extractor affect PromptPRF's performance?**, we evaluate PromptPRF using fixed PRF depths and feature types based on the best-performing configurations identified for TREC DL19 and DL20 (as detailed in Tables 11.4 and 11.5). Figures 11.2 and 11.3 present the nDCG@10 scores as heatmaps, visualizing the performance interaction between varying Feature Extractor sizes (x-axis) and Retriever sizes (y-axis) across the QWEN2.5, QWEN3, and GEMMA3 model families.

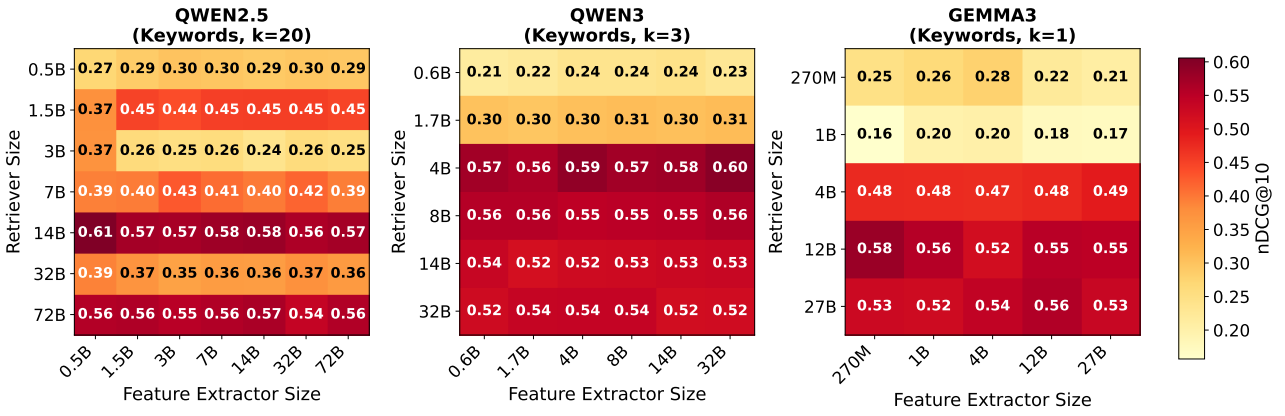


Figure 11.2: Impact of feature extractor size on nDCG@10 for TREC DL19 across three model families. Each heatmap shows retriever size (rows) versus feature extractor size (columns) using the optimal feature type and PRF depth per family: QWEN2.5 (Keywords, k=20), QWEN3 (Keywords, k=3), and GEMMA3 (Keywords, k=1). Darker colors indicate higher performance.

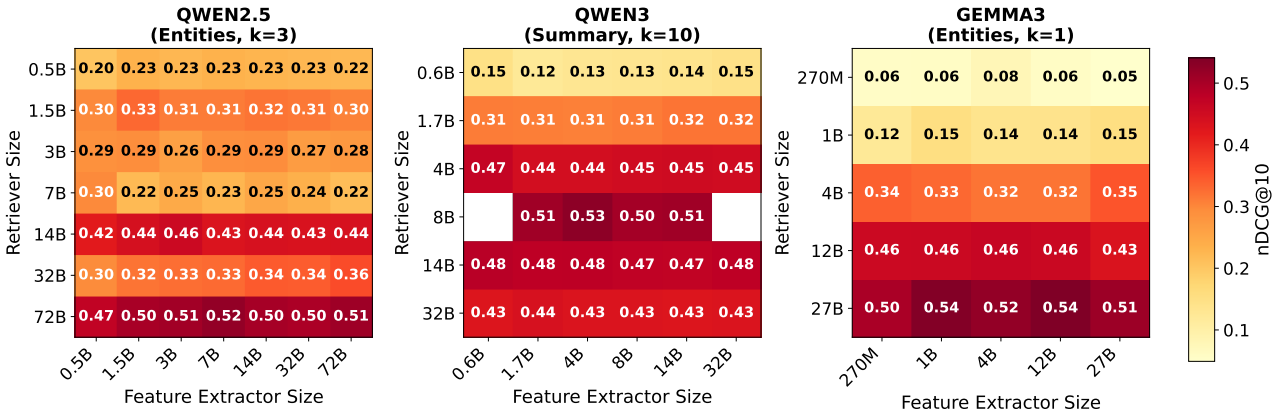


Figure 11.3: Impact of feature extractor size on nDCG@10 for TREC DL20 across three model families. Each heatmap shows retriever size (rows) versus feature extractor size (columns) using the optimal feature type and PRF depth per family: QWEN2.5 (Entities, k=3), QWEN3 (Summary, k=10), and GEMMA3 (Entities, k=1). Darker colors indicate higher performance.

Effectiveness

The results presented in the heatmaps reveal a distinct pattern regarding the interaction between feature extractor size and retriever capacity. Unlike the vertical axis (Retriever Size), where increasing model scale yields predictable and substantial gains in nDCG@10, the horizontal axis (Feature Extractor Size) demonstrates more stability.

For smaller retrievers, such as the QWEN2.5 0.5B (Figure 11.2, first left), we observe a slight benefit when scaling up the feature extractor. Performance improves from 0.27 (using a 0.5B extractor) to 0.30 (using a 3B extractor). However, this improvement plateaus quickly; utilizing significantly larger extractors (e.g., 72B) does not yield further gains, suggesting that the representational bottleneck lies within the retriever's ability to encode the signal, rather than the extractor's ability to generate it.

For larger and more capable retrievers, the choice of feature extractor becomes even less critical. As illustrated by the QWEN2.5 72B and GEMMA3 27B rows in both figures, the performance remains virtually constant across the horizontal axis. For instance, on TREC DL19, the QWEN2.5 72B retriever

achieves an nDCG@10 of 0.56 regardless of whether it is paired with a matching 72B extractor or a tiny 0.5B extractor. Similarly, on TREC DL20 (Figure 11.3), the GEMMA3 27B retriever maintains scores between 0.50 and 0.54 across all extractor sizes. This "horizontal homogeneity" indicates that even compact, efficient models are capable of generating the specific feedback features (e.g., Keywords, Entities) required to guide high-capacity dense retrievers effectively.

These findings challenge the assumption that larger generative models are strictly necessary for high-quality pseudo-relevance feedback. Instead, the results suggest that the semantic quality of structured features like keywords or entities saturates early. Consequently, mid-sized or even small extractors (e.g., QWEN2.5 1.5B or GEMMA3 4B) offer the optimal trade-off. They provide sufficient semantic enrichment to maximize the performance of both small and large retrievers without the heavy computational cost associated with large-scale generative inference. This insight confirms that PromptPRF is highly modular: one can deploy a lightweight, cost-effective model for offline feature extraction without compromising the online performance of a state-of-the-art dense retriever.

Efficiency

Because PromptPRF performs feature extraction offline, the primary efficiency consideration lies in the one-off computational cost of LLM inference during the preprocessing (indexing) phase. This design choice decouples feedback generation from real-time query execution, making the selection of the feature extractor a critical point of trade-off between offline resource usage and retrieval effectiveness. The heatmaps in Figures 11.2 and 11.3 demonstrate a phenomenon of surprising returns: massive extractors, such as QWEN2.5 72B, do not consistently provide superior performance compared to their significantly smaller counterparts.

In the vast majority of configurations, compact to mid-sized extractors such as GEMMA3 270M, QWEN3 0.6B, or QWEN2.5 1.5B, they not only offer competitive results but often match the effectiveness of the largest available models. For instance, in the GEMMA3 family on TREC DL19, the 4B retriever achieves an nDCG@10 of 0.48 regardless of whether it is paired with a 270M parameter extractor or a 27B parameter extractor. This indicates that the smaller models are capable of generating features (e.g., Keywords, Entities) that are sufficiently informative to support robust query reformulation, without incurring the prohibitive inference costs and memory requirements associated with 70B+ parameter models.

This observation has profound implications for practical deployment and index construction. In resource-constrained environments, utilizing a 0.5B or 1.5B parameter model for feature extraction offers a compelling trade-off: it drastically reduces the time and energy required to process large document corpora offline while maintaining strong online retrieval effectiveness. For system designers, this means that high-quality pseudo-relevance feedback is accessible without requiring clusters of enterprise-grade GPUs for the indexing process.

The results re-highlight a key design insight of PromptPRF: its flexibility allows the system to scale both up and down, depending on available infrastructure and application constraints. Effective PromptPRF configurations can be achieved without relying on the largest and most expensive models,

supporting a modular and cost-aware retrieval pipeline suitable for a wide range of operational scenarios.

Summary

These results directly address **RQ3.2.2** and demonstrate that the effectiveness of PromptPRF is not linearly dependent on the size of the feature extractor. Instead, we observe a phenomenon of rapid saturation, where compact to mid-scale extractors (e.g., QWEN2.5 0.5B or GEMMA3 270M) frequently match the utility of significantly larger models. This trend is particularly evident when using strong retrievers, where the specific choice of extractor size has negligible impact on final retrieval performance. Consequently, larger extractors often incur higher offline computational costs without yielding proportional gains in effectiveness. PromptPRF thus offers significant architectural flexibility: it enables high-quality retrieval by pairing lightweight, efficient extractors with powerful retrievers, rather than relying on brute-force scaling of both components. This adaptability makes PromptPRF suitable for a wide range of deployment settings, allowing system designers to optimize the offline indexing pipeline for efficiency without compromising online retrieval quality.

11.4.3 Impact of Feature Types and PRF Depth on TREC DL

To answer **RQ3.2.3: What is the impact of feature type and feedback depth on retrieval effectiveness, and how can they be tuned to balance quality and stability?**, we analyze how retrieval effectiveness varies with the choice of feature types and PRF depths within the PromptPRF framework.

Figures 11.4 and 11.5 provide a detailed view of nDCG@10 across varying feedback depths ($k \in \{1,3,5,10,20\}$) and ten distinct feature types on TREC DL19 and DL20. The analysis focuses on three representative configurations, one from each model family, QWEN2.5, QWEN3, and GEMMA3, using the specific retriever and extractor combinations that provided the most robust performance in previous sections (Table 11.4 and 11.5).

PRF Depth

Across most configurations, shallow to moderate depths ($k \in [3,10]$) tend to offer the most consistent improvements for retrieval effectiveness. These depths provide an optimal balance between refining the query representation with relevant context and mitigating the risks of introducing query drift from lower-ranked passages. For instance, in Figure 11.4 (left), the QWEN2.5 14B retriever on TREC DL19 demonstrates a clear upward trajectory, where effectiveness improves consistently as depth increases from 1 to 10, eventually plateauing at depth 20. This suggests that for mid-sized models, aggregating evidence from multiple passages is crucial for robust query refinement.

However, a different pattern emerges for varying model capacities and datasets, particularly on TREC DL20 (Figure 11.5). While the QWEN3 8B retriever (center) maintains stability up to depth 10, the GEMMA3 27B retriever (right) exhibits immediate saturation. In this case, the peak performance for most feature types is observed at depth 1, followed by a gradual decline as depth increases to 20.

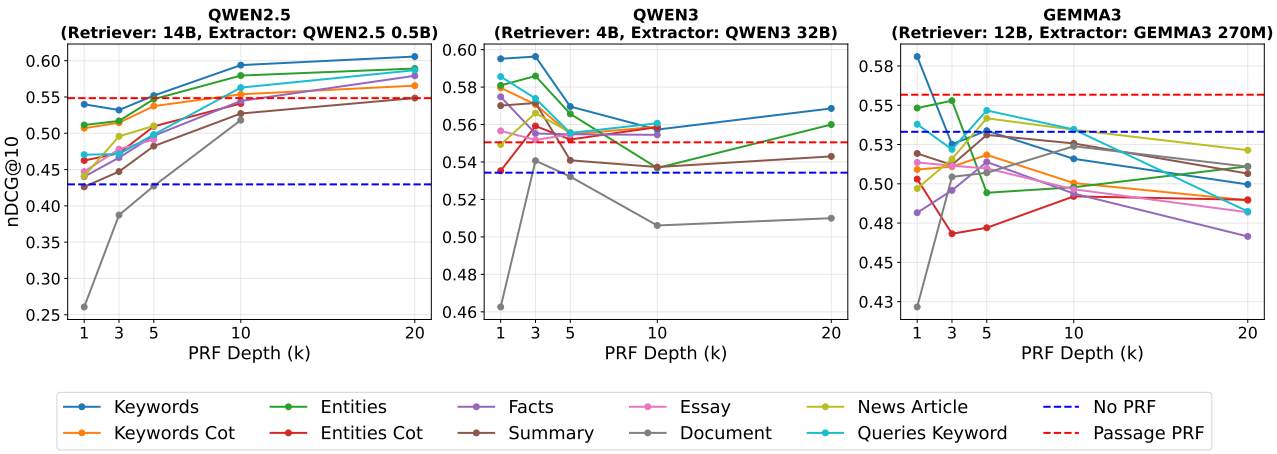


Figure 11.4: Impact of feature type and PRF depth on nDCG@10 for TREC DL19. Each subplot shows a different model family with its optimal retriever and extractor configuration: QWEN2.5 (14B retriever, 0.5B extractor), QWEN3 (4B retriever, 32B extractor), and GEMMA3 (12B retriever, 270M extractor). Blue and red dashed lines indicate No PRF and Passage PRF baselines, respectively.

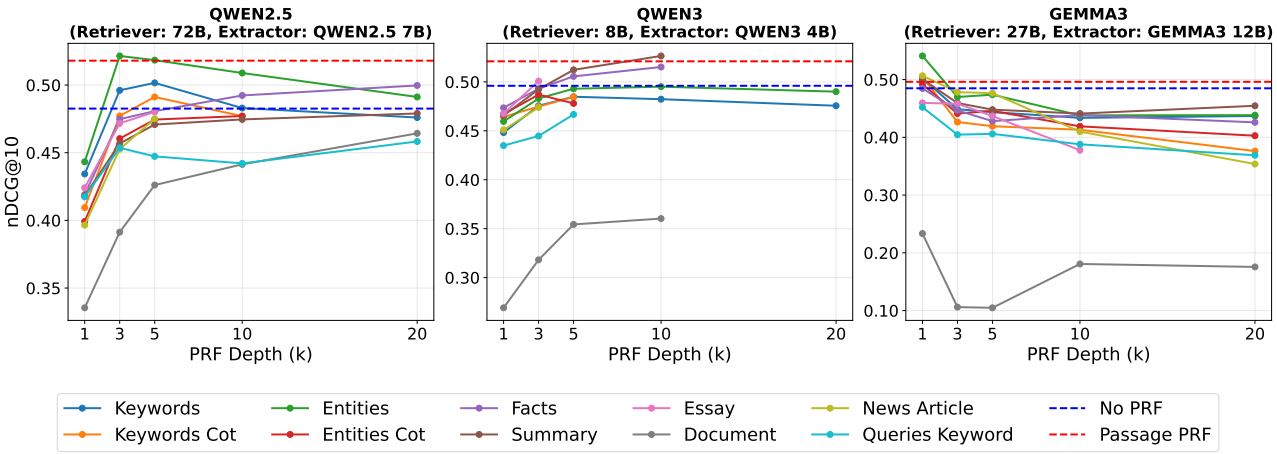


Figure 11.5: Impact of feature type and PRF depth on nDCG@10 for TREC DL20. Each subplot shows a different model family with its optimal retriever and extractor configuration: QWEN2.5 (72B retriever, 7B extractor), QWEN3 (8B retriever, 4B extractor), and GEMMA3 (27B retriever, 12B extractor). Blue and red dashed lines indicate No PRF and Passage PRF baselines, respectively.

This indicates that for high-capacity models on certain tasks, the initial top-ranked signal is highly precise, and adding further context from lower-ranked documents introduces noise that dilutes the query intent.

Furthermore, the impact of depth is heavily dependent on the feature type. As seen in both figures, the `Document` feature (grey line) consistently starts with very low performance at shallow depths and requires deeper pooling ($k = 10$ or 20) to become competitive, though it rarely surpasses the structured features. In contrast, structured features like `Entities` and `Keywords` remain robust even at deeper settings. These findings suggest that while moderate depth is generally safer, the optimal setting is a function of the retriever's base capability: weaker retrievers benefit from deeper feedback accumulation, whereas stronger retrievers require shallower, high-precision signals.

Feature Type

Features that are compact, factual, and semantically focused, such as `Keywords`, `Entities`, and `Facts`, consistently outperform other feature types across the majority of configurations. These features provide structured, topic-relevant signals that effectively guide the dense retriever without introducing the noise associated with longer text segments. As illustrated in Figure 11.4, on TREC DL19, the `Keywords` (blue line) and `Entities` (green line) features consistently provide the highest nDCG@10 scores for the QWEN2.5 and GEMMA3 families. Their performance remains robust across increasing PRF depths, suggesting that these concise semantic units are less likely to induce query drift even when aggregated from multiple passages.

In contrast, verbose or unstructured features such as `Document`, `Essay`, and `News-Article` often lead to suboptimal or unstable results. The `Document` feature (grey line), which acts as a proxy for standard passage expansion, exhibits a particularly distinct failure mode at shallow depths. In almost all subplots in Figures 11.4 and 11.5, the `Document` feature starts with performance significantly below the No PRF baseline at $k = 1$, only recovering competitiveness to No PRF baseline at deeper pools ($k = 20$) on TREC DL 2019, it always worse than the No PRF baseline on TREC DL 2020 even with larger k. This confirms that injecting raw, verbose passage content into the prompt often dilutes the query intent, whereas distilled features provide immediate value.

Furthermore, on more challenging benchmarks like TREC DL20 (Figure 11.5), the stability gap widens. While structured features maintain their effectiveness for the QWEN2.5 and QWEN3 models, verbose features tend to plateau early. Notably, in the GEMMA3 configuration (right), we observe a general degradation at depths beyond 5, but structured features like `Entities` decay much more gracefully than loosely focused types. These observations underscore the importance of factual, semantically rich feedback in PromptPRF and highlight the risks of using verbose, unstructured generation, particularly when the retriever is sensitive to noise.

Retriever Model Size

Consistent with earlier findings, smaller and medium-sized retrievers benefit most substantially from PromptPRF. In Figure 11.4 (left), the QWEN2.5 14B retriever improves dramaticallly over its No PRF baseline when equipped with structured feature types. The nDCG@10 score leaps from approximately 0.43 (No PRF) to over 0.60 when using `Keywords` at depth 20. This indicates that PromptPRF acts as a powerful boosting mechanism for models with moderate capacity, injecting the external semantic context necessary to bridge the gap between them and much larger architectures.

In contrast, the largest retrievers, such as the QWEN2.5 72B shown in Figure 11.5 (left) and GEMMA3 27B (right), demonstrate more modest relative improvements. Although these models already possess strong internal knowledge and perform well without feedback (starting baselines around 0.48), PromptPRF still offers additional gains, pushing performance to the 0.52–0.54 range with structured features like `Entities`. However, the margin of improvement is significantly narrower compared to the gains seen in the QWEN3 4B or QWEN2.5 14B models. This suggests that while

PromptPRF is universally beneficial, its impact is most transformative when the base retriever has room for semantic enhancement, rather than when it is already approaching the ceiling of zero-shot performance.

Importantly, these results highlight that a smaller retriever with PromptPRF can often outperform a significantly larger retriever without it. For instance, the QWEN2.5 14B with PromptPRF on TREC DL19 (0.6057) surpasses the QWEN2.5 72B baseline (0.5931). This confirms that PromptPRF offers a viable alternative to brute-force scaling: instead of deploying the largest available model, practitioners can achieve state-of-the-art results by pairing a more efficient, mid-sized retriever with strategic offline feature generation.

Summary

These results address **RQ3.2.3** and demonstrate that the effectiveness of PromptPRF is contingent upon the critical interplay between feature type and feedback depth. Structured and semantically focused features consistently offer the most robust performance, particularly when utilized at shallow to moderate PRF depths (e.g., $k = 3$ to 10). In contrast, verbose features (such as `Document`) often introduce noise, and extending feedback to deeper pools (e.g., $k = 20$) frequently leads to diminishing returns or performance degradation, particularly for high-capacity models like GEMMA3 that appear more sensitive to query drift.

Regarding feature selection, smaller models derive disproportionate benefit from semantically rich, structured feedback, which effectively augments their limited internal capacity. Larger retrievers also benefit, though they require high-precision signals to enhance their existing representations without introducing redundancy.

Overall, these findings validate PromptPRF as a flexible framework. Its success is driven not merely by model scale, but by the strategic selection of feature types and depths. With minimal hyperparameter tuning, smaller retrievers can achieve retrieval effectiveness comparable to or exceeding that of much larger architectures, all while maintaining the stability and efficiency required for practical deployment.

11.4.4 Generalisation to BEIR

To evaluate the generalizability of PromptPRF beyond the TREC DL datasets and answer **RQ3.2.4: Does PromptPRF generalize to out-of-domain datasets?**, we conduct experiments on the BEIR benchmark. From the original 19 datasets, we select 8 publicly available datasets that are most commonly used in dense retrieval research: ArguAna, FiQA, NFCorpus, Quora, SciDocs, SciFact, TREC-COVID, and Touche-2020. We evaluate the QWEN2.5 model family, ranging from the lightweight 0.5B to the large-scale 72B parameter variants, comparing three configurations for each: (i) Baseline (No PRF), (ii) Passage PRF (direct concatenation of top-k passages), and (iii) our PromptPRF (We use the best extractor/feature/depth combination from TREC DL 2019 results, as shown in Table 11.4).

Figures 11.6, 11.7, and 11.8 present nDCG@10 scores across all 8 BEIR datasets for small (0.5B–3B), mid-sized (7B–14B), and large (32B–72B) retrievers, respectively. The results indicate that while

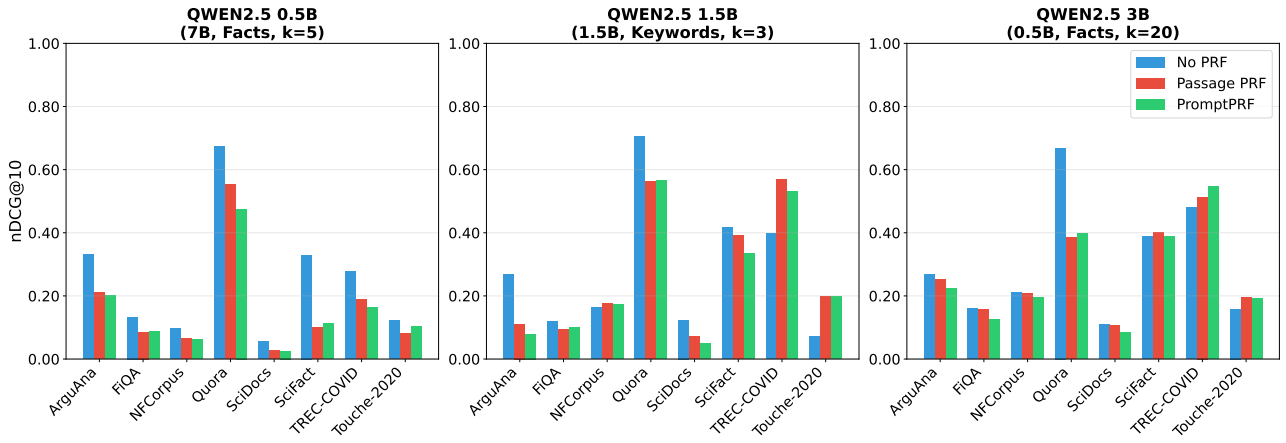


Figure 11.6: BEIR evaluation with small-size QWEN2.5 retrievers (0.5B–3B). Each retriever uses the best PromptPRF configuration from TREC DL 2019.

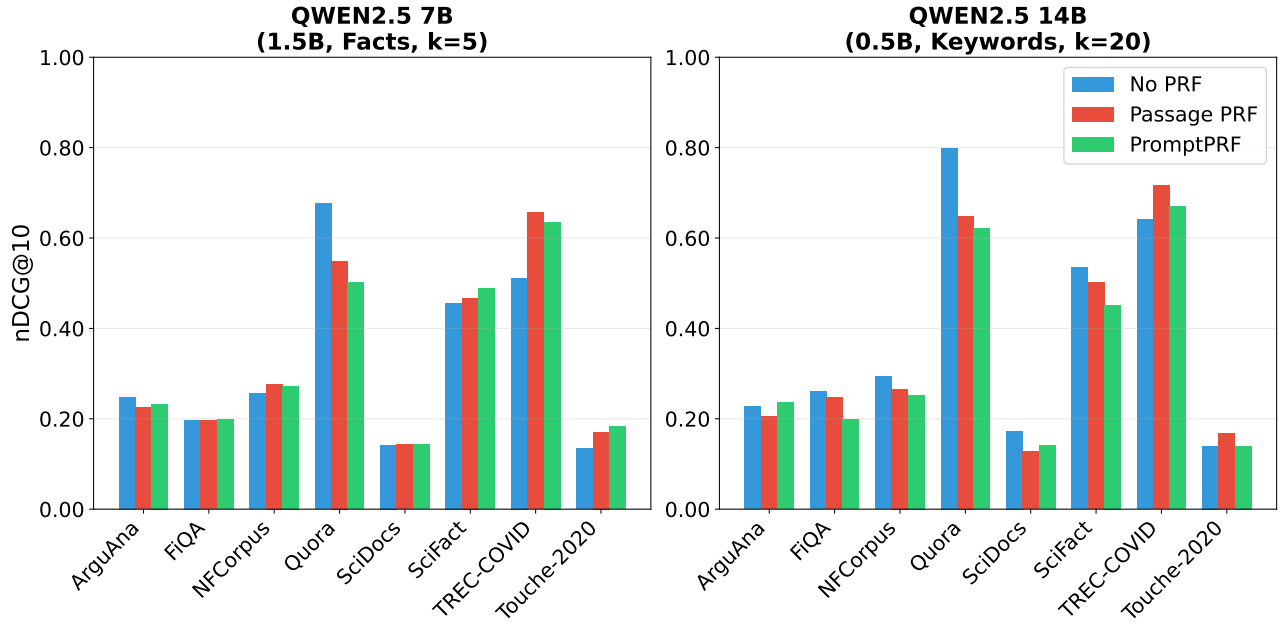


Figure 11.7: BEIR evaluation with mid-size QWEN2.5 retrievers (7B–14B). Each retriever uses the best PromptPRF configuration from TREC DL 2019.

the base QWEN2.5 models are strong zero-shot retrievers, PromptPRF provides notable improvements on specific challenging domains. The most consistent gains are observed on datasets requiring complex reasoning or handling shifting data distributions, such as **TREC-COVID** and **Touche-2020**. For instance, the QWEN2.5 3B retriever (Figure 11.6, right) sees its performance on TREC-COVID rise from approximately 0.48 (No PRF) to 0.55 (PromptPRF). Similarly, mid-sized models like the 7B variant (Figure 11.7, left) achieve superior performance on Touche-2020 with PromptPRF compared to both No PRF and Passage PRF.

However, the generalization is not uniform across all tasks. On the **Quora** dataset, the zero-shot baseline (No PRF) consistently outperforms PRF methods across most model sizes. This suggests that for duplicate question detection, the base embeddings are already highly optimized, and adding external feedback features may introduce semantic drift. Nevertheless, for the majority of datasets,

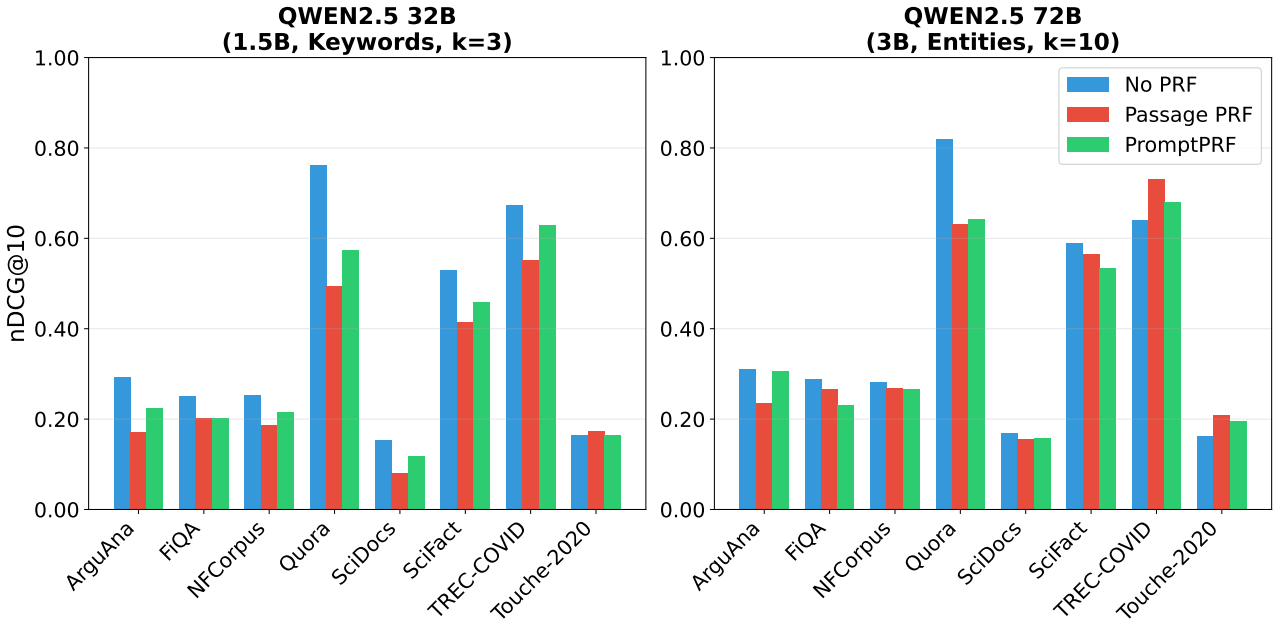


Figure 11.8: BEIR evaluation with large-size QWEN2.5 retrievers (32B–72B). Each retriever uses the best PromptPRF configuration from TREC DL 2019.

PromptPRF remains competitive or superior, particularly for smaller models where the semantic enrichment helps bridge the capacity gap.

This trend is comprehensively illustrated in Figure 11.9, which provides a scatter plot comparison across all QWEN2.5 retrievers and datasets. The right subplot (Passage PRF vs. PromptPRF) shows that the majority of points lie above the diagonal, confirming that PromptPRF's structured features generally yield better retrieval effectiveness than raw Passage PRF. The left subplot (No PRF vs. PromptPRF) shows a clustering around the diagonal, with distinct subsets of points, which are corresponding to tasks like TREC-COVID and Touche-2020, situated above the line, highlighting robust gains.

These results offer three key observations: (i) PromptPRF provides consistent improvements over Passage PRF, validating the utility of structured offline features over raw text injection; (ii) PromptPRF is particularly effective for domains with complex information needs (e.g., Argument retrieval in Touche, Biomedical in TREC-COVID), effectively compensating for the limited domain knowledge of smaller retrievers; and (iii) while some tasks like Quora favor pure zero-shot retrieval, PromptPRF demonstrates strong robustness across the broader benchmark without requiring task-specific fine-tuning.

Taken together with earlier findings from TREC DL, these BEIR results reaffirm that PromptPRF is adaptable to heterogeneous, zero-shot, out-of-domain settings. The method allows resource-constrained deployments, with models as small as QWEN2.5 0.5B or 3B, to achieve competitive performance on complex tasks where larger models might typically be required.

In summary, the BEIR evaluation answers **RQ3.2.4** and validates PromptPRF as a generalizable feedback strategy. By decoupling feature generation from query-time inference, it offers a practical, scalable solution for improving dense retrieval effectiveness across diverse domains.

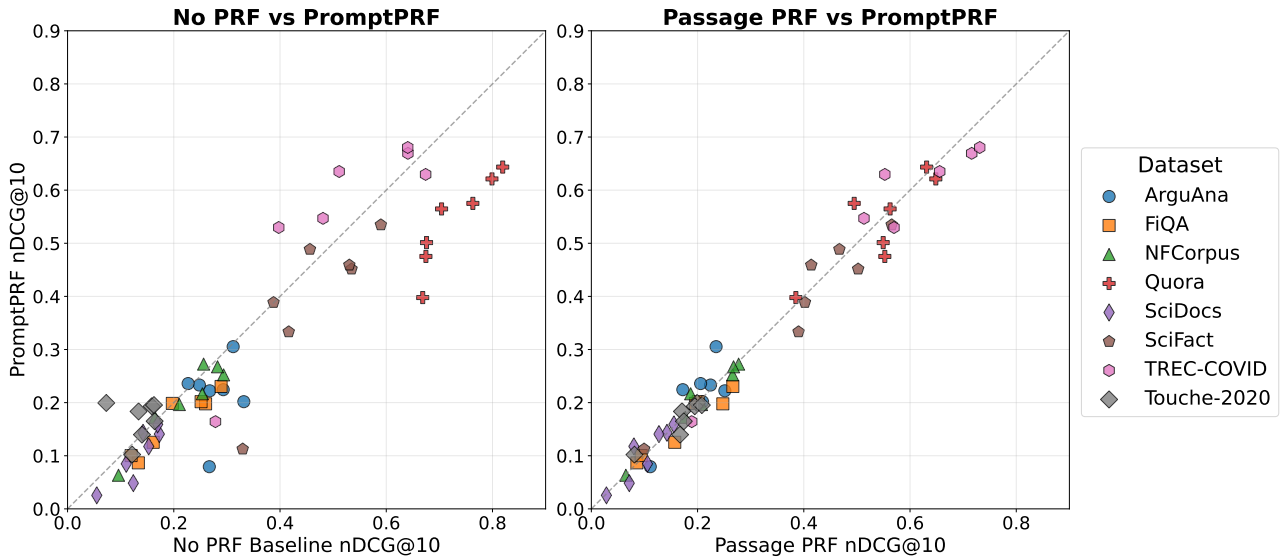


Figure 11.9: Scatter plot comparison of PromptPRF against baseline methods across eight BEIR datasets using QWEN2.5 retrievers (0.5B–72B parameters). Left: No PRF baseline vs. PromptPRF. Right: Passage PRF vs. PromptPRF. Each point represents a retriever-dataset combination, with the diagonal line indicating equal performance. Points above the diagonal indicate PromptPRF improvement.

11.4.5 Impact of Rank Information

PromptPRF constructs enhanced queries by conditioning on features extracted from top-ranked passages. During this process, a key design question arises: whether to include the rank position of each feedback unit (e.g., "Top 1 Retrieved Passage") or to treat all feedback uniformly. Therefore, in this section, we aim to investigate **RQ3.2.5: Does incorporating rank information into the feedback prompt contribute to more effective query refinement?** Unlike traditional PRF methods like RM3, which implicitly prioritize higher-ranked passages, most recent generative feedback models, including GRF [139] and GenPRF [217], ignore rank and assume equal contribution from all feedback signals. PromptPRF explicitly encodes rank as positional metadata in the prompt, enabling the LLM to condition its generation on retrieval order. This design choice aims to leverage retrieval-time confidence as a signal for weighting feedback relevance.

To evaluate this hypothesis, we conduct a comprehensive ablation study across the QWEN2.5 model family, utilizing the best-performing configurations identified in Tables 11.4 and 11.5. We analyze the impact of rank information across five dimensions:

- Performance on optimal configurations (Figure 11.10);
- Sensitivity to PRF depth in PromptPRF (Figures 11.11 and 11.12);
- Impact of Extractor scale (Figure 11.13); and
- Impact of Feature Types (Figure 11.14).

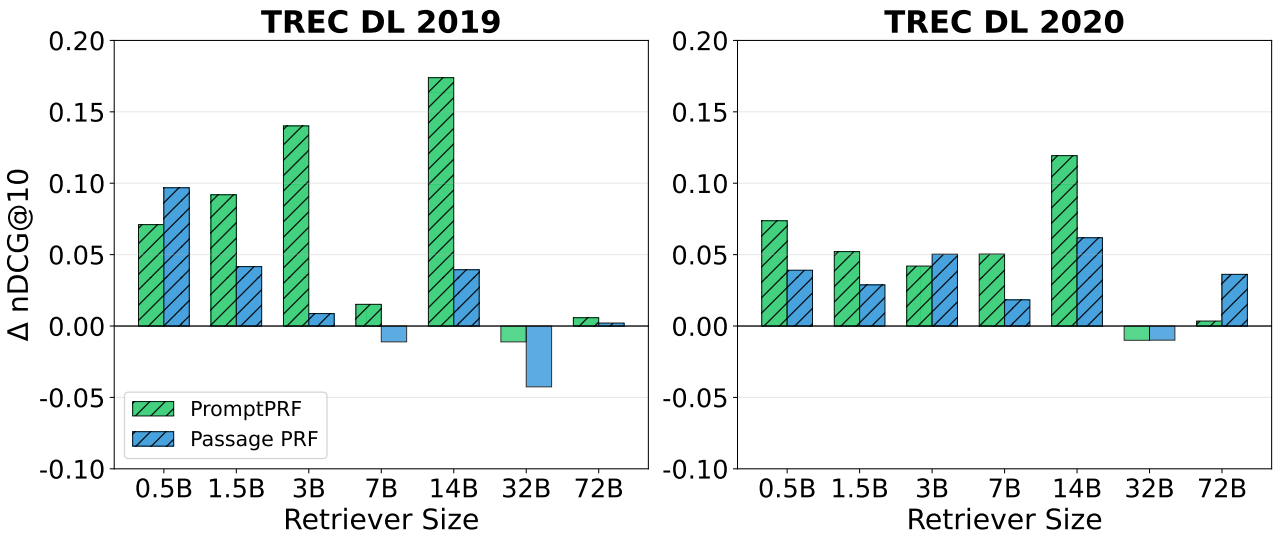


Figure 11.10: Impact of incorporating rank information in the feedback prompt on retrieval effectiveness. Each bar shows the performance difference (Δ nDCG@10) between prompts that include rank positions and prompts that omit this information. Positive values indicate that rank information improves performance. Results are shown for both PromptPRF (using LLM-generated features) and Passage PRF (using raw passages) across all QWEN2.5 retriever sizes. We use the best configurations from Tables 11.4 and 11.5.

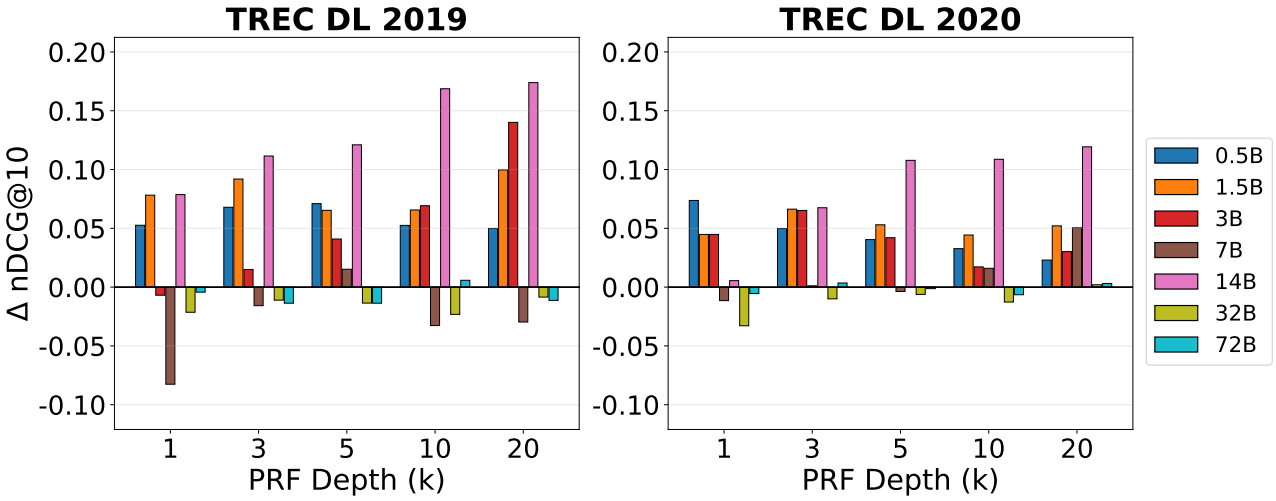


Figure 11.11: Impact of rank information on PromptPRF across different PRF depths. Each bar represents the performance difference (Δ nDCG@10) between prompts with rank information and prompts without rank information for each retriever model. Positive values indicate that including passage rank positions in the feedback prompt improves retrieval effectiveness. Results are shown using each retriever's best extractor and feature type configuration (Tables 11.4 and 11.5).

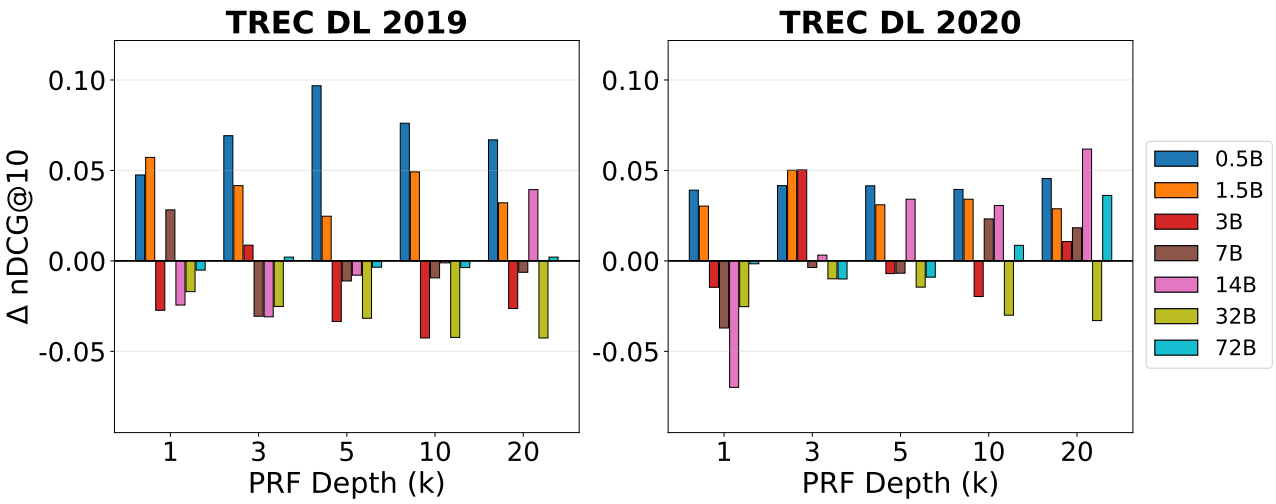


Figure 11.12: Impact of rank information on Passage PRF across different PRF depths. Each bar shows the performance difference (Δ nDCG@10) between prompt with and without rank information. Unlike PromptPRF, Passage PRF directly uses passages without LLM-generated features, isolating the effect of rank information in the prompt structure.

Each configuration is tested under two conditions: with rank information in the prompt (Figure 11.1) and without rank information (Figure 11.15). This systematic evaluation allows us to isolate the specific contribution of positional metadata to retrieval effectiveness.

Overall Impact of Rank Information

Figure 11.10 presents the overall impact of incorporating rank information across all QWEN2.5 retriever sizes. The y-axis represents the performance difference (Δ nDCG@10) between prompts with and without rank information.

The results reveal a strong positive impact, with PromptPRF (green bars) showing consistent improvements across most model sizes. The effect is particularly observable for the QWEN2.5 14B retriever, which sees a massive performance boost of over +0.17 on TREC DL19 and +0.12 on TREC DL20 when rank metadata is included. This suggests that mid-sized models, which have significant capacity but may lack the perfect noise-filtering of larger models, benefit most from explicit instructional cues regarding signal quality.

Notably, PromptPRF benefits more consistently from rank information than Passage PRF (blue bars). While Passage PRF shows gains in some cases (e.g., 0.5B on DL19), it yields negative or negligible impacts in others (e.g., 7B and 32B on DL19). This indicates that semantically distilled features (PromptPRF) are easier for the LLM to weight based on rank than raw, verbose passages, which may contain internal conflicting signals that dilute the rank metadata's effectiveness.

Interaction with PRF Depth

Figures 11.11 (PromptPRF) and 11.12 (Passage PRF) break down the impact of rank information across varying feedback depths (k).

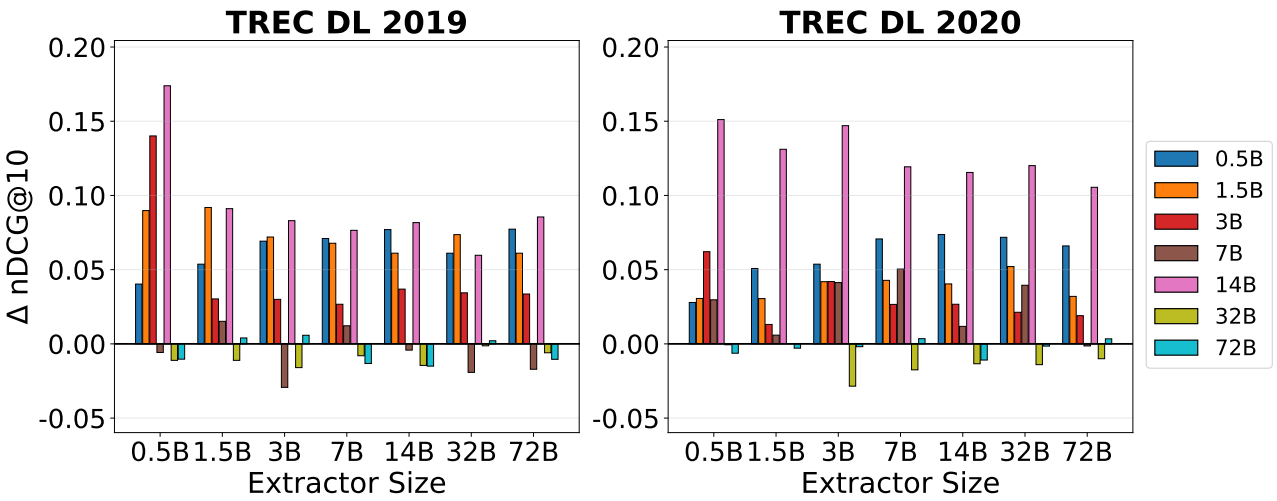


Figure 11.13: Impact of rank information on PromptPRF across different extractor model sizes. For each retriever, the delta is computed using the best feature type and PRF depth while varying the extractor size from 0.5B to 72B parameters (Tables 11.4 and 11.5) with prompts w/wo rank information.

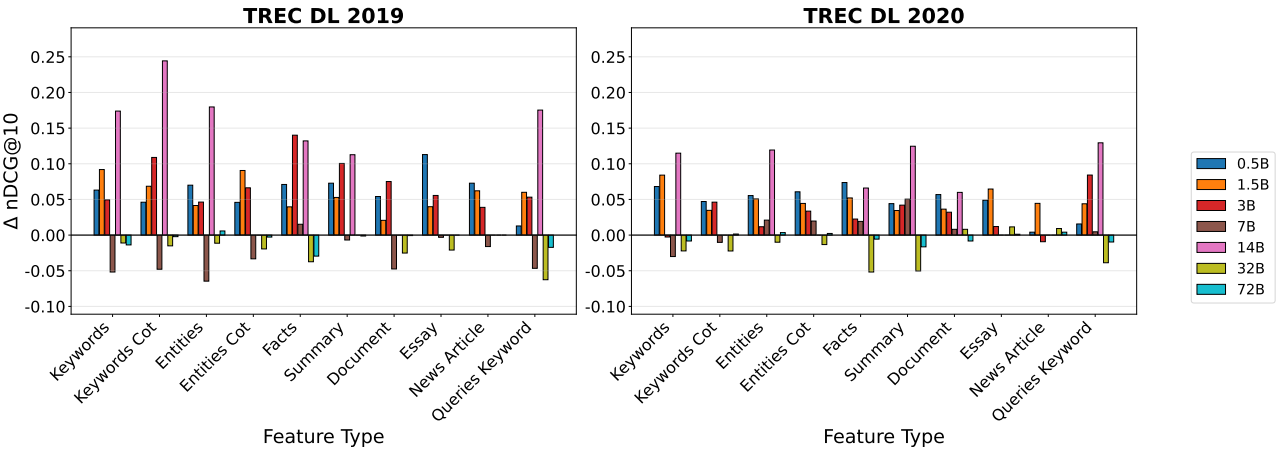


Figure 11.14: Impact of rank information on PromptPRF across different feature types. For each retriever model, the delta is computed using the best extractor and PRF depth configuration while varying the feature type (Tables 11.4 and 11.5) with prompts w/wo rank information.

For PromptPRF (Figure 11.11), rank-aware prompting proves critical for stabilizing performance at deeper depths. On TREC DL19, the QWEN2.5 14B retriever (pink bars) exhibits a dramatic increase in Δ nDCG@10 as depth increases to 10 and 20. Without rank information, aggregating 20 feedback items often introduces excessive noise, leading to query drift. However, with rank information, the retriever effectively "discounts" lower-ranked features while preserving their semantic utility, allowing it to leverage deeper pools of information without degradation.

In contrast, Passage PRF (Figure 11.12) shows a much more volatile relationship with depth. On TREC DL20, while the 14B model benefits at depth 20, the 1.5B and 32B models show significant performance regressions (negative bars) at various depths even when rank info is provided. This reinforces the finding that raw passage content is inherently noisier; simply adding "Rank 1" or "Rank 5" labels is often insufficient to prevent the LLM from being overwhelmed by verbose, off-topic text in the prompt.

```
[
  {"role": "system", "content":
    "You are an AI assistant that can understand human language."},
  {"role": "user", "content":
    'Query: "{QUERY}".\n
     {FEATURE NAME} for Retrieved Passage: "{PASSAGE OR FEATURE CONTENT}".
     ...
     {FEATURE NAME} for Retrieved Passage: "{PASSAGE OR FEATURE CONTENT}".\n
     Use one word to represent the query and the top passages in a retrieval task.
     Make sure your word is in lowercase."
  },
  {"role": "assistant", "content": 'The word is: "'}
]
```

Figure 11.15: Prompt format without rank information.

Impact across Feature Types

Figure 11.14 illustrates how different feedback features respond to rank metadata. Structured and concise features, specifically `Keywords-COT`, `Entities`, and `Facts` features, demonstrate the most robust improvements.

On TREC DL19, the `Entities` and `Keywords-COT` features show strong positive gains across almost all retriever sizes. The reasoning is that these features provide discrete semantic units (e.g., specific entities or key phrases). Rank metadata allows the model to form a hierarchy of importance: entities from "Rank 1" are treated as core query concepts, while entities from "Rank 10" are treated as peripheral context.

On the other hand, verbose features like `Document` and `News Article` show mixed or smaller gains. For example, on TREC DL20, the `Document` feature yields negligible improvement for several models. This suggests that when the feedback unit itself is long and complex, the positional signal ("Rank X") becomes less effective at guiding the attention mechanism compared to when the feedback is short and structured.

Robustness to Extractor Size

Figure 11.13 analyzes whether the benefit of rank information depends on the model size of the offline feature extractor. The results show a remarkable consistency: for a given retriever (e.g., QWEN2.5 14B, pink bars), the benefit of rank information remains high regardless of whether the features were generated by a 0.5B extractor or a 72B extractor.

This is a crucial finding for system design. It confirms that the effectiveness of rank-aware prompting is a function of the *retriever's* ability to interpret the prompt, not a property of the generated features themselves. It validates the strategy of using lightweight, efficient extractors (e.g., 1.5B) in conjunction with rank-aware prompting to achieve maximal performance.

Summary and Implications

Across the evaluated QWEN2.5 configurations, incorporating rank information yields improvements, particularly when utilizing structured features and scaling to deeper feedback pools (e.g., $k = 10$ or 20).

The empirical results indicate that rank metadata is essential for maintaining retrieval effectiveness as the volume of feedback increases. While rank-agnostic prompts often suffer from diminishing returns or degradation at higher depths, the rank-aware variants consistently maintain or improve performance. This stabilization is distinctively observable in mid-sized retrievers (e.g., QWEN2.5 14B), where rank metadata proves critical for preventing the effectiveness drops otherwise observed when aggregating evidence from a larger set of passages.

Importantly, these benefits are achieved without requiring modifications to the model architecture or additional training. The rank signal is inherently available from the initial retrieval step and can be leveraged at no additional runtime cost. Therefore, rank-aware prompting represents a practical and cost-effective enhancement for LLM-based feedback in zero-shot dense retrieval.

These findings provide a clear answer to **RQ3.2.5**: incorporating rank information into the prompt consistently enhances query representation in PromptPRF. It serves as a vital control mechanism that enables more accurate and robust retrieval across diverse retriever sizes and feature types, specifically unlocking the potential of deeper feedback integration.

11.4.6 Costs and Trade-offs

To evaluate if PromptPRF is practical in real-world retrieval applications, we analyze its computational requirements, online latency, and the trade-offs between effectiveness and efficiency. This analysis addresses **RQ3.2.6: What are the computational costs and efficiency trade-offs of PromptPRF compared to baseline methods?**

Hardware Requirements

We detail the minimum computational resources required to deploy the different LLM backbones in Table 11.1. The data illustrates a stark contrast in accessibility between model scales. Smaller models, such as the QWEN2.5 3B or GEMMA3 4B, are highly resource-efficient; the QWEN2.5 3B retriever requires only 8 GB of VRAM and can be deployed on widely available consumer-grade hardware (e.g., NVIDIA RTX 3060). Even the ultra-compact QWEN2.5 0.5B requires as little as 2 GB of VRAM.

In contrast, large-scale models impose prohibitive infrastructure demands. The QWEN2.5 72B and GEMMA3 27B models require approximately 165 GB and 60 GB of VRAM, respectively. Deploying the QWEN2.5 72B retriever typically requires a cluster of enterprise-grade GPUs (e.g., $3\times$ NVIDIA H100 or A100) and substantial system RAM ($>$140 GB). These high infrastructure requirements present significant barriers to deployment in edge computing, mobile environments, or cost-sensitive production systems, whereas the smaller backbones enabled by PromptPRF fit comfortably within commodity hardware constraints.

Query Encoding Latency

Since PromptPRF performs feature extraction offline, the primary runtime cost stems from the query encoding step during retrieval. Table 11.2 reports the latency measured on NVIDIA H100 GPUs. The

Table 11.6: Comparison of PromptPRF with GRF [139] and Query2Doc [209] on TREC DL 2019. Results for GRF and Query2Doc are directly copied from the original papers without independent reproduction. Results for PromptPRF are from Table 11.4. The best result is marked with **Bold**.

TREC DL 2019 (43 queries)				
Model	Feature Extractor	Feature	PRF Depth	nDCG@10
BM25+GRF[139]	GPT-3	All	1	0.6200
BM25+GRF[139]	GPT-3	Entities-COT	1	0.5630
BM25+GRF[139]	GPT-3	Essay	1	0.6090
BM25+Query2doc[209]	text-davinci-003	Document	1	0.6620
DPR+Query2doc[209]	text-davinci-003	Document	1	0.6870
E5+KD+Query2doc[209]	text-davinci-003	Document	1	**0.7490**
PromptPRF(QWEN2.5 14B)	QWEN2.5 0.5B	Keywords	20	0.6057
PromptPRF(QWEN3 4B)	QWEN3 32B	Keywords	3	0.5963
PromptPRF(GEMMA3 12B)	GEMMA3 270M	Keywords	1	0.5810

results highlight the massive speed advantage of smaller retrievers. The QWEN2.5 0.5B model encodes queries at an average of 11.76 ms/query. Even the more capable QWEN2.5 7B variant operates at 29.81 ms/query.

Conversely, the large-scale QWEN2.5 72B model incurs a latency of 281.29 ms/query, representing an increase of over 2,292% compared to the 0.5B baseline and approximately 9.4 times slower than the 7B variant. Similarly, the GEMMA3 27B model requires 165.63 ms/query, much slower than its lighter counterparts. This disparity demonstrates that using a smaller retriever with PromptPRF is not only more hardware-efficient but also substantially faster, making it far more practical for real-time applications where low latency is critical.

Comparison with Other Generative Feedback Methods

Tables 11.6 and 11.7 compare PromptPRF against recent generative feedback approaches like GRF [139] and Query2Doc [209] on TREC DL19 and DL20. Although PromptPRF does not strictly match the state-of-the-art nDCG@10 scores of methods utilizing massive commercial models (like text-davinci-003), it offers a radically superior efficiency-effectiveness trade-off.

For example, on TREC DL19, PromptPRF using the QWEN2.5 14B retriever achieves an nDCG@10 of 0.6057. This score is highly competitive, effectively matching BM25+GRF Essay (0.6090) and outperforming BM25+GRF Entities-COT (0.5630). Crucially, GRF requires online generation of up to 2,944 tokens ($64 \times 2 + 256 \times 5 + 512 \times 3$) per query. Assuming a standard generation speed of roughly 73 ms/token for large API-based models, this translates to an estimated latency of:

$$\textbf{Total latency} = 2,944 \times 73 \text{ ms} \approx \textbf{214,900 milliseconds}$$

Adding the retrieval time, the total latency exceeds 3.5 minutes per query. Besides the prohibitive latency, this incurs a per-query cost of approximately 0.044 USD via GPT-3 class APIs. Similarly, Query2Doc on DL20 achieves a high score of 0.7250 but shares the same bottleneck of expensive,

Table 11.7: Comparison of PromptPRF with GRF [139] and Query2Doc [209] on TREC DL 2020. Results for GRF and Query2Doc are directly copied from the original papers without independent reproduction. Results for PromptPRF are from Table 11.5. The best result is marked with **Bold**.

TREC DL 2020 (54 queries)				
Model	Feature Extractor	Feature	PRF Depth	nDCG@10
BM25+GRF[139]	GPT-3	All	1	0.6070
BM25+GRF[139]	GPT-3	Facts	1	0.5830
BM25+GRF[139]	GPT-3	Keywords-COT	1	0.5420
BM25+Query2doc[209]	text-davinci-003	Document	1	0.6290
DPR+Query2doc[209]	text-davinci-003	Document	1	0.6710
E5+KD+Query2doc[209]	text-davinci-003	Document	1	**0.7250**
PromptPRF(QWEN2.5 72B)	QWEN2.5 7B	Entities	3	0.5215
PromptPRF(QWEN3 8B)	QWEN3 4B	Summary	10	0.5265
PromptPRF(GEMMA3 27B)	GEMMA3 12B	Entities	1	0.5409

synchronous query expansion. Therefore, these on-the-fly generation methods are impractical for high-throughput or low-latency scenarios.

In contrast, PromptPRF decouples feedback generation from the online retrieval process. Features are computed offline and reused. The major online cost consists only of two passes of query encoding. Using the QWEN2.5 14B model (Table 11.2), the latency is 41.57 ms/query. Thus, the full retrieval latency is approximately:

$$\textbf{PromptPRF latency} \approx 41.57 \text{ ms} \times 2 \approx \textbf{83 milliseconds}$$

This makes the QWEN2.5 14B configuration roughly 2,500 times faster than the GRF baseline while delivering comparable retrieval quality (0.6057 vs 0.6200). Furthermore, smaller configurations like QWEN3 4B (0.5963 nDCG) operate at just $\sim$55 ms per query, offering even faster speed.

Efficiency Summary

PromptPRF trades marginal differences in absolute retrieval performance for transformative gains in efficiency:

- **Latency**: Significantly faster than GRF (214,900ms) or Query2Doc (BM25+Query2doc: > 2,177ms [209]), enabling sub-100ms response times;
- **Cost**: Zero per-query generation cost, as all features are pre-computed offline;
- **Hardware**: Can be deployed on single consumer-grade GPUs (e.g., QWEN2.5 3B on an RTX 3060), avoiding the complexity of the multi-GPU or API-dependent setups required for models like GPT-3 or LLaMA3.3 70B.

These advantages make PromptPRF particularly attractive for practical deployment, where hardware availability, latency budgets, and operational costs are primary concerns. While large online

generative models remain the ceiling for theoretical retrieval effectiveness, PromptPRF enables compact dense retrievers to achieve competitive performance while remaining lightweight, scalable, and cost-effective.

11.5 Summary

This chapter investigated how Pseudo-Relevance Feedback can enhance the effectiveness and efficiency of zero-shot LLM-based dense retrieval, addressing the high-level research question **RQ3.2: Can we make LLM-based Pseudo-Relevance Feedback more effective and efficient?** To this end, we introduced **PromptPRF**, a generative PRF method that leverages LLM-generated features extracted from top-ranked passages and integrates them into query representations via prompting. Importantly, PromptPRF operates without incurring significant query-time costs, hence avoiding the latency, compute, and hardware burdens typically associated with model scaling. Therefore, PromptPRF offers a practical alternative for improving retrieval in resource-constrained or latency-sensitive settings.

To address **RQ3.2.1**, our results demonstrate that PromptPRF consistently outperforms both No PRF and Passage PRF baselines across TREC DL and BEIR benchmarks. The most substantial gains are observed for smaller retrievers, such as the QWEN2.5 3B and GEMMA3 4B, which benefit significantly from high-quality feedback features. In many cases, these compact models achieve performance comparable to or exceeding significantly larger models (e.g., QWEN2.5 14B or 32B) without PRF. This finding challenges conventional scaling laws [71, 82] by showing that retrieval effectiveness can be substantially improved through PRF rather than brute-force increasing the model size.

To address **RQ3.2.2**, we examined the role of the feature extractor. While larger extractors (e.g., QWEN2.5 72B) can enhance performance, the improvements saturate quickly. Our analysis revealed that compact to mid-sized extractors (e.g., QWEN2.5 1.5B or GEMMA3 270M) frequently provide feedback features of equal utility to those generated by massive models. This suggests that feature quality for retrieval tasks depends more on the semantic clarity of the output than on the raw parameter count of the generator, highlighting the viability of using lightweight models for efficient offline indexing.

To address **RQ3.2.3**, we explored how retrieval effectiveness varies by different PRF depths and feature types. Structured, semantically concise features—especially `Entities`, `Keywords-COT`, and `Facts`—consistently outperform verbose or less-focused alternatives like `Document` or `Essay`. Moreover, moderate PRF depths ($k = 3$ to 10) typically offer optimal performance, whereas deeper depths (e.g., $k = 20$) often introduce diminishing returns or query drift, particularly for higher-capacity retrievers. These results highlight the importance of selecting suitable feedback features and depths to balance signal enrichment with noise control.

To address **RQ3.2.4**, we evaluated PromptPRF across 8 diverse BEIR datasets to assess generalization. PromptPRF delivered consistent improvements over baselines for the QWEN2.5 family ranging from 0.5B to 72B parameters. We can observe gains for smaller models on complex tasks

like TREC-COVID and Touche-2020, effectively bridging the performance gap with larger baselines. This confirms that PromptPRF can be generalized to out-of-domain tasks, reaffirming its utility as a scalable and retriever-agnostic method for zero-shot dense retrieval.

To address **RQ3.2.5**, we conducted an extensive ablation study on rank-aware prompting. Incorporating retrieval rank metadata into the prompt consistently improves effectiveness, particularly for structured features and when scaling to deeper feedback pools. The rank signal helps the retriever prioritize higher-confidence signals and suppress noise from lower-ranked passages, serving as a lightweight but impactful control mechanism for guiding query refinement.

Finally, to address **RQ3.2.6**, we analyzed the trade-offs between effectiveness, efficiency, and cost. While PromptPRF does not strictly match the absolute ceiling of expensive online methods like GRF [139] or Query2Doc [209], it offers advantages in runtime and deployability. By offloading feature generation to the offline phase, PromptPRF enables real-time query inference with negligible latency overhead (sub-50ms for small models). It supports fast, low-cost retrieval using compact retrievers like QWEN2.5 3B on consumer-grade hardware, making it a highly favorable option for production-scale and resource-limited applications compared to API-dependent or multi-GPU solutions.

PromptPRF offers a flexible, training-free, and cost-efficient framework for PRF in LLM-based dense retrieval. It enhances both small and large retrievers, generalizes well to heterogeneous tasks, and can be deployed under strict latency constraints. The effectiveness of PromptPRF relies on the strategic selection of feedback types (concise, structured), PRF depth (moderate), and the efficient pairing of retrievers with compact feature extractors. Together, these findings answer **RQ3.2**: *it is indeed possible to make LLM-based pseudo-relevance feedback both more effective and more efficient* with our PromptPRF framework.

While PromptPRF demonstrates strong performance and deployability, it also brings forward new challenges and design trade-offs, such as stability across depths, query drift, feature selection, and the interplay between retriever capacity and feedback noise. These open questions are not unique to PromptPRF but span across all PRF methods explored in this thesis. The next chapter, **Challenges and Case Studies for Pseudo-Relevance Feedback**, will take a closer look into these limitations and failures. Through comprehensive case studies and qualitative analyses, we aim to provide a deeper understanding of when, how, and why different PRF strategies succeed or fail in neural retrieval settings.

Part 4

Analysis, Conclusion, and Future Work

Introduction to Part 4

In this final part of the thesis, we aim to examine the broader implications of our findings by addressing the unresolved challenges and analyzing the failure cases across the various PRF methods we have introduced in the earlier chapters. While PRF has demonstrated its potential to enhance the retrieval effectiveness in neural retrieval settings, its performance remains inconsistent in certain scenarios. Through a comprehensive evaluation, we aim to identify the persistent limitations that prevent PRF from delivering its full potential consistently, including the sensitivity to query ambiguity, the instability of feedback signals, and the tendency towards query drift. These issues become especially significant when PRF is applied in the complex neural retrieval settings which involve dense representations or large language models.

To gain a deeper understanding of these challenges, we conduct an extensive set of case studies that are aimed at investigating the causes of the failures. These case studies span multiple retrieval architectures and datasets, offering insights into when and why PRF fails to offer meaningful improvements. We observe that when using the ambiguous or under-specified queries, they often lead to weak and misleading feedback signals, where the expansion signals reinforce the non-relevant directions in the semantic space. Additionally, we find that certain dense retrievers are more prone to drift even with the top-ranked feedback passages that are only marginally relevant. By systematically characterizing these failure patterns, we aim to provide an empirical grounding for the theoretical limitations identified throughout the thesis.

Building on this critical analysis, we aim to synthesize the main contributions of this thesis in a cohesive and reflective conclusion. We revisit the central research questions posed at the beginning of the thesis and illustrate how they have been addressed through the proposed methods and experimental findings. Our work demonstrates that PRF can be successfully adapted to the transformer-based rerankers, dense retrievers, and even zero-shot LLM-based retrieval pipelines, with computational constraints considered. We also emphasize the importance of feedback modularity and generalizability, as captured by our proposed frameworks, in enabling systematic exploration of design choices across architectures.

Finally, we outline several directions for future research that emerge from the limitations and opportunities identified in this thesis. These include the development of more robust feedback mechanisms capable of adapting to query-specific uncertainty, the strategies for integrating reliability estimation directly into the feedback loop, and the scalable designs for real-time or resource-constrained environments. Furthermore, as large language models continue to reshape the landscape of neural information retrieval, we anticipate a growing need for methods that combine their generative capabilities with lightweight, interpretable feedback structures. These future directions highlight the evolving nature of PRF as a fundamental component of neural information retrieval, with broad implications for both research and deployment in modern search systems.

Chapter 12

Challenges and Case Studies for Pseudo-Relevance Feedback

This chapter examines the practical limitations and failure cases of the PRF methods proposed in this thesis. While prior chapters have demonstrated that PRF can consistently enhance retrieval effectiveness across different retrievers, it is equally important to understand the boundary conditions under which these methods fail or underperform. Such understanding not only provides critical insights into the robustness and generalizability of PRF but also reveals open challenges for future work. We aim to address the following research question:

RQ3.3: What are the key challenges in adapting Pseudo-Relevance Feedback to neural information retrieval pipeline?

We conduct a series of targeted case studies to analyze the behaviour of three representative PRF approaches introduced in this thesis: VPRF (Chapter 5), TPRF (Chapter 8), and PromptPRF (Chapter 11). Each case study investigates both performance improvements and degradations at the query level, aiming to isolate and characterize the factors that contribute to PRF effectiveness or failure. These analyses are conducted with respect to query ambiguity, feedback signal quality, retrieval overlap, and representation shift, which have emerged as recurring themes throughout our empirical studies.

Section 12.1 presents a representative case study by applying VPRF to the ANCE dense retriever. This analysis seeks to understand when, and potentially why, the method leads to performance gains or degradations by examining query-level performance shifts in relation to representation similarity, top-k retrieval overlap, and the limitations of these factors in explaining PRF effectiveness.

Section 12.2 presents a case study of TPRF, with a focus on understanding the conditions under which it improves or degrades retrieval performance. The analysis evaluates TPRF's behavior across three dense retrievers on both TREC DL 2019 and 2020, and highlights how its effectiveness depends on the quality, diversity, and alignment of top-ranked feedback passages. The case study reveals that TPRF is capable of capturing and amplifying useful semantic signals when the initial retrieval is topically coherent and lexically rich. In contrast, failure cases are often driven by overemphasis on

specific terms, semantic drift due to feedback repetition, or ranking distortions caused by unjudged passages. This examination provides deeper insight into TPRF's strengths, limitations, and the retrieval settings where it is most effective.

Section 12.3 presents a case study of PromptPRF applied to decoder-style LLM dense retrievers, focusing on three widely used LLM families: QWEN2.5, QWEN3, and GEMMA3. The analysis examines when the method enhances or degrades performance by identifying query-level patterns of effectiveness change, with particular attention to feedback quality, alignment between feedback and query intent, and failure cases driven by ambiguity-induced query drift.

Overall, these case studies offer a complementary perspective to the quantitative evaluations presented in previous chapters. By analyzing representative failure modes, we aim to refine our understanding of when and why the proposed methods are effective, and when they are not.

12.1 Vector-Based Pseudo-Relevance Feedback Case Study

To complement the quantitative results presented in Chapter 5 and independent of the core research questions, we conduct a targeted case study to gain a qualitative understanding of when, and potentially why, Vector-based Pseudo-Relevance Feedback (VPRF) is effective or ineffective. This analysis focuses on the retrieval task using the ANCE dense retriever and VPRF-Rocchio, to better illustrate the differences between before-PRF and after-PRF of top-ranked passages, we use a moderate feedback depth of $k = 5$ with optimal configuration ($\alpha = 0.4$, $\beta = 0.6$). Figure 12.1 (top) presents a query-level comparison of performance gains and losses introduced by VPRF-Rocchio relative to ANCE on the TREC DL 2019 and 2020 datasets, evaluated using nDCG@5. This cut-off is chosen to match the feedback depth and to better isolate the impact of the feedback signal. Similar trends were observed with other rank cut-offs.

As expected, the application of Vector-based PRF results in performance improvements for some queries, degradations for others, and no observable change for a substantial number of queries. To better understand the neutral cases, we compared the original and PRF-enhanced query representations using inner product similarity. The analysis revealed no consistent relationship between the magnitude of representational change and retrieval effectiveness: queries exhibiting no performance change often displayed representation shifts comparable to those associated with gains or losses. This indicates that alterations in the query embedding alone are insufficient to predict the effectiveness of Vector-based PRF.

For each query, Figure 12.1 (top) also reports the degree of *overlap* between the top-k passages retrieved by the initial ANCE model and those retrieved after applying the Vector-based PRF method. This overlap quantifies the extent to which the PRF-enhanced results retain or replace the original feedback signal. The overlap is computed as the percentage of passages common to both ranked lists at depth k; with $k = 5$, an overlap of 80% indicates that four out of five passages appear in both the original and PRF-enhanced results.

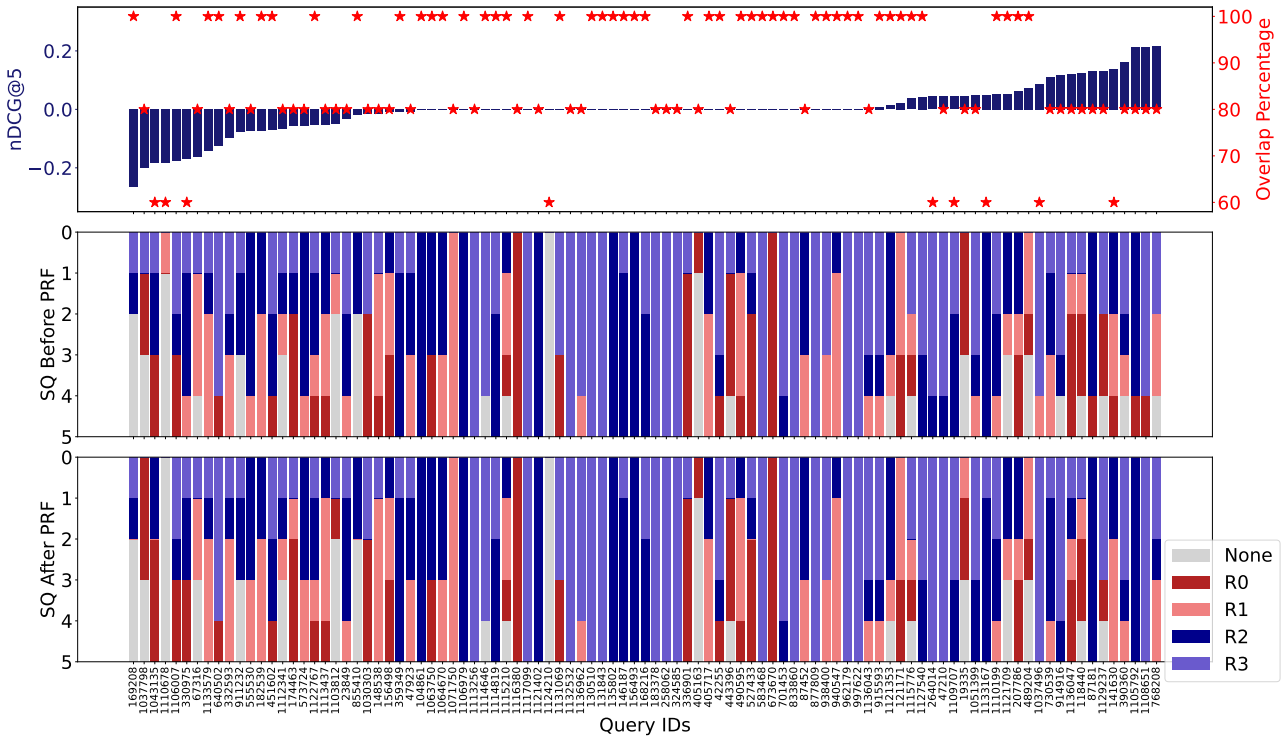


Figure 12.1: The query-by-query analysis on the combined set of TREC DL 2019 and 2020 queries of (1) the gain/loss obtained by Vector-based PRF (Rocchio) with respect to ANCE, as represented by the barplot at the top; (2) the retrieved passages overlap, as represented by the red dots in the top plot; (3) and the analysis of the passage relevance for the feedback signal provided as input to the PRF (top $k = 5$ passages from ANCE – mid plot) and the top 5 passages retrieved by the PRF method (bottom plot). For y-axis, SQ Before/After PRF, SQ stands for Signal Quality. In the legend, None represents Unjudged, R{0, 1, 2, 3} represent the Judged Relevance Level from 0 (Not Relevant) to 3 (Highly Relevant).

This metric allows us to assess whether the PRF method primarily re-ranks the same set of top passages from the first-stage retrieval or introduces new passages into the top ranks. Queries with performance degradation following PRF tend to exhibit higher overlap: for these queries, the average overlap between pre- and post-PRF results is 85.2%. This suggests that the observed losses in effectiveness are not due to the inclusion of fewer relevant passages or the substitution of highly relevant passages with marginally relevant ones. Instead, the degradation appears to stem from suboptimal reordering of passages already present in the initial ANCE retrieval.

In contrast, queries that benefit from the application of PRF tend to show slightly lower average overlap, at 82.8%. This indicates a modest tendency for PRF to promote previously lower-ranked – but potentially more relevant passages into the top-k positions, thereby improving retrieval effectiveness. However, the difference in average overlap between the gain and loss groups is small and not statistically significant, as confirmed by Pearson's Chi-squared test. Thus, while overlap provides some insight into the dynamics of PRF-induced changes, it does not serve as a strong predictor of retrieval success or failure in isolation.

Figure 12.1 (middle) presents an analysis of the relevance of the top-k passages retrieved during the first-stage retrieval, while Figure 12.1 (bottom) shows the relevance distribution of the top-k passages

Table 12.1: Example queries for which the Vector-based PRF (Rocchio) method produces losses, along with example non-relevant passages ranked in the top k from the first stage ranker (and thus used as input to PRF). We only included the representative passages in the table. The number after **Rel** shows the relevance level: 0 - Not Relevant to 3 - Highly Relevant, Unjudged means this passage is not judged in corresponding Qrels.

	Query ID	Query	Top Passage Content
QUERIES FOR WHICH PRF PRODUCES A LOSS	**169208**	does mississippi have an income tax	**2888361**: Unemployment Tax Rates. Reporting and Filing. In Mississippi, the tax rate for a start-up business is 1.00% the first year of liability, 1.10% the second year of liability and 1.20% the third and subsequent years of liability until the employer is eligible for a modified rate (**Rel: Unjudged**)
	1037798	who is robert gray	**2868740**: (Redirected from Gary Leroi Gray) Gary LeRoi Gray (born February 12, 1987) is an actor and voice actor involved with movies, television, and animation. He is most recognized for his childhood role as Nelson Tibideaux, the son of Sondra Huxtable Tibideaux and Elvin Tibideaux on the NBC sitcom The Cosby Show. He appeared on the series during its eighth and final season (1991-1992). (**Rel: Unjudged**)
			2866248: Matthew Gray Gubler. Matthew Gray Gubler is an Emmy award winning actor, director, producer, painter, and voice over actor from Las Vegas Nevada. While studying film directing at NYU he interned for Wes Anderson who gave him his first feature film role as Bill Murray's loyal intern Nico in The Life Aquatic with Steve Zissou (2004). For the past eleven years Gubler ... (**Rel: Unjudged**)
	1110678	what is the un fao	**5253767**: Uno (/EuEnoE/; from Italian and Spanish for 'one') (stylized as UNO) is an American card game that is played with a specially printed deck (see Mau Mau for an almost identical game played with normal playing cards). The game was originally developed in 1971 by Merle Robbins in Reading, Ohio, a suburb of Cincinnati. (**Rel: Unjudged**)
			3386130: Uno (card game) For the video game adaptation, see Uno (video game). Uno (/EuEnoE/; from Italian and Spanish for 'one') (stylized as UNO) is an American card game that is played with a specially printed deck (see Mau Mau for an almost identical game played with normal playing cards). The game was originally developed in 1971 by Merle Robbins in Reading, Ohio, a suburb of Cincinnati. (**Rel: Unjudged**)

retrieved after applying PRF. The relevance of the initial top-k passages effectively reflects the quality of the feedback signal provided to the PRF method. Each stacked column in both figures corresponds to an individual query and is aligned with the query's gain or loss in performance shown in Figure 12.1 (top), enabling a direct comparison between the quality of the feedback signal and the resulting impact of PRF on effectiveness.

At a glance, no consistent pattern emerges. Feedback signals containing non-relevant documents are observed in both gain and loss cases, and in some instances they result in no performance change. That is, the presence of non-relevant passages in the feedback set does not deterministically predict the direction of the PRF effect.

However, a closer examination reveals a more subtle insight. By inspecting the five queries associated with the largest performance losses and the five queries with the largest gains (among those where at least one non-relevant passage appears in the feedback), a clearer distinction can be observed. For queries associated with losses, the non-relevant feedback passages (although sometimes

Table 12.2: Example queries for which the Vector-based PRF (Rocchio) method produces gains, along with example non-relevant passages ranked in the top k from the first stage ranker (and thus used as input to PRF). We only included the representative passages in the table. The number after **Rel** shows the relevance level: 0 - Not Relevant to 3 - Highly Relevant, Unjudged means this passage is not judged in corresponding Qrels.

	Query ID	Query	Top Passage Content
QUERIES FOR WHICH PRF PRODUCES A GAIN	**1129237**	hydrogen is a liquid below what temperature	**8588226**: Hydrogen is a liquid below what temperature? was asked by Shelly Notetaker on May 31 2017. 426 students have viewed the answer on StudySoup. View the answer on StudySoup. Sign Up Login (**Rel: 0**)
			8588222: Answer to: Hydrogen is a liquid below what temperature? By signing up, you'll get thousands of step-by-step solutions to your homework questions.... for Teachers for Schools for Companies (**Rel: 0**)
	87181	causes of left ventricular hypertrophy	**47203**: Causes of Right Ventricular Hypertrophy. There are four usual causes of right ventricular hypertrophy. The first one is pulmonary hypertension. As stated earlier, pulmonary hypertension is a condition wherein the blood pressure increases in the pulmonary artery. And this can lead to shortness of breath, dizziness and fainting. (**Rel: 0**)
	1136047	difference between a company's strategy and business model is	**8724032**: 6. The difference between a company's business model and a company's strategy is that: a. a company's business model is - Answered by a verified Business Tutor (**Rel: 0**)
			8724038: Now, we address our second question. What is the difference between a strategy and a business model? A strategy is about the external logic of a business. How are we going to compete? Check out my earlier post on the elements of a strategy for more about strategies. A business model is about the internal logicâ¦ (**Rel: 0**)
			8724036: Now, we address our second question. What is the difference between a strategy and a business model? A strategy is about the external logic of a business. How are we going to compete? Check out my earlier post on the elements of a strategy for more about strategies. A business model is about the internal logic of the business. See my post on the elements of a business model. Does it make operational and economic sense? Do all of the pieces fit together? A business model is a tool that complements both a business strategy and a business plan. It is an important tool because 1) it ensures that you understand the logic of your business; and, 2) it helps you communicate the logic of your business. Of course, this begs the question, what is a business plan? (**Rel: 0**)

unjudged in Qrels) tend to be severely off-topic and exhibit minimal, if any, semantic alignment with the original query. A representative example is query `1110678: what is the un fao`, for which the PRF signal contains four non-relevant passages. These passages refer to the card game "Uno" and contain no connection to the United Nations Food and Agriculture Organization (FAO). This degree of semantic mismatch dilutes the feedback signal and likely leads to ineffective or even harmful query modification. Representative examples of such non-relevant passages are provided in Table 12.1, along with other similar cases.

This analysis suggests that the impact of non-relevant feedback passages is highly dependent on their semantic proximity to the query intent. While mildly off-topic or tangentially relevant passages may not substantially hinder PRF effectiveness, and can occasionally still offer gains, feedback signals dominated by strongly misaligned content tend to correlate with retrieval performance degradation.

Another illustrative case is query `169208: does mississippi have an income tax`, for which the PRF signal contains three non-relevant passages (we included one representative passage in the table, it is non-relevant although unjudged in Qrels). Although these passages are topically related to taxation and the state of Mississippi, their content focuses on topics such as unemployment tax or start-up tax, which do not address the information need concerning general income taxation. When such misaligned feedback is used to construct the new query representation, the model exhibits signs of *query drift*: that is, the updated query representation places higher scores on passages that are thematically linked to the off-target feedback rather than the original query intent. A similar phenomenon is observed in query `1110678`, where many of the top-ranked passages retrieved after PRF pertain to the card game "UNO", which is a theme introduced by the non-relevant feedback passages, even beyond the initial top-k depth.

In contrast, this form of drift is not observed in queries for which PRF yields substantial gains. Although the feedback signals for such queries may also contain non-relevant passages, the nature of their non-relevance differs significantly. Specifically, these passages tend to exhibit a strong topical or conceptual alignment with the query, despite not being judged relevant in the official TREC assessments.

A representative example is query `1136047: difference between a company's strategy and business model is`, where the PRF method achieves a substantial improvement in effectiveness. Although the feedback set includes several non-relevant passages, as shown in Table 12.2, these passages frequently contain key query terms and demonstrate a clear aboutness relation to the query [144]. This alignment allows the PRF mechanism to reinforce semantically coherent aspects of the query representation, thereby improving retrieval performance.

It is also worth noting that the relevance labels assigned to some of these passages may need some reconsiderations. For instance, passage `8724036` for query `1136047` appears to exhibit at least marginal relevance based on content, even though it was originally judged as non-relevant in the TREC annotations.

Additional queries exhibiting similar characteristics to query `1136047` are provided in Table 12.2, further supporting the observation that the impact of non-relevant passages in PRF is contingent not only on their relevance labels, but also on their semantic proximity to the original query intent.

Summary of Vector-Based PRF Case Study

This case study examined the effectiveness of Vector-based Pseudo-Relevance Feedback using the Rocchio method applied to ANCE, with a focus on query-level performance shifts and their potential causes. Our analysis revealed that applying PRF offers significant improvements, with more improvements for some queries, less degradations for others, and no effect in a substantial number of cases.

From the analysis, one key finding is that changes in query representations, as measured by inner product similarity, are not reliable predictors of PRF effectiveness. Queries that experience no

performance change often show similar degrees of representational shift as those associated with gains or losses, suggesting that representational distance alone is not sufficient to explain the impact of PRF.

We further investigated the degree of top-k passage overlap between the initial and PRF-enhanced retrievals. While losses tend to be associated with higher overlap, which implies ineffective re-ranking of already retrieved content, gains are somewhat more likely to occur when PRF introduces new passages into the top ranks. However, the difference in overlap between these cases was not statistically significant, indicating that overlap alone cannot account for the directionality of PRF outcomes.

A more fine-grained analysis of the feedback signal content revealed a more significant factor: the semantic quality of the feedback passages. In loss cases, non-relevant feedback passages were often topically unrelated or misleading, leading to query drift and a breakdown in alignment between the modified query and the original intent. In contrast, gain cases occasionally included non-relevant passages, but these tended to exhibit topical alignment or contain key query terms, effectively reinforcing the query semantics despite being judged non-relevant by TREC assessors.

These findings hightlight several challenges for Vector-based PRF. First, the method remains sensitive to the quality of the feedback signal, particularly in cases involving semantically off-topic content. Second, its deterministic nature, lacking adaptive mechanisms for assessing or mitigating noisy feedback, makes it vulnerable to degradation under ambiguous or noisy retrieval conditions. Addressing these limitations may require integrating mechanisms for feedback validation, adaptive weighting, or selective filtering to improve robustness.

12.2 Transformer-Based Pseudo-Relevance Feedback Case Study

This section provides a case study of the Transformer-based Pseudo-Relevance Feedback (TPRF) model, focusing on its query-level performance across three dense retrievers: ANCE [223], DistilBERT-Balanced [73], and TCTv2 [115]. The analysis is based on the TREC DL 2019 (DL19) and TREC DL 2020 (DL20) benchmarks. For each configuration, we evaluate the best-performing TPRF variant as identified in Section 8.3.1 of Chapter 8 (see Tables 8.1 and 8.2).

Aggregate Query-Level Analysis

To better understand the distributional behavior of TPRF across queries, we conduct an aggregate comparison between TPRF-enhanced retrieval and the original retrievers on both DL19 and DL20. The analysis focuses on per-query score differences, illustrating how TPRF impacts individual queries under different dense retriever backbones.

Figure 12.2 shows the per-query score differences between ANCE-TPRF and the original ANCE retriever. On DL19, TPRF yields a mean gain of +0.0196, indicating modest improvements across a substantial portion of queries. However, a few queries exhibit performance degradation, reflecting instability in how TPRF incorporates feedback within ANCE's embedding space. On DL20, the overall effect is less favorable, with a much lower mean gain of +0.0089. While some queries benefit from

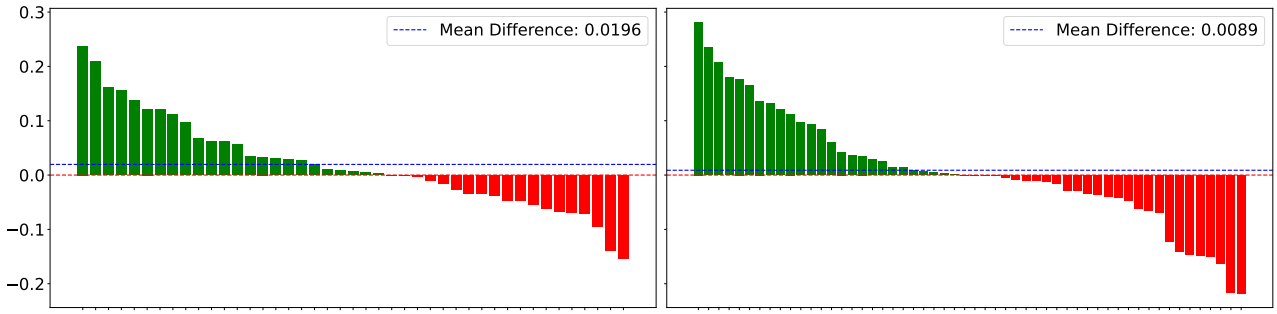


Figure 12.2: Per-query analysis of ANCE-TPRF compared to ANCE for DL19 (left) and DL20 (right). Each bar represents the difference in nDCG@10 for a single query: Green bars indicate queries for which TPRF improves (red bars: degradation). The blue dashed line marks the mean difference. TPRF yields a positive average gain in all settings, with a notable number of large improvements and relatively fewer severe degradations.

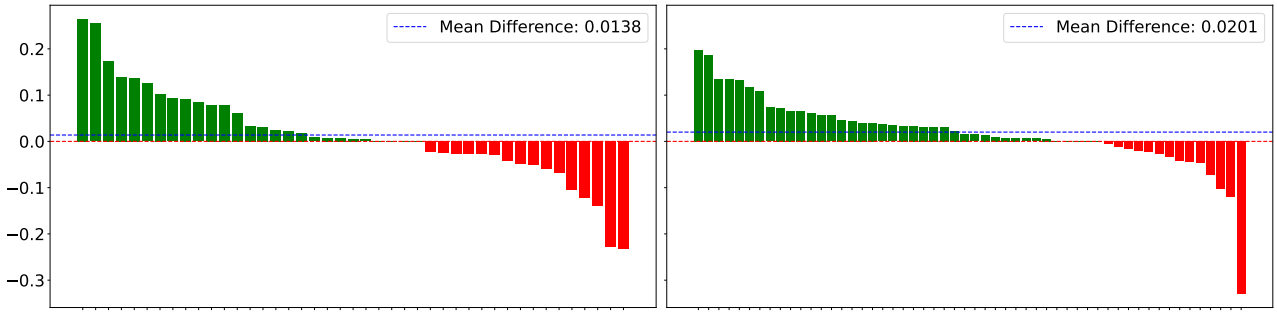


Figure 12.3: Per-query analysis of DistilBERT-Balanced-TPRF compared to DistilBERT-Balanced for DL19 (left) and DL20 (right). Each bar represents the difference in nDCG@10 for a single query: Green bars indicate queries for which TPRF improves (red bars: degradation). The blue dashed line marks the mean difference. TPRF yields a positive average gain in all settings, with a notable number of large improvements and relatively fewer severe degradations.

large gains, an almost equal number of queries suffer substantial losses, which collectively bring down the overall effectiveness. This observation aligns with findings in Chapter 8, where ANCE-TPRF consistently lags behind both DistilBERT-Balanced-TPRF and TCTv2-TPRF. These results suggest that the representations generated by ANCE are less aligned with TPRF's attention mechanism, leading to higher variance and greater sensitivity to the quality and distribution of feedback passages.

Figure 12.3 presents the query-level score differences between DistilBERT-Balanced-TPRF and the base DistilBERT-Balanced retriever. TPRF achieves a mean gain of +0.0138 on DL19 and +0.0201 on DL20, indicating consistent improvements across both datasets. Compared to ANCE, the distribution of gains and losses is more symmetric, with only a single outlier showing substantial degradation. This pattern aligns with findings in Chapter 8, where DistilBERT-Balanced-TPRF consistently outperforms its baseline across multiple metrics. These observations suggest that the dense representations produced by DistilBERT-Balanced are better suited to TPRF's vector-based attention mechanism, enabling more stable and effective integration of feedback signals.

Figure 12.4 shows the per-query performance differences between TCTv2-TPRF and the base TCTv2 retriever. On DL19, TPRF achieves the highest mean gain among all three retrievers, at +0.0245, with the majority of queries exhibiting consistent improvements and relatively few cases of

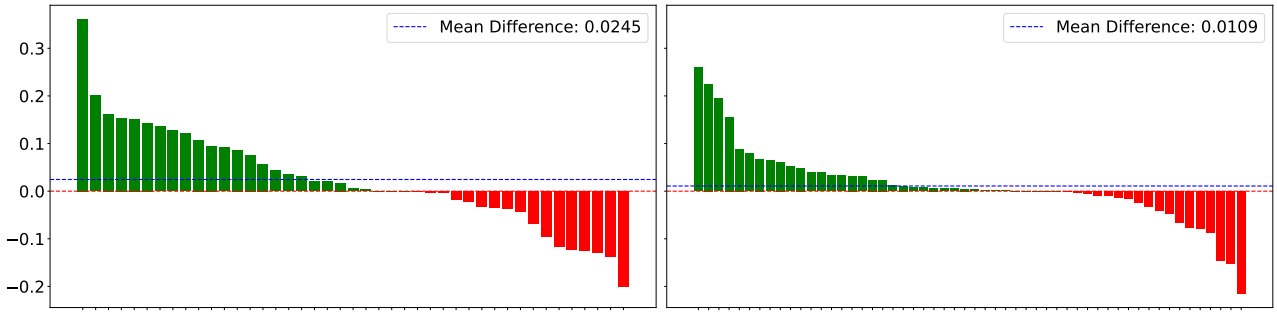


Figure 12.4: Per-query analysis of TCTv2-TPRF compared to TCTv2 for DL19 (left) and DL20 (right). Each bar represents the difference in nDCG@10 for a single query: Green bars indicate queries for which TPRF improves (red bars: degradation). The blue dashed line marks the mean difference. TPRF yields a positive average gain in all settings, with a notable number of large improvements and relatively fewer severe degradations.

substantial degradation. On DL20, the mean gain decreases to +0.0109, yet still surpasses that observed for ANCE-TPRF. These patterns are consistent with the effectiveness trends reported in Chapter 8, where TCTv2-TPRF demonstrates the most stable improvements across metrics and feedback depths. The results suggest that TCTv2's embedding space offers a more compatible foundation for TPRF's attention mechanism, supporting more robust and effective feedback integration.

In summary, these aggregate results indicate that the effectiveness of TPRF is shaped by both dataset characteristics and the representational properties of the underlying retriever. TPRF tends to perform more reliably when paired with retrievers whose embeddings exhibit semantic coherence and are well aligned with TPRF's attention mechanism. The consistently higher mean gains on DL19, which comprises relatively easier and more well-formed queries, suggest that TPRF is more effective when the initial retrieval provides clearer feedback signals. In contrast, the lower gains and increased variance on DL20, composed by more ambiguous and challenging queries, highlight TPRF's sensitivity to feedback quality and relevance distribution. These findings reaffirm the need for both retriever compatibility and feedback robustness when applying TPRF in more difficult retrieval scenarios.

Success Cases

In this subsection, we analyze representative success cases where the TPRF model produces substantial improvements in retrieval effectiveness. These cases are characterized by high-quality top-k feedback passages retrieved by the first-stage dense retriever. Specifically, these passages tend to be highly relevant (almost exclusively annotated with a relevance label of 3), semantically rich, and well-aligned with the user's intent. Importantly, the passages offer complementary rather than redundant content, allowing the TPRF attention mechanism to integrate and amplify key semantic signals into a refined query embedding.

As shown in Table 12.3, the top-ranked feedback passages used as TPRF input already contain strong evidence of relevance. For instance, in the query "*what types of food can you cook sous vide*" (Query ID 915593), all retrieved passages are annotated with a relevance level of 3. These passages collectively describe a wide array of food types (e.g., meat, vegetables, eggs, desserts) that can be

Table 12.3: Example queries for which the TPRF method produces gains, along with example passages ranked in the top k from the first stage ranker (and thus used as input to PRF). We only included the representative passages in the table. The number after **Rel** shows the relevance level: 0 - Not Relevant to 3 - Highly Relevant, unjudged means this passage is not judged in corresponding Qrels.

	Query ID	Query	Top Passage Content
QUERIES FOR WHICH TPRF PRODUCES A GAIN	**915593**	what types of food can you cook sous vide	**3538160**: All kinds! Any type of meat, such as beef, pork, lamb, game, or poultry, is ideal for sous vide. It works especially well with fish and seafood, ensuring that these delicate foods ... any vegetable can also be cooked sous vide with delicious results, as can eggs and many fruits. You can even use it to make ... yogurt, and even cakes. (**Rel: 3**)
			82107: What kind of foods can you cook sous vide? Sous vide is traditionally seen as an alternative method for cooking meats. However, the technique is extremely versatile, meaning all manner of ingredients can be cooked such as: Pork, Lamb, Beef, Chicken, Duck, Turkey, Quail. (**Rel: 3**)
	1133579	how does granulation tissue start	**1817055**: Granulation Tissue. in Pathology. After tissue damage, repair process starts. It can begin as early as 24 hours. Fibroblasts and endothelial cells begin proliferating to form a specialized type of tissue ... can begin as early as 24 hours. Fibroblasts and endothelial cells begin proliferating to form a ... called granulation tissue. The term derives from its pink, soft, granular appearance on surface of wound but its histological features are: (**Rel: 3**)
	330975	how much would it cost to install my own wind turbine	**3297858**: Answer by Answers Publisher. Confidence votes 263K. The estimated cost of installing a wind turbine is $8000 - $10000 for a small turbine and up to $20000 for a larger one. The average wind speed will also determine how expensive a model is required. More affordable wind turbines range from $200 to $600. These turbines are typically smaller, which means less electricity is generated. Larger wind turbines can cost as much as $2,000 for those used in the home and upwards of $20,000 for those used industrially. 1 person found this useful. (**Rel: 3**)
			6693239: The estimated cost of installing a wind turbine is $8000 - $10000 for a small turbine and up to $20000 for a larger one. The average wind speed will also determine how expensive a model is required. (**Rel: 3**)

prepared using sous vide techniques, while also explaining its versatility and cooking process. The content is detailed, well-scoped, and exhibits low lexical redundancy across passages. The TPRF model effectively leverages these complementary signals, as reflected in Table 12.4, where the final ranked output includes highly relevant passages that mirror and reinforce the semantic space of the input.

A similar pattern is evident in the query "*how does granulation tissue start*" (Query ID 1133579). The top-k passages from the initial retrieval explain distinct but interconnected biological mechanisms, including fibroblast activation, wound healing phases, and histological features of granulation tissue. All selected passages are highly relevant (Rel: 3), and their specificity enables TPRF to refine the original query embedding to a richer form. The output from TPRF maintains this level of granularity, retrieving passages that describe the same physiological process from different but highly related perspectives.

The query "*how much would it cost to install my own wind turbine*" (Query ID 330975) offers another representative example. The initial retrieval yields passages containing detailed cost breakdowns

Table 12.4: Same queries for which the TPRF method produces gains, along with example passages ranked in the top k from the second round retrieval (and thus are output from TPRF). We only included the representative passages in the table. The number after **Rel** shows the relevance level: 0 - Not Relevant to 3 - Highly Relevant, unjudged means this passage is not judged in corresponding Qrels.

	Query ID	Query	Top Passage Content
QUERIES FOR WHICH TPRF PRODUCES A GAIN	**915593**	what types of food can you cook sous vide	**8402972**: What is sous vide? It's the easiest, most consistent way to get delicious results in the kitchen, from succulent steaks to flavor-packed veggies and fork-tender salmon. To prepare food sous vide (say: sue veed), we pick our desired doneness, then set a water bath to the appropriate time-and-temperature settings. (**Rel: 3**)
	1133579	how does granulation tissue start	**1200778**: These 10 animal facts will amaze you. Tissue granulation is a process by which fibrous tissue rich with blood capillaries replaces blood clots formed at the site of a healing wound. Through this process, healthy and normal skin is able to ... is called granulation tissue.The tissue that forms usually has a ... during which new, healthy tissue reforms and rebuilds the area of the wound. The final phase of wound healing is the remodeling phase, when the new tissue fully integrates with the old tissue. (**Rel: 3**)
	330975	how much would it cost to install my own wind turbine	**3297857**: The estimated cost of installing a wind turbine is $8000 - $10000 for a small turbine and up to $20000 for a larger one.ore affordable wind turbines range from $200 to $600. These turbines are typically smaller, which means less electricity is generated. Larger wind turbines can cost as much as $2,000 for those used in the home and upwards of $20,000 for those used industrially. 1 person found this useful. (**Rel: 3**)

for both small-scale and large-scale wind turbines, along with contextual factors like turbine size, usage type, and energy output. Again, all passages have a relevance level of 3, and the information is complementary without being repetitive. TPRF is able to encode this diverse set of cost and technical details into a single refined query representation, which results in second-stage retrieval that preserves and even sharpens these relevant aspects (see Table 12.4).

These cases demonstrate that the effectiveness of TPRF relies on the strength and diversity of the initial feedback. When top-k passages are highly relevant and semantically well-aligned with the query, the self-attention mechanism in TPRF is capable of identifying and aggregating distinct but useful content dimensions, producing a richer and more discriminative query embedding. Moreover, because the output passages retrieved after applying TPRF closely resemble the strongest elements of the initial feedback, we observe that TPRF functions as an effective semantic amplifier to magnify and propagate key relevance signals while maintaining robustness against noise.

These findings reinforce the broader conclusion that TPRF is most effective in scenarios where the initial feedback context is well-defined, topically focused, and lexically diverse. The model's strength lies not in recovering missing relevance signals, but in enhancing those that are already embedded in high-quality feedback retrieved during the first stage.

Failure Cases

Although the TPRF model demonstrates strong overall effectiveness, query-level failures reveal several recurring causes of degradation. These include: (i) semantic shifts caused by the attention mechanism disproportionately weighting specific passage-level terms during query reformulation; (ii)

Table 12.5: Example queries for which the TPRF method produces losses, along with example passages ranked in the top k from the first stage ranker (and thus used as input to PRF). We only included the representative passages in the table. The number after **Rel** shows the relevance level: 0 - Not Relevant to 3 - Highly Relevant, unjudged means this passage is not judged in corresponding Qrels.

	Query ID	Query	Top Passage Content
QUERIES FOR WHICH TPRF PRODUCES A LOSS	**1108651**	what the best way to get clothes white	**4911228**: Quick Answer. To remove dye from white clothing, chlorine bleach works best. To use this method, users should add 1/2 cup of bleach to a gallon of water and soak the clothing for 30 minutes; after the clothes have soaked, they should then be washed as normal. For fabrics that are not white, a dye or color remover is optimal. (**Rel: 1**)
			8175407: Here is Most Effective Way To Get The White Clothes White Again. What can do the trick? Check it out and you will be pretty pleased with the results. See More (**Rel: 0**)
	1136043	difference between a hotel and motel	**1056535**: What is the difference between Hostel and Hotel? Hostel is either attached to a university or a college or sometimes situated away from a university or college. Hostel is a place primarily meant for lodging. Hotel is a place meant primarily for boarding. Messes are attached to hostels. Rooms are attached to hotels for the purpose of staying. (**Rel: 1**)
			8728077: Difference Between Hotel and motel Definition of the term hotel: A Hotel or Inn may be defined as an establishment whose primary business is providing lodging facilities for the general public, and which furnishes one or more of the following services. (**Rel: 1**)
	1043135	who killed nicholas ii of russia	**5262285**: Nicholas II of Russia (May 18, 1868 - July 17, 1918) (Russian: II, Nikolay II) was the last tsar of Russia, the King of Poland, and Grand Duke of Finland. He ruled from 1894 until his forced abdication in 1917. (**Rel: 0**)
			5262282: Nicholas II of Russia (May 18, 1868 - July 17, 1918) (Russian: II, Nikolay II) was the last tsar of Russia, the King of Poland, and Grand Duke of Finland. He ruled from 1894 until his forced abdication in 1917. Nicholas proved unable to manage a country in political turmoil and to command its army during World War I. (**Rel: 0**)

the promotion of unjudged passages at the expense of judged relevant ones, which lowers evaluation scores despite potential topical relevance; and (iii) topic drift resulting from repeated or dominant terms in the initial feedback set, this corpus-level redundancy shifts the refined query representation toward the most frequently occurring concepts rather than the original intent.

As illustrated in Table 12.5, the query "*what the best way to get clothes white*" (Query ID 1108651) shows a semantic shift in focus after applying TPRF on the passage-level. Some of the initial top-ranked passages address methods for removing dye from white clothes, such as the use of bleach and other fabric treatments (e.g., Passage ID **4911228** with Rel: 1). However, the overall relevance remains limited, as several other top-ranked passages fail to provide the actual methods for whitening clothes (e.g., Passage ID **8175407** with Rel: 0). Among these passages, terms like "*dye*" and "*white clothes*" appear with higher frequency. Therefore, the TPRF model tends to overemphasize these terms during attention-based reformulation, resulting in a diverted query that shifts from the intended goal, which is how to make clothes white again, to a narrower focus on how to wash or care for already white clothes. The second-stage outputs (Table 12.6) hence retrieves passages discussing general laundry procedures,

Table 12.6: Same queries for which the TPRF method produces losses, along with example passages ranked in the top k from the second round retrieval (and thus are output from TPRF). We only included the representative passages in the table. The number after **Rel** shows the relevance level: 0 - Not Relevant to 3 - Highly Relevant, unjudged means this passage is not judged in corresponding Qrels.

	Query ID	Query	Top Passage Content
QUERIES FOR WHICH TPRF PRODUCES A LOSS	**1108651**	what the best way to get clothes white	**1457502**: Report Abuse. You can wash brown clothes with other dark clothes. If you mix white clothes with bright colored clothes, the dyes in bright clothes could bleed into your white clothes. White clothes and towels are often washed in hot water ... Towels, sheets, underwear, etc. are usually washed in hot water to sanitize them-but a hot dryer works for that too ... White clothes and towels are often washed in hot water ... Towels, sheets, underwear, etc. are usually washed in hot water to sanitize them-but a hot dryer works for that too. (**Rel: 0**)
	1136043	difference between a hotel and motel	**8728074**: Difference Between Hotel and motel. 1 Food and beverage service. 2 Room attendant (House keeping) service. 3 Concierge. 4 Laundry or dry cleaning service. 5 Use of furniture or fixtures. 6 Bell and Door attendant service. 7 Conference and Banqueting. 8 Business centre etc. (**Rel: Unjudged**)
			852195: Best Answer: A hotel is a facility that provides accommodations (bed & bath basically) for transients like travellers and tourists. e.g., Holiday Inns all over the USA. A resort is a recreational complex that normally has a center of attraction, like a beach, a garden park, a swimming pool and a restaurant. e.g., Boracay Beach Resort in the Philippines A resort hotel combines the amenities of a regular hotel and a resort. (**Rel: Unjudged**)
	1043135	who killed nicholas ii of russia	**2514953**: Nicholas II abdicated following the February Revolution of 1917 during which he and his family were imprisoned first in the Alexander Palace at Tsarskoye Selo, then later in the Governor's Mansion in Tobolsk, and finally at the Ipatiev House in Yekaterinburg. (**Rel: Unjudged**)
			2664994: Nicholas II of Russia. Nicholas II of Russia, (May 18, 1868 - July 17, 1918) was the last Tsar or Emperor of the Russian Empire. He became Tsar in 1894 after his father, Tsar Alexander III died. He married Princess Alix of Hesse, who was the granddaughter of Queen Victoria, and they had five children, Olga, Tatiana, Maria, Anastasia, and Alexi. (**Rel: 0**)

fabric maintenance, and dye transfer prevention. While topically related, these results are misaligned with the original whitening intent, leading to degraded retrieval performance.

A second major source of degradation stems from the introduction of a large number of unjudged passages in the TPRF output. This is observable in the query "*difference between a hotel and motel*" (Query ID 1136043). The initial feedback (Table 12.5) includes passages that provide basic but partially relevant definitions (e.g., Passage ID **1056535** and **8728077** with Rel: 1). After applying TPRF (Table 12.6), the reformulated query retrieves a broader mix of content, many of them are labeled as unjudged. Although some of these passages appear reasonably related (e.g., Passage ID **8728074**), their inclusion pushes previously judged passages further down in the ranking. Since unjudged passages are excluded from evaluation metrics like nDCG, their presence negatively impacts the measured effectiveness. Thus, the drop in effectiveness may not be due to a semantic error, but rather a result of the lack of judgment coverage in the dataset.

The third observed failure mode involves semantic shift caused by feedback term reinforcement on a corpus-level. In the query "*who killed nicholas ii of russia*" (Query ID 1043135), the top-k

passages from the initial retrieval all focus heavily on Nicholas II's biography, repeatedly mentioning his name and title (e.g., Passage ID **5262285** and Passage ID **5262282** with Rel: 0). The redundancy of these terms in the input passages causes TPRF to overly amplify the topic of "*Nicholas II*" rather than the actual target of the query, i.e., identifying the person who killed him. Therefore, the second-stage output (Table 12.6) includes more detailed biographical content, but continues to omit relevant information about the killer's information. In this case, the attention mechanism reinforces the wrong semantic signal, further diverting the retrieval process from the intended question.

These failure cases reaffirms the importance of the quality and semantic alignment of initial feedback passages. When feedback contains repetitive or ambiguous terms, or when relevant passages are sparsely judged, TPRF may struggle to form a precise and focused query embedding. The model's reliance on attention over static vectors can lead to undesirable shifts in emphasis, ranking distortion, or the amplification of marginally related concepts. These limitations point to potential avenues for improvement, such as relevance-aware gating mechanisms, filtering of unjudged content, or contextual regularization to prevent semantic drift during feedback integration.

Summary of Transformer-Based PRF Case Study

This case study examined the query-level behavior of the Transformer-Based Pseudo-Relevance Feedback (TPRF) model across three dense retrievers (ANCE, DistilBERT-Balanced, and TCTv2) on the TREC DL 2019 and 2020 datasets. Through aggregate and query-specific analysis, we identified both strengths and limitations of TPRF in practical retrieval scenarios.

TPRF consistently achieved substantial gains across most queries when paired with suitable retrievers, particularly DistilBERT-Balanced and TCTv2. The effectiveness was especially observable in cases where the top-k feedback passages were highly relevant, topically aligned with the query intent, and lexically diverse. In these success cases, TPRF was able to effectively integrate distinct yet complementary signals, offering improved query representations and second-stage retrieval performance.

However, our analysis also revealed several recurring failure modes. These included (i) semantic drift caused by overemphasis on specific terms on passage-level, (ii) the promotion of unjudged passages at the expense of judged relevant ones, and (iii) shifts in query intent driven by term repetition in the feedback set on corpus-level. These issues often led to degraded retrieval performance despite the presence of potentially relevant content. In particular, TPRF demonstrated sensitivity to the quality and distribution of feedback signals.

In summary, these findings suggest that TPRF is most effective in settings where the initial retrieval provides strong, focused, and diverse feedback signals. The model's reliance on static embeddings and self-attention makes it vulnerable to both semantic misalignment and judgment sparsity.

12.3 Prompt-Based Pseudo-Relevance Feedback Case Study

This section presents a detailed case study of PromptPRF, examining its query-level performance relative to the Initial Retrieval (No PRF) baseline. The analysis is conducted using QWEN2.5, QWEN3, and GEMMA3 series dense retrievers on the TREC DL 2019 (DL19) and 2020 (DL20) datasets. For each setting, we adopt the best-performing feature model identified in Section 11.4.1 in Chapter 11 (see Table 11.4 and 11.5).

Aggregate Query-Level Analysis

Unlike traditional PRF techniques, PromptPRF operates by leveraging pre-generated, query-independent feature representations extracted from top-k passages. This architecture offers practical advantages in efficiency, particularly under strict latency or computational constraints, and promotes modularity across retrievers. However, this decoupling of feedback representation from query context introduces uncertainty about the semantic alignment between the feature vectors and the original query intent.

To investigate this, we complement the aggregate effectiveness evaluations with a query-level gain/loss analysis. This approach allows us to examine not only where PromptPRF is effective, but also to characterize the conditions under which it fails. In particular, we analyze whether observed performance differences are associated with the nature of the feedback signal and its interaction with the query semantics.

Figure 12.5, Figure 12.6, and Figure 12.7 show per-query nDCG@10 differences between PromptPRF and the No PRF baseline for QWEN2.5, QWEN3, and GEMMA3, respectively. Queries are sorted by the magnitude of effectiveness change. Each bar represents the nDCG@10 difference for a given query, with green indicating performance improvement and red indicating degradation. This breakdown highlights that while PromptPRF often offers meaningful gains, it also results in performance drops for a non-trivial subset of queries, warranting further examination of the underlying causes.

We make the following observations based on the query-level effectiveness comparisons:

1. **Impact on Mid-Sized Models (QWEN2.5 14B):** As shown in Figure 12.5 (left), the QWEN2.5 14B retriever on DL19 exhibits the most dramatic improvements. The majority of queries show positive gains, resulting in a substantial mean difference of +0.1760. The scarcity of negative bars suggests that for mid-sized models with sufficient capacity to interpret feedback but imperfect zero-shot coverage, PromptPRF provides a highly reliable boosting signal.

2. **Diminishing Returns on Large Models (QWEN2.5 72B):** In contrast, the QWEN2.5 72B model on DL20 (Figure 12.5, right) shows a more moderate mean improvement of +0.0389. While the majority of queries still benefit (green bars), the magnitude of these gains is smaller, and a noticeable tail of performance degradation (red bars) appears. This indicates that as the base retriever becomes more powerful, the room for improvement shrinks, and the risk of feedback introducing noise relative to the model's internal knowledge increases.

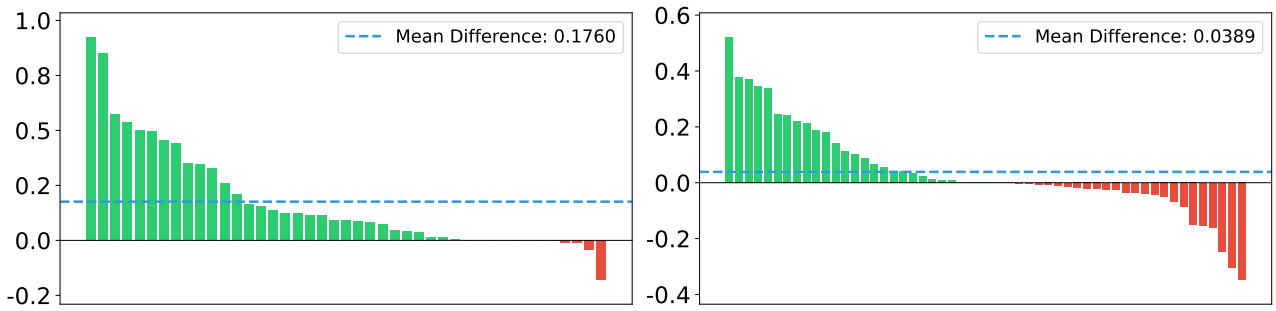


Figure 12.5: Per-query analysis of PromptPRF compared to No PRF (PromptReps) for DL19 (left) and DL20 (right) using QWEN2.5 series dense retrievers (14B and 72B). Each bar represents the difference in nDCG@10 for a single query: Green bars indicate queries for which PromptPRF improves (red bars: degradation). The blue dashed line marks the mean difference. PromptPRF yields a positive average gain in all settings, with a notable number of large improvements and relatively fewer severe degradations.

3. **Consistent Gains across Families (QWEN3 and GEMMA3):** Figures 12.6 and 12.7 confirm the robustness of the method. The QWEN3 4B model (DL19) achieves a mean gain of +0.0620, while the GEMMA3 27B model (DL20) achieves +0.0559. In both cases, the distribution is skewed positively, though distinct failure cases (large red bars) persist at the tail end of the distribution, particularly for the QWEN3 4B model where certain queries suffer significant degradation.

These observations imply that PromptPRF acts primarily as a ranking correction mechanism. The substantial gains for mid-sized models (e.g., QWEN2.5 14B) suggest that while these retrievers successfully identify relevant content within the top-k candidates, they often fail to place them at the very top ranks. In such scenarios, PromptPRF leverages these under-ranked but relevant passages to refine the query vector, steering the retrieval focus more effectively. Conversely, for larger models that already achieve high zero-shot ranking precision, the potential for correction is limited, leading to diminishing returns.

However, non-trivial performance degradations are also observed across all model families. Manual inspection indicates that these failures tend to cluster around queries that are either ambiguous or overly broad. In such cases, the pre-generated feedback features may introduce semantic drift, misaligning the revised query representation with the user's intent. These findings highlight both the potential and limitations of PromptPRF, especially in handling semantically underspecified queries.

Success Cases

PromptPRF demonstrates substantial effectiveness gains when the top-ranked feedback passages are semantically coherent with the query and when the extracted features, such as `Entities` and `Facts`, preserve and reinforce the core query intent. In such cases, the feedback-derived features act as effective signals to refine the query representation, enhancing alignment with relevant passages. Representative examples identified from the query-level analysis include:

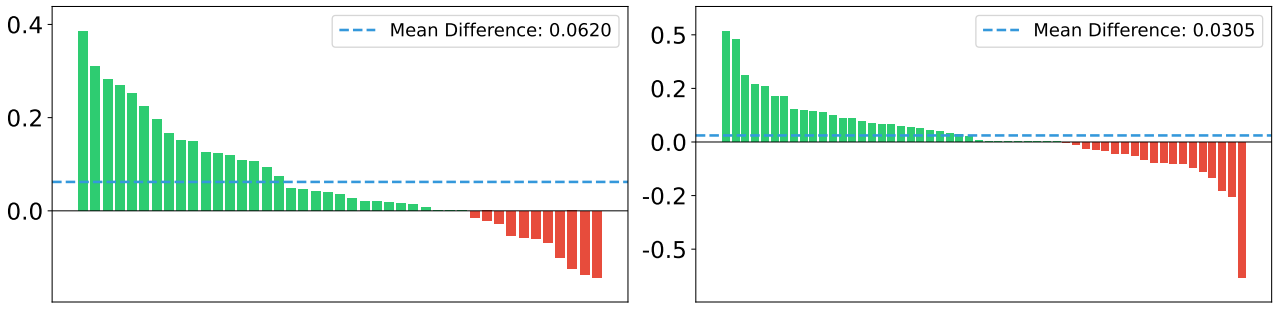


Figure 12.6: Per-query analysis of PromptPRF compared to No PRF (PromptReps) for DL19 (left) and DL20 (right) using QWEN3 series dense retrievers (4B and 8B). Each bar represents the difference in nDCG@10 for a single query: Green bars indicate queries for which PromptPRF improves (red bars: degradation). The blue dashed line marks the mean difference. PromptPRF yields a positive average gain in all settings, with a notable number of large improvements and relatively fewer severe degradations.

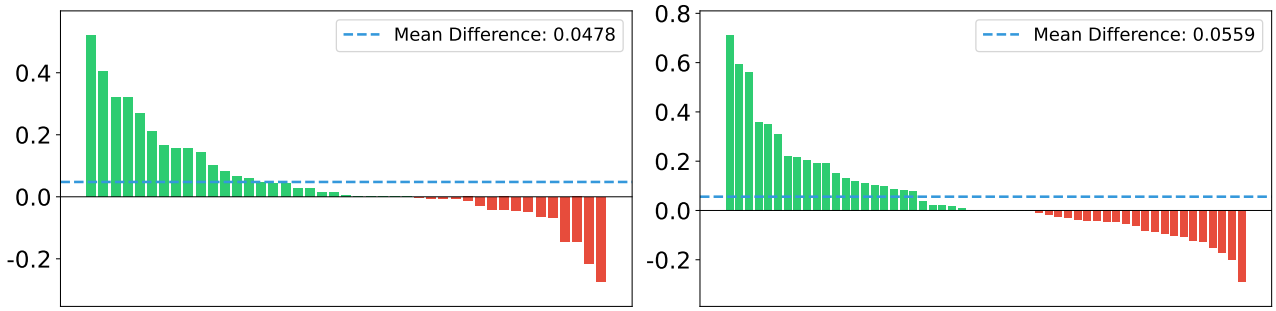


Figure 12.7: Per-query analysis of PromptPRF compared to No PRF (PromptReps) for DL19 (left) and DL20 (right) using GEMMA3 series dense retrievers (12B and 27B). Each bar represents the difference in nDCG@10 for a single query: Green bars indicate queries for which PromptPRF improves (red bars: degradation). The blue dashed line marks the mean difference. PromptPRF yields a positive average gain in all settings, with a notable number of large improvements and relatively fewer severe degradations.

1. **DL19, Query** `87181: causes of left ventricular hypertrophy` **(Figure 12.5, Left)**. Using the QWEN2.5 14B retriever, PromptPRF yields a massive improvement of +0.9243 in nDCG@10. The initial retrieval identifies passages containing specific medical causes such as hypertension (high blood pressure) and aortic valve stenosis. The extracted features capture these clinical entities effectively, enabling the updated query representation to shift focus from a general medical inquiry to precise physiological mechanisms, resulting in a nearly perfect ranking of relevant documents.

2. **DL20, Query** `1113256: what is reba mcentire's net worth` **(Figure 12.7, Right)**. Using the GEMMA3 27B retriever, PromptPRF achieves a gain of +0.7100 in nDCG@10. The top-ranked passages contain specific financial figures and biographical details regarding the singer's assets and career earnings. By extracting `Entities` and `Facts` (e.g., "$95 million", "country singer", "acting career"), PromptPRF injects precise constraints into the query vector. This ensures that the dense retriever prioritizes documents containing the exact answer over generic biographical pages that lack the specific financial data requested.

These examples illustrate the strengths of PromptPRF when applied in settings where the feedback signal is both topically relevant and terminologically rich. In such cases, the resulting query vector is not only semantically well-formed but also better tuned to retrieve passages that meet the user's information need. The modular design of PromptPRF allows it to benefit from structured feedback even when using fixed, query-agnostic feature templates, provided that the top-k passages are contextually appropriate.

Failure Cases

In contrast to the success cases, degradations in retrieval performance are frequently associated with query ambiguity or specific entity queries where the feedback signal introduces noise rather than precision. In such cases, PromptPRF introduces *query drift*, which often lead to the retrieval of top-ranked passages that are potentially related to the broader topic but fail to address the query's specific intent. Representative examples from the query-level analysis include:

1. **DL19, Query** `1115776: what is an aml surveillance analyst` **(Figure 12.7, Left)**. Using the GEMMA3 12B retriever, PromptPRF results in a performance drop of -0.2741. The acronym "AML" is contextually ambiguous (Anti-Money Laundering vs. Acute Myeloid Leukemia). While the query targets a specific financial compliance role, the offline features generated from top-ranked passages drifted towards generic "analyst" descriptions or introduced mixed signals regarding medical AML. This semantic dilution causes the dense retriever to rank generic job descriptions higher than the specific definition sought.

2. **DL20, Query** `997622: where is the show shameless filmed` **(Figure 12.6, Right)**. Using the QWEN3 8B retriever, this query exhibits a massive degradation of -0.6316 in nDCG@10. The query targets a specific filming location (Chicago for the US version, or Manchester for the UK version). PromptPRF broadened the scope by extracting generic features related to the show's plot, characters, or general production trivia rather than the specific geographic entity. This "topic broadening" distracts the retriever, pushing relevant location-specific passages down the ranking in favor of general reviews or episode summaries.

3. **DL20, Query** `701453: what is a statutory deed` **(Figure 12.5, Right)**. Using the QWEN2.5 72B retriever, PromptPRF results in a loss of -0.3480. Although this query targets a precise legal definition (a deed defined by statute, often implying specific warranties), the feedback features shifted focus toward broader real estate or property law concepts. By reinforcing generic terms like "property transfer" or "real estate" without preserving the specific legal term of "statutory," the updated query embedding becomes less discriminative, retrieving tangentially related legal documents instead of the precise definition.

These failure cases illustrate a recurring pattern: when the initial top-ranked passages are semantically diffuse or contain ambiguous acronyms, PromptPRF's reliance on query-independent feature extraction can amplify the breadth rather than refining the specificity of retrieval. Because the features

are derived without direct conditioning on the query, they may reinforce peripheral or non-central aspects of the feedback content, leading to misalignment in the updated query representation.

Furthermore, these examples suggest that even when LLM-extracted features are linguistically or semantically rich, they can degrade performance if they fail to capture the precise informational focus of the query. This highlights a key challenge in prompt-based feedback design: the need to balance generalizable, reusable features with query-aware contextualization to mitigate semantic drift, particularly in cases involving ambiguous entities or highly specific fact-checking needs.

Failure Due to Passage Type

While PromptPRF demonstrates consistent effectiveness gains across a wide range of BEIR datasets, we observe notable performance degradations on specific benchmarks such as QUORA (see Figures 11.6, 11.7, 11.8 and 11.9 in Section 11.4.4, Chapter 11). A recurring failure mode in these cases stems from the syntactic and discourse characteristics of the corpus passages, many of which are phrased as questions rather than declarative statements. This stylistic framing introduces ambiguity that reduces the utility of such passages when used as PRF input, particularly in the context of LLM-based feature extraction.

A representative example is passage 4849 from QUORA: *"What is a Marketing Director?"* When this passage is used as input for keyword feature generation, even a state-of-the-art LLM such as QWEN2.5 72B fails to produce a meaningful output. Instead of extracting meaningful concepts, the model responds with a meta-level prompt interpretation:

```
"Unfortunately, I don't see a passage provided.  Please provide the passage
about a Marketing Director, and I'll be happy to generate a bullet-point list
of relevant keywords for you!"
```

This failure illustrates a broader issue in LLM-based prompt construction: the model interprets the interrogative passage not as a content-bearing passage, but as an instruction or incomplete prompt. As a result, the output fails to reflect any semantic understanding of the passage and does not contain usable features for feedback. When such outputs are incorporated into PromptPRF, they introduce semantic noise into the revised query representation, triggering query drift and resulting in sharp performance degradation.

This issue highlights a broader challenge for PRF methods that rely on LLM-based generative rewriting or feature extraction from raw passage text. The effectiveness of such methods is highly sensitive to the structural and pragmatic properties of the input text. In datasets like QUORA, where interrogative or conversational styles dominate, the LLM may misinterpret the functional role of the passage, treating it as a prompt requiring completion rather than as content to analyze. This misalignment leads to generation failures and degrades the reliability of the feedback signal.

These findings highlight the need for more robust pre-processing or input conditioning strategies in PromptPRF. Specifically, mechanisms for detecting and filtering ill-formed or syntactically ambiguous passages, such as those dominated by questions or fragmentary statements, may help prevent

the inclusion of non-informative feedback content. Incorporating such mechanisms could improve PromptPRF's reliability in noisy or stylistically heterogeneous corpora.

Summary of Prompt-Based PRF Case Study

This case study examined the query-level behavior of PromptPRF when applied to QWEN2.5, QWEN3, and GEMMA3 dense retrievers on the TREC DL 2019 and 2020 datasets. The analysis focused on identifying where PromptPRF improves or degrades retrieval effectiveness, and on understanding the underlying causes of these outcomes.

PromptPRF was shown to produce consistent effectiveness gains across a majority of queries, particularly when the initial top-ranked passages contained semantically rich and query-relevant content. In such cases, the pre-extracted features, such as `Keywords` and `Entities`, aligned closely with the query intent, enhancing the revised query representation and leading to substantial improvements in nDCG@10. This effect was especially observable for mid-sized retrievers, such as the QWEN2.5 14B, where PromptPRF provided highly consistent gains, effectively bridging the performance gap to significantly larger architectures like the QWEN2.5 72B.

However, the analysis also revealed non-trivial failure cases. Performance degradation was frequently associated with ambiguous acronyms (e.g., "AML") or highly specific entity queries (e.g., filming locations), where the feedback signal introduced semantic drift by emphasizing peripheral or unrelated aspects of the topic. This misalignment led to the retrieval of passages that failed to address the user's core information need. Additionally, the diminishing returns observed on the largest models (e.g., QWEN2.5 72B) suggest that as the base retriever's zero-shot capacity increases, it becomes more sensitive to noise introduced by imperfect feedback features.

These findings highlight both the strengths and limitations of PromptPRF. While it offers a lightweight and modular alternative to traditional PRF techniques, its effectiveness is still closely tied to the semantic precision of the feedback signal. Addressing these limitations, particularly the risks of topic broadening and ambiguity resolution, remains a key challenge for improving the robustness of prompt-based PRF methods in real-world retrieval scenarios.

12.4 Summary

This chapter presented a qualitative analysis of failure cases and limitations across three classes of Pseudo-Relevance Feedback (PRF) methods introduced in this thesis, namely Vector-based PRF (VPRF), Transformer-based PRF (TPRF), and Prompt-based PRF (PromptPRF). The case studies complemented prior quantitative evaluations by highlighting specific query-level behaviors and failure patterns.

Section 12.1 showed that VPRF, while effective in some settings, is prone to query drift when feedback passages are only loosely related to the query. Without relevance-aware filtering, even small embedding changes can lead to substantial semantic divergence.

Section 12.2 analyzed TPRF across three dense retrievers. Gains were observed when feedback passages were relevant, lexically diverse, and topically focused. However, TPRF frequently failed due to overemphasis on specific terms, the elevation of unjudged passages, and term repetition in the initial feedback passage set, which introduced semantic drift and ranking distortion.

Section 12.3 demonstrated that PromptPRF performs well when extracted features are structurally complete and aligned with the query, but suffers from degradation on datasets with question-framed or vague passages. Failures often stemmed from ambiguous queries or structurally poor feedback that may confuse the generative LLMs.

Overall, these findings highlight an important reality: despite the architectural innovations introduced by our approaches, such as the generative feature expansion in PromptPRF and the learned weighting in TPRF, the fundamental dependency on feedback quality persists. While these methods successfully transcend the lexical limitations of traditional PRF, they remain vulnerable to semantic drift and topic broadening within the dense vector space. This indicates that transitioning to latent representations does not inherently immunize the retrieval pipeline against noise. Therefore, addressing these challenges requires future explorations into explicit mechanisms for detecting and filtering semantic misalignment to further robustify neural feedback loops.

Chapter 13

Conclusions and Research Overview

The primary objective of this thesis is to advance PRF methods for neural information retrieval by systematically investigating their integration, practicality, and applicability across evolving retrieval architectures. Specifically, the thesis focuses on three key dimensions structured around the following three high-level research questions:

- **RQ1: How can Pseudo-Relevance Feedback be integrated into neural models?**
- **RQ2: How to make Pseudo-Relevance Feedback models more practical and extensible?**
- **RQ3: Are Pseudo-Relevance Feedback models still applicable to decoder-style Large Language Models?**

13.1 Integrating Pseudo-Relevance Feedback into Neural Retrieval Models

In Part 1, we investigated the integration of PRF into neural retrieval models, addressing **RQ1: How can Pseudo-Relevance Feedback be integrated into neural models?** This part comprises four chapters that introduce and evaluate a range of PRF strategies, Text-based (Chapter 3), Vector-based (Chapter 5), and hybrid Sparse-Dense methods (Chapter 6), offering a systematic comparison of their effectiveness, generalisability, and practicality within modern neural IR architectures.

Chapter 3 focuses on the integration of PRF into transformer-based rerankers via query reformulation. A two-stage reranking pipeline is included (Figure 3.1), where a BERT-based reranker [155] is applied first to the original query and then to an expanded query formed from feedback passages. To address the input length constraints of BERT-based models, three text handling strategies are proposed: Concatenate and Truncate (CT) (Eq. 3.1), Concatenate and Aggregate (CA) (Eq. 3.2), and Sliding Window (SW) (Eq. 3.3). In conjunction with these, three score aggregation methods, Average, Borda, and Max, are employed to consolidate rankings over multiple segments. Our experiments show that CA and CT yield notable improvements on TREC DL 2019. However, these gains are inconsistent across

datasets such as DL HARD and WebAP. Moreover, the requirement of an additional reranking stage introduces substantial inference latency, limiting the feasibility of these methods for latency-sensitive deployment. These findings highlight the trade-off between effectiveness and computational cost in Text-based PRF for rerankers.

Chapter 4 reproduces the ANCE-PRF technique [234] to assess the integration of learned PRF in dense retrieval architectures. ANCE-PRF is trained on concatenated query-passage pairs, allowing the dense encoder to embed feedback information directly (Figure 4.1). Although effective under the original training configuration, we identified several limitations: the model exhibits high sensitivity to input pre-processing, demonstrates poor generalisability across retrievers, and shows strong dependency on the PRF depth used during training. Particularly, applying the encoder at a different depth during inference significantly degrades performance (Section 4.3). These limitations reveal that dense retrievers trained with feedback signals are fragile and difficult to adapt, motivating the development of more flexible, retraining-free alternatives.

Chapter 5 introduces Vector-Based PRF (VPRF) as a simple, efficient, and model-agnostic vector manipulation approach, alternative to the learned feedback methods (Figure 5.1). VPRF operates entirely in the dense embedding space by aggregating the representations of top-ranked feedback passages and interpolating them with the original query vector. This approach eliminates the need for retraining, token-level expansion, or additional inference steps. Evaluations across multiple dense retrievers, including RepBERT [238] and ANCE [223], on both TREC DL benchmarks demonstrate that VPRF offers consistent improvements in MAP and nDCG@10 (Section 5.4). Additionally, VPRF is fully implemented within Pyserini [112], making it reproducible and easily integrable into dense retrieval pipelines. These results demonstrate that effective feedback can be achieved through lightweight representational modifications without altering retriever architectures.

Chapter 6 extends the investigation by introducing hybrid sparse-dense interpolation strategies within the PRF process. We examine three configurations, Pre-PRF, Post-PRF, and Both-PRF, to combine dense feedback with sparse signals from retrievers such as BM25 and uniCOIL [111] (Figure 6.1). Our findings show that Post-PRF and Both-PRF consistently outperform dense-only configurations, confirming that interpolating dense feedback with sparse scores after the feedback application enhances robustness and effectiveness. In contrast, Pre-PRF introduces greater variability in feedback document selection, leading to less stable performance (Section 6.3). These results highlight the importance of feedback entry points and signal combination timing when designing hybrid PRF systems.

Taken together, the contributions of Part 1 demonstrate multiple viable paths for integrating PRF into neural information retrieval systems. Text-based augmentation strategies, Vector-space feedback mechanisms, and sparse-dense hybrid approaches each offer complementary strengths. Our findings reveal key trade-offs between effectiveness, robustness, latency, and extensibility, establishing a methodological foundation for the subsequent parts of the thesis. Part 2 builds on these insights by addressing the challenges of reliability, efficiency, and real-world deployability in PRF design.

13.2 Designing Practical and Extensible Pseudo-Relevance Feedback Methods for Neural Information Retrieval

In Part 2, we addressed **RQ2: How to make Pseudo-Relevance Feedback models more practical and extensible?** While Part 1 focused on integrating PRF into neural retrievers, the methods developed there still face critical challenges related to computational efficiency, feedback reliability, and architectural coupling. This part investigates how PRF methods can be designed and implemented to be lightweight, robust, and deployable under realistic constraints, where latency, memory usage, and initial retrieval quality vary widely across tasks and systems.

Chapter 7 critically evaluates the reliability of PRF signals within the dense retrieval paradigm. While selective query expansion has long challenged the assumption that top-ranked documents are inherently relevant in retrieval, this premise remains largely implicit in modern neural PRF models. By applying a controlled diagnostic framework to dense architectures, we quantify how different feedback mechanisms respond to signal degradation, ranging from idealized ground-truth judgments to noisy, random top-ranked passages (Section 7.1). Our comparative analysis reveals a distinct divergence in robustness: while strong signals universally boost performance, learned models like ANCE-PRF exhibit extreme sensitivity to noise, degrading sharply when the feedback is unreliable. In contrast, the non-parametric VPRF-Rocchio demonstrates greater stability under degraded conditions (Section 7.3). These findings provide new empirical evidence that learned feedback encoders, despite their high potential capacity, lack the inherent robustness of simpler vector-based operations, highlighting the necessity for explicit noise-mitigation strategies in neural feedback training.

In Chapter 8, we introduce Transformer-based Pseudo-Relevance Feedback (TPRF) as a lightweight, adaptable feedback encoder designed to reduce the resource demands of PRF without sacrificing performance. Unlike ANCE-PRF, which requires pretraining on concatenated query-passage pairs, TPRF learns to encode the set of PRF dense passage vectors into an enhanced query representation (Figure 8.1). It operates entirely in the vector space, avoiding the complexities of token-level manipulation, and allows for variable model sizes through architecture tuning. We conduct extensive evaluations across multiple retrievers and benchmark datasets. TPRF outperforms VPRF-Average and even surpasses ANCE-PRF on early precision metrics such as RR and nDCG@1 (Section 8.3). Importantly, TPRF introduces only minimal inference latency, typically sub-20ms per query, and maintains efficiency even when deployed on CPUs (Section 8.3.2). These findings challenge the assumption that transformer-based feedback must be computationally intensive, showing that careful design enables models that are both effective and practical for deployment.

In Chapter 9, we shifted focus from algorithmic innovation to infrastructure and standardization, proposing a modular architecture for dense feedback (Figure 9.1). While implemented within the Pyserini [112] toolkit, this framework serves as a generalized blueprint designed to support the full spectrum of methods explored in this thesis, from the static vector operations of VPRF to the learned adaptivity of TPRF and the generative logic of PromptPRF (Chapter 11). Using VPRF-Average and VPRF-Rocchio as foundational proof-of-concept integrations, we demonstrated a plug-and-play design

where feedback mechanisms are decoupled from the underlying retriever. Our extensive evaluation across eight dense retrievers confirmed that this infrastructure delivers consistent effectiveness gains across TREC DL and BEIR benchmarks with negligible latency overhead (Section 9.3.2). The validity of this standardized approach is further evidenced by the parallel adoption of these strategies in other major toolkits, such as PyTerrier [136]. Therefore, this framework provides the necessary engineering groundwork to transition advanced neural feedback from research prototypes to reproducible, production-ready components.

Taken together, the contributions of Part 2 show that PRF can be redesigned for practical use in modern retrieval systems. We demonstrate that signal quality directly impacts performance reliability, and that feedback models must be tested under both ideal and adverse conditions to gauge robustness. TPRF shows that strong performance does not require heavy-weight models, and that compact feedback encoders can support low-latency and resource constraint applications. Finally, our extensible PRF framework ensures that these methods can be widely adopted and compared, facilitating ongoing research and real-world deployment.

These insights form a critical bridge between integration and application. While Part 1 established how PRF can be incorporated into neural retrievers, Part 2 address how to make those methods deployable, generalisable, and scalable in practice. This sets the stage for Part 3, where we turn our attention to an emerging frontier: the application of PRF in the context of decoder-style large language models.

13.3 Reassessing Pseudo-Relevance Feedback in the Era of Decoder-Style Large Language Models

In Part 3, we addressed **RQ3: Are Pseudo-Relevance Feedback models still applicable to decoder-style Large Language Models (LLMs)?** As retrieval technology advances toward decoder-based architectures that demonstrate stronger retrieval performance, it becomes necessary to re-examine the applicability of the PRF methods. This part investigates whether these PRF methods can remain effective within LLM-based retrieval settings and, if so, how they must be adapted to align with the representational properties and computational constraints of decoder-style retrievers.

Chapter 10 investigates LLM-VPRF, an extension of previously proposed VPRF approach to dense retrievers built from decoder-style LLMs. These retrievers, including PromptReps [250], RepLLaMA [128], and LLM2Vec [13], generate dense query and passage representations often without contrastive learning. LLM-VPRF modifies the query embedding by aggregating top-ranked passage vectors using average pooling or Rocchio-style interpolation. Importantly, this feedback process is implemented without retriever-specific tuning, preserving model-agnostic nature. Our evaluation on BEIR and TREC DL benchmarks shows that LLM-VPRF consistently improves nDCG@10 and Recall@100 across all three retriever variants (Section 10.3). In particular, models like RepLLaMA benefit from stable feedback integration, while PromptReps and LLM2Vec exhibit higher variance due

to sensitivity to the top-k passages. Importantly, LLM-VPRF adds no additional latency or architectural complexity, confirming that Vector-based PRF remains effective and scalable in decoder-style dense retrieval.

Chapter 11 proposes PromptPRF, a prompt-driven feature-based PRF method tailored to zero-shot LLM-based retrievers. Rather than altering embeddings or retriever training, PromptPRF incorporates structured features, such as keywords, entities, etc., generated from top-k feedback passages into the query encoding prompts (Figure 2.7). These features are produced offline using generative LLMs and embedded into template-based prompts at inference time. PromptPRF is evaluated across various LLM retrievers and model sizes. Our experiments show that PromptPRF consistently improves retrieval effectiveness (Section 11.4.1). Particularly, applying PromptPRF to small models narrows or even closes the performance gap with much larger models on several datasets. Furthermore, PromptPRF significantly outperforms a strong passage concatenation baseline, indicating that structured feedback features provide a more semantically aligned and efficient retrieval signal than raw passage text. These results suggest that generative feedback, when constructed carefully, can serve as an effective mechanism for enhancing LLM retrieval without requiring retriever modification or brutal-force scaling up the underlying retriever.

Together, Chapters 10 and 11 demonstrate that PRF continues to offer meaningful benefits in LLM-based retrieval environments, though its mechanisms must be adapted to suit the nature of decoder-style architectures. Vector-based feedback can still be applied through dense representations generated by autoregressive models, while prompt-based techniques enable lightweight, interpretable, and retriever-agnostic improvements, especially in zero-shot and small-model scenarios. These findings reaffirm the relevance of PRF in the context of modern LLM retrieval and set the stage for further exploration of dynamic, and adaptive feedback strategies.

13.4 Investigating Failure Modes

Finally, Chapter 12 complements the quantitative evaluations presented throughout the thesis with a diagnostic investigation into the practical limitations and failure modes of VPRF (Chapter 5), TPRF (Chapter 8), and PromptPRF (Chapter 11). Through targeted case studies across these methods, we identified that retrieval failures are rarely random, but rather stem from specific vulnerabilities in how neural models process noisy or ambiguous signals. The analysis revealed that while these architectures successfully transcend lexical mismatch, they still exhibit classic feedback challenges in distinct, architecture-specific ways, such as latent semantic drift in vector embeddings and instructional misalignment in generative models. These insights highlight that architectural advancement alone does not guarantee robustness; the effectiveness of the more advanced neural PRF methods remains tightly coupled to the quality of the initial feedback, emphasizing the continued need for signal-aware adaptation mechanisms.

13.5 Key Takeaways

This thesis presents a comprehensive study on the adaptation of PRF to the modern neural retrieval landscape, encompassing transformer-based rerankers, encoder-style dense retrievers, and decoder-style LLM-based architectures. Through the lens of three high-level research questions, we explored the integration, practicality, and applicability of PRF, uncovering several overarching insights that define the contributions of this work.

A central takeaway is that PRF remains a fundamental component of the retrieval pipeline, having successfully evolved from lexical expansion to latent representation refinement. We demonstrated that the classical notion of leveraging top-ranked passages to enhance query representations continues to offer meaningful gains when reinterpreted in the dense vector space. Our methods consistently demonstrate that feedback mechanisms retain immense value in neural IR, provided they are appropriately adapted to the underlying representation logic of the model.

Throughout the thesis, we established that lightweight, representation-level feedback approaches offer a scalable and retraining-free path to integration. The VPRF framework demonstrated that simple geometric operations over dense embeddings can be both effective and efficient. By avoiding the complexity of query rewriting or expensive model retraining, these methods can be seamlessly plugged into existing dense retrieval pipelines with minimal computational overhead, democratizing access to feedback mechanisms.

Our diagnostic analysis further revealed that the quality of the feedback signal is a critical determinant of success, often acting as a bottleneck also for neural methods. Contrary to the assumption that deep learning models automatically handle noise, our controlled experiments showed that learned feedback encoders (e.g., ANCE-PRF) can exhibit extreme sensitivity to signal degradation, degrading sharply when feedback is unreliable. In contrast, non-parametric methods like VPRF-Rocchio demonstrated greater stability under noisy conditions. This suggests a clear trade-off: while learned models offer higher potential capacity, they lack the inherent robustness of simpler vector operations when the pseudo-relevance assumption is violated.

Efficiency and deployability emerged as core constraints addressed by our architectural innovations. With TPRF, we showed that the trade-off between adaptability and efficiency is not zero-sum; a lightweight transformer adapter can learn dynamic weighting strategies without the prohibitive cost of full backbone fine-tuning. Similarly, PromptPRF demonstrated that the reasoning power of Generative LLMs can be harnessed efficiently by decoupling feature generation (offline) from query encoding (online). This architecture allows retrieval systems to benefit from generative feedback signals without incurring the high latency typically associated with large language models.

Furthermore, we found that hybrid sparse-dense feedback models significantly enhance robustness. Our investigation into interpolation strategies, Pre-PRF, Post-PRF, and Both-PRF, confirmed that sparse retrieval signals remain a vital complement to semantic embeddings. Integrating lexical signals at both the feedback selection and final ranking stages (Both-PRF) yields the most stable performance,

preventing semantic drift and ensuring that abstract semantic matching is robustly reinforced by precise lexical evidence.

Finally, this thesis highlights the necessity of standardized infrastructure for feedback research. By implementing a modular PRF framework within the Pyserini toolkit, we provided a generalized blueprint for integrating diverse feedback strategies. This infrastructure not only facilitates reproducibility but also establishes the engineering groundwork required to transition advanced methods, from VPRF to TPRF, and from research prototypes to production-ready components.

In summary, the findings in this thesis reaffirm the vitality of PRF in the era of neural search. While retrieval architectures have grown increasingly complex, the fundamental principle of feedback remains powerful. Our contributions offer both the theoretical insights and the practical tools necessary to advance the next generation of feedback-aware retrieval systems.

13.6 Future Directions

This thesis has established that Pseudo-Relevance Feedback (PRF), a technique rooted in classical sparse retrieval, remains a vital component in the era of neural and LLM-based retrieval. However, our investigation across vector-based (VPRF), transformer-based (TPRF), and prompt-based (PromptPRF) methods has revealed specific boundary conditions where these methods falter. The transition from lexical to latent representations solves the vocabulary mismatch problem but still exhibits challenges regarding semantic drift and signal noise.

Reflecting on the empirical findings presented in Chapters 8 through 12, we identify four primary avenues for future research. These directions are not merely speculative but are direct extensions of the unresolved limitations identified in this work.

13.6.1 Revisiting the *Negative* Component of Rocchio in Dense Spaces

In Chapter 5, we introduced Vector-Based PRF (VPRF), which manipulates query embeddings using a simplified Rocchio algorithm. Our formulation focused exclusively on the *positive* expansion component (β), pulling the query vector toward the centroid of the top-k pseudo-relevant passages. While this yielded consistent gains in Recall@1000, our failure analysis in Chapter 12 revealed that VPRF is susceptible to query drift when the top-k passages share a dominant but non-relevant semantic theme (e.g., retrieving "Uno card game" rules for a query about "UN FAO").

Classical sparse PRF literature suggests that the full Rocchio algorithm includes a *negative* feedback component ($-\gamma$) to push the query away from non-relevant documents [179]. In the sparse domain, identifying pseudo-negatives is difficult due to the scarcity of overlapping terms. However, dense retrieval spaces are continuous, making negative feedback potentially more actionable. Future work should investigate Vector-Based Negative Feedback, where the query embedding is explicitly pushed away from the bottom-ranked passages of the initial retrieval list (acting as soft negatives) or from varying hard negative definitions. By re-incorporating the negative component of the classical Rocchio

formalism into VPRF, we may mathematically constrain the query vector, preventing the semantic drift observed in our case studies without requiring the computational overhead of a trained encoder.

13.6.2 Adaptive Feature Selection for Prompt-Based Feedback

Our evaluation of PromptPRF in Chapter 11 demonstrated that offline feature extraction (e.g., generating `Keywords` or `Facts`) significantly improves efficiency compared to online generation. However, a key limitation identified in our results was the rigid application of feature types. We observed in Section 11.4 that structured features like `Entities` performed best for factoid queries (e.g., "reba mcentire net worth"), while broader features like `Summary` or `Essay` were required for complex, exploratory topics. Currently, PromptPRF applies a single feature type uniformly across all queries, which is a suboptimal one-size-fits-all approach.

Future research should focus on Query-Adaptive Feature Selection. Drawing on the concept of Selective Query Expansion (SQE) from sparse retrieval, where expansion is only applied when query performance is predicted to be low, LLM-based feedback should dynamically select which features to inject based on the query intent. This could be achieved via a lightweight classifier or by prompting the LLM to self-select the most appropriate feedback type (e.g., "Does this query require factual entities or a thematic summary?"). By moving from static feature injection to dynamic, per-query feature selection, PromptPRF can better mitigate the noise introduced by verbose features in simple queries while retaining the richness required for complex tasks.

13.6.3 Enhancing the Reasoning Capabilities of Lightweight Adapters

In Chapter 8, we proposed TPRF, a lightweight transformer adapter that learns to weight feedback signals. Our results showed that TPRF is exceptionally efficient and scales gracefully to high PRF depths ($k = 100$). However, while it outperforms static vector averaging, it still treats feedback integration primarily as a relevance accumulation task. It essentially learns *which passages are relevant*, but it does not explicitly model *what information is missing*.

The findings in Chapter 6 regarding the success of sparse-dense interpolation suggest that lexical precision is often the missing piece in dense feedback loops. Future work should explore Hybridized TPRF Architectures that attend not only to dense passage vectors but also to sparse lexical signals or explicit diversity markers. Since TPRF is computationally cheap, it offers an ideal testbed for experimenting with more complex attention mechanisms that enforce diversity (Maximal Marginal Relevance-style) within the feedback loop, ensuring that the expanded query representation covers multiple aspects of the user's intent rather than simply reinforcing the dominant semantic cluster of the top-k results.

13.6.4 Closing the Loop: Iterative and Agentic Feedback

Finally, the failure cases analyzed in Chapter 12 highlight a fundamental limitation of the *retrieve-then-expand* paradigm: it is a single-pass process. If the initial retrieval set is poor (as seen in the DL HARD dataset), PRF inevitably reinforces noise. Generative LLMs offer a unique opportunity to break this dependency through Iterative Agentic Feedback.

Rather than treating PRF as a static augmentation step, future systems should model feedback as a multi-turn reasoning process. Using the efficiency gains from PromptPRF's offline indexing, an LLM agent could inspect the initial retrieval set, detect potential drift (e.g., "these results discuss a card game, but the query implies an organization"), and iteratively refine the prompt or generate negative constraints before performing a second retrieval. This aligns PRF with emerging trends in agentic workflows, evolving Pseudo-Relevance Feedback from a statistical approximation of relevance into an active, reasoning-driven search refinement process.

In conclusion, this thesis bridges the gap between classical feedback principles and modern neural architectures. By addressing the outstanding issues of signal noise, static feature selection, and single-pass fragility, future research can develop feedback mechanisms that are not only more accurate but also more intelligent and adaptable to the user's varied information needs.

Bibliography

[1] Nasreen Abdul-Jaleel, James Allan, W. Bruce Croft, Fernando Diaz, Leah S. Larkey, Xiaoyan Li, Mark D. Smucker, and Courtney Wade. Umass at TREC 2004: Novelty and HARD. In *Proceedings of the 13th Text REtrieval Conference, TREC 2004, Gaithersburg, Maryland, USA, November 16-19, 2004*, volume 500-261 of *NIST Special Publication*. National Institute of Standards and Technology (NIST), 2004.

[2] Giambattista Amati. *Probability models for information retrieval based on divergence from randomness*. PhD thesis, University of Glasgow, 2003.

[3] Giambattista Amati. Frequentist and bayesian approach to information retrieval. In *Proceedings of the 28th European conference on information retrieval (ECIR)*, pages 13–24. Springer, 2006.

[4] Gianni Amati and Cornelis Joost Van Rijsbergen. Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Transactions on Information Systems (TOIS)*, 20(4):357–389, 2002.

[5] Mayank Anand, Jiarui Zhang, Shane Ding, Ji Xin, and Jimmy Lin. Serverless BM25 search and BERT reranking. In *Proceedings of the Second International Conference on Design of Experimental Search & Information REtrieval Systems, Padova, Italy, September 15-18, 2021*, volume 2950 of *CEUR Workshop Proceedings*, pages 3–9. CEUR-WS.org, 2021.

[6] Peter G. Anick. Using terminological feedback for web search refinement: a log-based study. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, July 28 - August 1, 2003, Toronto, Canada*, pages 88–95. ACM, 2003.

[7] Negar Arabzadeh, Chuan Meng, Mohammad Aliannejadi, and Ebrahim Bagheri. Query performance prediction: techniques and applications in modern information retrieval. In *Proceedings of the 2024 Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*, pages 291–294, 2024.

[8] Javed A. Aslam and Mark H. Montague. Models for metasearch. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, September 9-13, 2001, New Orleans, Louisiana, USA*, pages 275–284. ACM, 2001.

[9] Hiteshwar Kumar Azad and Akshay Deepak. Query expansion techniques for information retrieval: A survey. *Information Processing and Management*, 56(5):1698–1735, 2019.

[10] Hiteshwar Kumar Azad and Akshay Deepak. Query expansion techniques for information retrieval: A survey. *Information Processing and Management*, 56(5):1698–1735, 2019.

[11] Soyuj Basnet, Jerry Gou, Antonio Mallia, and Torsten Suel. Deeperimpact: Optimizing sparse learned index structures. *arXiv preprint arXiv:2405.17093*, 2024.

[12] Elias Bassani, Nicola Tonellotto, and Gabriella Pasi. Personalized query expansion with contextual word embeddings. *ACM Transactions on Information Systems*, 42(2):1–35, 2023.

[13] Parishad BehnamGhader, Vaibhav Adlakha, Marius Mosbach, Dzmitry Bahdanau, Nicolas Chapados, and Siva Reddy. LLM2Vec: Large language models are secretly powerful text encoders. In *Proceedings of 1st Conference on Language Modeling*, 2024.

[14] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.

[15] Chris Buckley, Gerard Salton, James Allan, and Amit Singhal. Automatic query expansion using SMART: TREC 3. In *Proceedings of The 3rd Text REtrieval Conference, TREC 1994, Gaithersburg, Maryland, USA, November 2-4, 1994*, volume 500-225 of *NIST Special Publication*, pages 69–80. National Institute of Standards and Technology (NIST), 1994.

[16] Guihong Cao, Jian-Yun Nie, Jianfeng Gao, and Stephen Robertson. Selecting good expansion terms for pseudo-relevance feedback. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2008, Singapore, July 20-24, 2008*, pages 243–250. ACM, 2008.

[17] David Carmel and Elad Yom-Tov. *Estimating the Query Difficulty for Information Retrieval*. Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan & Claypool Publishers, 2010. ISBN 978-3-031-01144-3.

[18] Claudio Carpineto and Giovanni Romano. A survey of automatic query expansion in information retrieval. *ACM Computing Surveys (CSUR)*, 44(1):1:1–1:50, 2012.

[19] Youjin Chang, Iadh Ounis, and Minkoo Kim. Query reformulation using automatically generated query concepts from a document space. *Information processing & management*, 42(2):453–468, 2006.

[20] Xiaoyang Chen, Kai Hui, Ben He, Xianpei Han, Le Sun, and Zheng Ye. Incorporating ranking context for end-to-end bert re-ranking. In *Proceedings of the 2022 European Conference on Information Retrieval*, pages 111–127. Springer, 2022.

[21] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734. ACL, 2014.

[22] Jooyoung Choi, Hyun Kim, Hansol Jang, Changwook Jun, Kyunghoon Bae, Hyewon Choi, Stanley Jungkyu Choi, Honglak Lee, and Chulmin Yun. Lg-anna-embedding technical report. *arXiv preprint arXiv:2506.07438*, 2025.

[23] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Y. Zhao, Yanping Huang, Andrew M. Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25:70:1–70:53, 2024.

[24] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. ELECTRA: pre-training text encoders as discriminators rather than generators. In *Proceedings of the 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[25] Stéphane Clinchant and Éric Gaussier. A theoretical analysis of pseudo-relevance feedback models. In *Proceedings of the International Conference on the Theory of Information Retrieval, ICTIR '13, Copenhagen, Denmark, September 29 - October 02, 2013*, page 6. ACM, 2013.

[26] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. Overview of the TREC 2020 deep learning track. In *Proceedings of the 29th Text REtrieval Conference, TREC 2020, Virtual Event [Gaithersburg, Maryland, USA], November 16-20, 2020*, volume 1266 of *NIST Special Publication*. National Institute of Standards and Technology (NIST), 2020.

[27] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. Overview of the TREC 2019 deep learning track. *arXiv preprint arXiv:2003.07820*, 2020.

[28] Steve Cronen-Townsend, Yun Zhou, and W Bruce Croft. A framework for selective query expansion. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 236–237, 2004.

[29] Sebastian Cross, Hang Li, Arvin Zhuang, Ahmed Mourad, Guido Zuccon, and Bevan Koopman. IELAB for TREC conversational assistance track (cast) 2020. In *Proceedings of the Twenty-Ninth Text REtrieval Conference, TREC 2020, Virtual Event [Gaithersburg, Maryland, USA], November 16-20, 2020*, volume 1266 of *NIST Special Publication*. National Institute of Standards and Technology (NIST), 2020.

[30] J. Shane Culpepper, Fernando Diaz, and Mark D. Smucker. Research frontiers in information retrieval: Report from the third strategic workshop on information retrieval in lorne (swirl 2018). *SIGIR Forum*, 52(1):34–90, August 2018. ISSN 0163-5840.

[31] Zhuyun Dai and Jamie Callan. Deeper text understanding for IR with contextual neural language modeling. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*, pages 985–988. ACM, 2019.

[32] Zhuyun Dai and Jamie Callan. Context-aware term weighting for first stage passage retrieval. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1533–1536, 2020.

[33] Zhuyun Dai and Jamie Callan. Context-aware document term weighting for ad-hoc search. In *Proceedings of The Web Conference 2020*, pages 1897–1907, 2020.

[34] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc Viet Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 2978–2988. Association for Computational Linguistics, 2019.

[35] Jeffrey Dalton, Chenyan Xiong, and Jamie Callan. TREC cast 2019: The conversational assistance track overview. *arXiv preprint arXiv:2003.13624*, 2020.

[36] Suchana Datta, Debasis Ganguly, Sean MacAvaney, and Derek Greene. A deep learning approach for selective relevance feedback. In *European Conference on Information Retrieval*, pages 189–204. Springer, 2024.

[37] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019.

[38] Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Rühle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. Hybrid LLM: cost-efficient and

quality-aware query routing. In *Proceedings of the 12th International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

[39] Ming Ding, Chang Zhou, Hongxia Yang, and Jie Tang. Cogltx: Applying BERT to long texts. In *Proceedings of the Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[40] Cícero Nogueira dos Santos, Xiaofei Ma, Ramesh Nallapati, Zhiheng Huang, and Bing Xiang. Beyond [CLS] through ranking by generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 1722–1727. Association for Computational Linguistics, 2020.

[41] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library. *arXiv preprint arXiv:2401.08281*, 2024.

[42] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[43] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

[44] Hui Fang and ChengXiang Zhai. An exploration of axiomatic approaches to information retrieval. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Salvador, Brazil, August 15-19, 2005*, pages 480–487. ACM, 2005.

[45] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. SPLADE v2: Sparse lexical and expansion model for information retrieval. *arXiv preprint arXiv:2109.10086*, 2021.

[46] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. SPLADE: sparse lexical and expansion model for first stage ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, pages 2288–2292. ACM, 2021.

[47] George W. Furnas, Thomas K. Landauer, Louis M. Gomez, and Susan T. Dumais. The vocabulary problem in human-system communication. *Communications of the ACM*, 30(11): 964–971, 1987.

[48] Luyu Gao and Jamie Callan. Condenser: a pre-training architecture for dense retrieval. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 981–993. Association for Computational Linguistics, 2021.

[49] Luyu Gao and Jamie Callan. Long document re-ranking with modular re-ranker. In Enrique Amigó, Pablo Castells, Julio Gonzalo, Ben Carterette, J. Shane Culpepper, and Gabriella Kazai, editors, *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022*, pages 2371–2376. ACM, 2022.

[50] Luyu Gao and Jamie Callan. Unsupervised corpus aware language model pre-training for dense passage retrieval. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 2843–2853. Association for Computational Linguistics, 2022.

[51] Luyu Gao, Zhuyun Dai, and Jamie Callan. COIL: revisit exact lexical match in information retrieval with contextualized inverted list. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 3030–3042. Association for Computational Linguistics, 2021.

[52] Luyu Gao, Zhuyun Dai, and Jamie Callan. Rethink training of BERT rerankers in multi-stage retrieval pipeline. In *Proceedings of the 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28 - April 1, 2021, Proceedings, Part II*, volume 12657 of *Lecture Notes in Computer Science*, pages 280–286. Springer, 2021.

[53] Luyu Gao, Zhuyun Dai, Tongfei Chen, Zhen Fan, Benjamin Van Durme, and Jamie Callan. Complement lexical retrieval model with semantic residual embeddings. In *Proceedings of the 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28 - April 1,*

2021, Proceedings, Part I, volume 12656 of *Lecture Notes in Computer Science*, pages 146–160. Springer, 2021.

[54] Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. Tevatron: An efficient and flexible toolkit for neural retrieval. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2023, Taipei, Taiwan, July 23-27, 2023*, pages 3120–3124. ACM, 2023.

[55] Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 6894–6910. Association for Computational Linguistics, 2021.

[56] Michael R. Glass, Gaetano Rossiello, Md. Faisal Mahbub Chowdhury, Ankita Naik, Pengshan Cai, and Alfio Gliozzo. Re2g: Retrieve, rerank, generate. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 2701–2715. Association for Computational Linguistics, 2022.

[57] Ozan Gökdemir, Carlo Siebenschuh, Alexander Brace, Azton Wells, Brian Hsu, Kyle Hippe, Priyanka Setty, Aswathy Ajith, J. Gregory Pauloski, Varuni Sastry, Sam Foreman, Huihuo Zheng, Heng Ma, Bharat Kale, Nicholas Chia, Thomas Gibbs, Michael E. Papka, Thomas S. Brettin, Francis J. Alexander, Anima Anandkumar, Ian T. Foster, Rick Stevens, Venkatram Vishwanath, and Arvind Ramanathan. Hiperrag: High-performance retrieval augmented generation for scientific insights. In *Proceedings of the Platform for Advanced Scientific Computing Conference, PASC 2025, FHNW University of Applied Sciences and Arts Northwestern Switzerland, Brugg-Windisch, Switzerland, June 16-18, 2025*, pages 1–13. ACM, 2025.

[58] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016*, pages 55–64. ACM, 2016.

[59] Jiafeng Guo, Yixing Fan, Liang Pang, Liu Yang, Qingyao Ai, Hamed Zamani, Chen Wu, W. Bruce Croft, and Xueqi Cheng. A deep look into neural ranking models for information retrieval. *Information Processing and Management*, 57(6):102067, 2020.

[60] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. Retrieval augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 3929–3938. PMLR, 2020.

[61] Kyoung-Soo Han. Dualized topic-preserving pseudo relevance feedback for question answering. *Transactions on Information Systems (TOIS)*, 100-D(7):1550–1553, 2017.

[62] Xu Han, Weize Chen, Zhiyuan Liu, Yankai Lin, and Maosong Sun. Knowledge representation learning and knowledge-guided nlp. *Representation Learning for Natural Language Processing*, page 273, 2023.

[63] Donna Harman. Relevance feedback revisited. In *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 1–10, 1992.

[64] Ben He and Iadh Ounis. Query performance prediction. *Information Systems*, 31(7):585–594, 2006.

[65] Ben He and Iadh Ounis. Combining fields for query expansion and adaptive query expansion. *Information processing & management*, 43(5):1294–1307, 2007.

[66] Ben He and Iadh Ounis. Finding good feedback documents. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 2011–2014, 2009.

[67] Ben He and Iadh Ounis. Studying query expansion effectiveness. In *European conference on information retrieval*, pages 611–619. Springer, 2009.

[68] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[69] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: decoding-enhanced bert with disentangled attention. In *Proceedings of the 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[70] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

[71] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pages 30016–30030, 2022.

[72] Sebastian Hofstätter, Sophia Althammer, Michael Schröder, Mete Sertkan, and Allan Hanbury. Improving efficient neural ranking models with cross-architecture knowledge distillation. *arXiv preprint arXiv:2010.02666*, 2020.

[73] Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy Lin, and Allan Hanbury. Efficiently teaching an effective dense retriever with balanced topic aware sampling. In *Proceedings*

of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021, pages 113–122. ACM, 2021.

[74] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1 (2):3, 2022.

[75] Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. Unsupervised dense information retrieval with contrastive learning. *Transactions on Machine Learning Research*, 2022, 2022.

[76] Thomas Jaenich, Graham McDonald, and Iadh Ounis. Colbert-fairprf: towards fair pseudo-relevance feedback in dense retrieval. In *The Proceedings of the 2023 European Conference on Information Retrieval*, pages 457–465. Springer, 2023.

[77] Nour Jedidi, Yung-Sung Chuang, Leslie Shing, and James R. Glass. Zero-shot dense retrieval with embeddings from relevance feedback. *arXiv preprint arXiv:2410.21242*, 2024.

[78] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

[79] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547, 2021.

[80] Karen Spärck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21, 1972.

[81] Parul Kalra, Deepti Mehrotra, and Abdul Wahid. Query expansion using pseudo-relevance feedback on ad hoc. In *Proceedings of the International Conference on Emerging Trends in Artificial Intelligence, Data Science and Signal Processing*, pages 23–32. Springer, 2023.

[82] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[83] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 6769–6781. Association for Computational Linguistics, 2020.

[84] Mostafa Keikha, Jae Hyun Park, and W. Bruce Croft. Evaluating answer passages using summarization measures. In *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '14, Gold Coast , QLD, Australia - July 06 - 11, 2014*, pages 963–966. ACM, 2014.

[85] Omar Khattab and Matei Zaharia. Colbert: Efficient and effective passage search via contextualized late interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pages 39–48. ACM, 2020.

[86] Bevan Koopman, Ahmed Mourad, Hang Li, Anton van der Vegt, Shengyao Zhuang, Simon Gibson, Yash Dang, David Lawrence, and Guido Zuccon. Agask: an agent to help answer farmer's questions from scientific documents. *International Jorunal on Digital Libraries*, 25(4): 569–584, 2024.

[87] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[88] Robert Krovetz and W Bruce Croft. Lexical ambiguity and information retrieval. *ACM Transactions on Information Systems (TOIS)*, 10(2):115–141, 1992.

[89] Solomon Kullback and Richard A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[90] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. In *Proceedings of the 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[91] Victor Lavrenko and W. Bruce Croft. Relevance-based language models. In W. Bruce Croft, David J. Harper, Donald H. Kraft, and Justin Zobel, editors, *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, September 9-13, 2001, New Orleans, Louisiana, USA*, pages 120–127. ACM, 2001.

[92] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[93] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7871–7880. Association for Computational Linguistics, 2020.

[94] Canjia Li, Yingfei Sun, Ben He, Le Wang, Kai Hui, Andrew Yates, Le Sun, and Jungang Xu. NPRF: A neural pseudo relevance feedback framework for ad-hoc information retrieval. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 4482–4491. Association for Computational Linguistics, 2018.

[95] Chaofan Li, Zheng Liu, Shitao Xiao, Yingxia Shao, and Defu Lian. Llama2vec: Unsupervised adaptation of large language models for dense retrieval. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 3490–3500. Association for Computational Linguistics, 2024.

[96] Hang Li, Harrisen Scells, and Guido Zuccon. Systematic review automation tools for end-to-end query formulation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pages 2141–2144. ACM, 2020.

[97] Hang Li, Ahmed Mourad, Bevan Koopman, and Guido Zuccon. Agvaluate: A new test collection for both passage and document retrieval in the agriculture domain. *The University of Queensland Data Collection*, 2022. doi: 10.48610/0160dc7.

[98] Hang Li, Ahmed Mourad, Bevan Koopman, and Guido Zuccon. How does feedback signal quality impact effectiveness of pseudo relevance feedback for passage retrieval. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022*, pages 2154–2158. ACM, 2022.

[99] Hang Li, Ahmed Mourad, Bevan Koopman, and Guido Zuccon. How does feedback signal quality impact effectiveness of pseudo relevance feedback for passage retrieval. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022*, pages 2154–2158. ACM, 2022.

[100] Hang Li, Shuai Wang, Shengyao Zhuang, Ahmed Mourad, Xueguang Ma, Jimmy Lin, and Guido Zuccon. To interpolate or not to interpolate: Prf, dense and sparse retrievers. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022*, pages 2495–2500. ACM, 2022.

[101] Hang Li, Shengyao Zhuang, Xueguang Ma, Jimmy Lin, and Guido Zuccon. Pseudo-relevance feedback with dense retrievers in pyserini. In *Proceedings of the 26th Australasian Document Computing Symposium, ADCS 2022, Adelaide, SA, Australia, December 15-16, 2022*, pages 1:1–1:6. ACM, 2022.

[102] Hang Li, Shengyao Zhuang, Ahmed Mourad, Xueguang Ma, Jimmy Lin, and Guido Zuccon. Improving query representations for dense retrieval with pseudo relevance feedback: A reproducibility study. In *Proceedings of the 44th European Conference on IR Research, ECIR 2022,*

Stavanger, Norway, April 10-14, 2022, Proceedings, Part I, volume 13185 of *Lecture Notes in Computer Science*, pages 599–612. Springer, 2022.

[103] Hang Li, Bevan Koopman, Ahmed Mourad, and Guido Zuccon. Agask: A conversational search agent for answering agricultural questions. In *Proceedings of the 16th ACM International Conference on Web Search and Data Mining, WSDM 2023, Singapore, 27 February 2023 - 3 March 2023*, pages 1140–1143. ACM, 2023.

[104] Hang Li, Ahmed Mourad, Shengyao Zhuang, Bevan Koopman, and Guido Zuccon. Pseudo relevance feedback with deep language models and dense retrievers: Successes and pitfalls. *ACM Transactions on Information Systems (TOIS)*, 41(3):62:1–62:40, 2023.

[105] Hang Li, Chuting Yu, Ahmed Mourad, Bevan Koopman, and Guido Zuccon. TPRF: A transformer-based pseudo-relevance feedback model for efficient and effective retrieval. *arXiv preprint arXiv:2401.13509*, 2024.

[106] Hang Li, Chuting Yu, Xiao Wang, Bevan Koopman, and Guido Zuccon. Pseudo-relevance feedback can improve zero-shot llm-based dense retrieval. *arXiv preprint arXiv:2503.14887*, 2025.

[107] Hang Li, Shengyao Zhuang, Bevan Koopman, and Guido Zuccon. LLM-VPRF: large language model based vector pseudo relevance feedback. *arXiv preprint arXiv:2504.01448*, 2025.

[108] Jinhao Li, Haopeng Li, Sarah Monazam Erfani, Lei Feng, James Bailey, and Feng Liu. Visual-text cross alignment: Refining the similarity score in vision-language models. In *Proceedings of the 41st International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

[109] Zihan Liao, Hang Yu, Jianguo Li, Jun Wang, and Wei Zhang. D2LLM: decomposed and distilled large language models for semantic search. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 14798–14814. Association for Computational Linguistics, 2024.

[110] Jimmy Lin. The simplest thing that can possibly work: Pseudo-relevance feedback using text classification. *arXiv preprint arXiv:1904.08861*, 2019.

[111] Jimmy Lin and Xueguang Ma. A few brief notes on deepimpact, coil, and a conceptual framework for information retrieval techniques. *arXiv preprint arXiv:2106.14807*, 2021.

[112] Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. Pyserini: An easy-to-use python toolkit to support replicable IR research with sparse and dense representations. *arXiv preprint arXiv:2102.10073*, 2021.

[113] Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. *Pretrained Transformers for Text Ranking: BERT and Beyond*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2021. ISBN 978-3-031-01053-8.

[114] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. Distilling dense representations for ranking using tightly-coupled teachers. *arXiv preprint arXiv:2010.11386*, 2020.

[115] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. In-batch negatives for knowledge distillation with tightly-coupled teachers for dense retrieval. In *Proceedings of the 6th Workshop on Representation Learning for NLP, RepL4NLP@ACL-IJCNLP 2021, Online, August 6, 2021*, pages 163–173. Association for Computational Linguistics, 2021.

[116] Wei-Chao Lin. Block-based pseudo-relevance feedback for image retrieval. *Journal of Experimental and Theoretical Artificial Intelligence*, 34(5):891–903, 2022.

[117] Christina Lioma and Iadh Ounis. A syntactically-based query reformulation technique for information retrieval. *Information processing & management*, 44(1):143–162, 2008.

[118] Weijie Liu, Peng Zhou, Zhiruo Wang, Zhe Zhao, Haotang Deng, and Qi Ju. Fastbert: a self-distilling BERT with adaptive inference time. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 6035–6044. Association for Computational Linguistics, 2020.

[119] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[120] Yu-An Liu, Ruqing Zhang, Jiafeng Guo, Maarten de Rijke, Yixing Fan, and Xueqi Cheng. Robust neural information retrieval: An adversarial and out-of-distribution perspective. *arXiv preprint arXiv:2407.06992*, 2024.

[121] Yu-An Liu, Ruqing Zhang, Jiafeng Guo, and Maarten de Rijke. Robust information retrieval. In *Proceedings of the 18th ACM International Conference on Web Search and Data Mining, WSDM 2025, Hannover, Germany, March 10-14, 2025*, pages 1008–1011. ACM, 2025.

[122] Yuqi Liu. Simple yet effective pseudo relevance feedback with rocchio's technique and text classification. Master's thesis, University of Waterloo, 2022.

[123] Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. Sparse, dense, and attentional representations for text retrieval. *Transactions of the Association for Computational Linguistics*, 9:329–345, 2021.

[124] Hans Peter Luhn. Keyword-in-context index for technical literature (kwic index). *American Documentation*, 11(4):288–295, 1960.

[125] Yuanhua Lv and ChengXiang Zhai. A comparative study of methods for estimating query language models with pseudo feedback. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009, Hong Kong, China, November 2-6, 2009*, pages 1895–1898. ACM, 2009.

[126] Yuanhua Lv and ChengXiang Zhai. Positional relevance model for pseudo-relevance feedback. In *Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2010, Geneva, Switzerland, July 19-23, 2010*, pages 579–586. ACM, 2010.

[127] Yuanhua Lv and ChengXiang Zhai. When documents are very long, BM25 fails! In *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2011, Beijing, China, July 25-29, 2011*, pages 1103–1104. ACM, 2011.

[128] Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. Fine-tuning llama for multi-stage text retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, Washington DC, USA, July 14-18, 2024*, pages 2421–2425. ACM, 2024.

[129] Yanjun Ma, Dianhai Yu, Tian Wu, and Haifeng Wang. Paddlepaddle: An open-source deep learning platform from industrial practice. *Frontiers of Data and Domputing*, 1(1):105–115, 2019.

[130] Sean MacAvaney and Xi Wang. Online distillation for pseudo-relevance feedback. *arXiv preprint arXiv:2306.09657*, 2023.

[131] Sean MacAvaney, Andrew Yates, Arman Cohan, and Nazli Goharian. CEDR: contextualized embeddings for document ranking. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*, pages 1101–1104. ACM, 2019.

[132] Sean MacAvaney, Franco Maria Nardini, Raffaele Perego, Nicola Tonellotto, Nazli Goharian, and Ophir Frieder. Expansion via prediction of importance with contextualization. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 1573–1576, 2020.

[133] Craig Macdonald and Iadh Ounis. Expertise drift and query expansion in expert search. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 341–350, 2007.

[134] Craig Macdonald and Iadh Ounis. Using relevance feedback in expert search. In *European Conference on Information Retrieval*, pages 431–443. Springer, 2007.

[135] Craig Macdonald and Iadh Ounis. Voting techniques for expert search. *Knowledge and Information Systems*, 16(3):259–280, 2008.

[136] Craig Macdonald, Nicola Tonellotto, Sean MacAvaney, and Iadh Ounis. Pyterrier: Declarative experimentation in python from bm25 to dense retrieval. In *Proceedings of the 30th acm international conference on information & knowledge management*, pages 4526–4533, 2021.

[137] Iain Mackie, Jeffrey Dalton, and Andrew Yates. How deep is your learning: the DL-HARD annotated deep learning dataset. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, pages 2335–2341. ACM, 2021.

[138] Iain Mackie, Shubham Chatterjee, and Jeffrey Dalton. Generative and pseudo-relevant feedback for sparse, dense and learned sparse retrieval. *arXiv preprint arXiv:2305.07477*, 2023.

[139] Iain Mackie, Shubham Chatterjee, and Jeffrey Dalton. Generative relevance feedback with large language models. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2023, Taipei, Taiwan, July 23-27, 2023*, pages 2026–2031. ACM, 2023.

[140] Iain Mackie, Shubham Chatterjee, Sean MacAvaney, and Jeffrey Dalton. Adaptive latent entity expansion for document retrieval. *arXiv preprint arXiv:2306.17082*, 2023.

[141] Iain Mackie, Shubham Chatterjee, Sean MacAvaney, and Jeffrey Dalton. Adaptive latent entity expansion for document retrieval. *arXiv preprint arXiv:2306.17082*, 2023.

[142] Antonio Mallia, Omar Khattab, Torsten Suel, and Nicola Tonellotto. Learning passage impacts for inverted indexes. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1723–1727, 2021.

[143] Christopher D Manning. *An introduction to information retrieval*. Cambridge University Press, 2009.

[144] M. E. Maron. On indexing, retrieval and the meaning of about. *Journal of the American Society for Information Science*, 28(1):38–43, 1977.

[145] Donald Metzler, Yi Tay, Dara Bahri, and Marc Najork. Rethinking search: making domain experts out of dilettantes. In *ACM SIGIR Forum*, volume 55, pages 13:1–13:27, 2021.

[146] Jun Miao, Jimmy Xiangji Huang, and Zheng Ye. Proximity-based rocchio's model for pseudo relevance. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '12, Portland, OR, USA, August 12-16, 2012*, pages 535–544. ACM, 2012.

[147] Bhaskar Mitra and Nick Craswell. An introduction to neural information retrieval. *Foundations and Trends in Information Retrieval*, 13(1):1–126, 2018.

[148] Mirza Alim Mutasodirin and Radityo Eko Prasojo. Investigating text shortening strategy in bert: Truncation vs summarization. In *Proceedings of the 2021 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, pages 1–5. IEEE, 2021.

[149] Shahrzad Naseri, Jeff Dalton, Andrew Yates, and James Allan. CEQE: contextualized embeddings for query expansion. In *Proceedings of the 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28 - April 1, 2021, Proceedings, Part I*, volume 12656 of *Lecture Notes in Computer Science*, pages 467–482. Springer, 2021.

[150] Hien Nguyen. Capturing user intent for information retrieval. In *Proceedings of the 19th National Conference on Artificial Intelligence, 16th Conference on Innovative Applications of Artificial Intelligence, July 25-29, 2004, San Jose, California, USA*, pages 997–998. AAAI Press / The MIT Press, 2004.

[151] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. MS MARCO: A human generated machine reading comprehension dataset. In *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, December 9, 2016*, volume 1773 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2016.

[152] Jianmo Ni, Gustavo Hernández Ábrego, Noah Constant, Ji Ma, Keith B. Hall, Daniel Cer, and Yinfei Yang. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. In *Proceedings of the Findings of the Association for Computational Linguistics: ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 1864–1874. Association for Computational Linguistics, 2022.

[153] Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernández Ábrego, Ji Ma, Vincent Y. Zhao, Yi Luan, Keith B. Hall, Ming-Wei Chang, and Yinfei Yang. Large dual encoders are generalizable retrievers. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 9844–9855. Association for Computational Linguistics, 2022.

[154] Tong Niu, Shafiq Joty, Ye Liu, Caiming Xiong, Yingbo Zhou, and Semih Yavuz. Judgerank: Leveraging large language models for reasoning-intensive reranking. *arXiv preprint arXiv:2411.00142*, 2024.

[155] Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with BERT. *arXiv preprint arXiv:1901.04085*, 2019.

[156] Rodrigo Nogueira, Jimmy Lin, and AI Epistemic. From doc2query to doctttttquery. *Online preprint*, 6(2), 2019.

[157] Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. Multi-stage document ranking with BERT. *arXiv preprint arXiv:1910.14424*, 2019.

[158] Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. Document expansion by query prediction. *arXiv preprint arXiv:1904.08375*, 2019.

[159] Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. Document ranking with a pretrained sequence-to-sequence model. In *Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 708–718. Association for Computational Linguistics, 2020.

[160] Baharan Nouriinanloo and Maxime Lamothe. Re-ranking step by step: Investigating pre-filtering for re-ranking with large language models. *arXiv preprint arXiv:2406.18740*, 2024.

[161] OpenAI. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[162] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Proceedings of the Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[163] Ramith Padaki, Zhuyun Dai, and Jamie Callan. Rethinking query expansion for BERT reranking. In *Proceedings of the 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14-17, 2020, Proceedings, Part II*, volume 12036 of *Lecture Notes in Computer Science*, pages 297–304. Springer, 2020.

[164] Min Pan, Yu Liu, Jinguang Chen, Ellen Anne Huang, and Jimmy X Huang. A multi-dimensional semantic pseudo-relevance feedback framework for information retrieval. *Scientific Reports*, 14 (1):31806, 2024.

[165] Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 24-28 1998, Melbourne, Australia*, pages 275–281. ACM, 1998.

[166] Ronak Pradeep, Rodrigo Nogueira, and Jimmy Lin. The expando-mono-duo design pattern for text ranking with pretrained sequence-to-sequence models. *arXiv preprint arXiv:2101.05667*, 2021.

[167] Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. Rankvicuna: Zero-shot listwise document reranking with open-source large language models. *arXiv preprint arXiv:2309.15088*, 2023.

[168] Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. Rocketqa: An optimized training approach to dense passage retrieval for open-domain question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 5835–5847. Association for Computational Linguistics, 2021.

[169] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *OpenAI Blog*, 1(8), 2018.

[170] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. In *OpenAI Blog*, 2019.

[171] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:140:1–140:67, 2020.

[172] Revanth Gangi Reddy, Pradeep Dasigi, Md. Arafat Sultan, Arman Cohan, Avirup Sil, Heng Ji, and Hannaneh Hajishirzi. A large-scale study of reranker relevance feedback at inference. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2025, Padua, Italy, July 13-18, 2025*, pages 3010–3014. ACM, 2025.

[173] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 3980–3990. Association for Computational Linguistics, 2019.

[174] Ruiyang Ren, Shangwen Lv, Yingqi Qu, Jing Liu, Wayne Xin Zhao, Qiaoqiao She, Hua Wu, Haifeng Wang, and Ji-Rong Wen. PAIR: leveraging passage-centric similarity relation for improving dense passage retrieval. In *Proceedings of the Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 2173–2183. Association for Computational Linguistics, 2021.

[175] Ruiyang Ren, Yingqi Qu, Jing Liu, Wayne Xin Zhao, Qiaoqiao She, Hua Wu, Haifeng Wang, and Ji-Rong Wen. Rocketqav2: A joint training method for dense passage retrieval and passage re-ranking. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language*

Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021, pages 2825–2835. Association for Computational Linguistics, 2021.

[176] Stephen E. Robertson and Karen Spärck Jones. Relevance weighting of search terms. *Journal of the American Society for Information science*, 27(3):129–146, 1976.

[177] Stephen E. Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389, 2009.

[178] Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. Okapi at TREC-3. In *Proceedings of The 3rd Text REtrieval Conference, TREC 1994, Gaithersburg, Maryland, USA, November 2-4, 1994*, volume 500-225 of *NIST Special Publication*, pages 109–126. National Institute of Standards and Technology (NIST), 1994.

[179] J.J. Rocchio. Relevance feedback in information retrieval. In *The SMART Retrieval System - Experiments in Automatic Document Processing*, pages 313–323, 1971.

[180] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in bertology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8: 842–866, 2020.

[181] Tetsuya Sakai. On fuhr's guideline for ir evaluation. *SIGIR Forum*, 54(1), February 2021.

[182] Gerard Salton and Chris Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[183] Gerard Salton and Chris Buckley. Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science (JASIS)*, 41(4):288–297, 1990.

[184] Gerard Salton and Michael McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill Book Company, 1984. ISBN 0-07-054484-0.

[185] Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM (CACM)*, 18(11):613–620, 1975.

[186] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[187] Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. Colbertv2: Effective and efficient retrieval via lightweight late interaction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 3715–3734. Association for Computational Linguistics, 2022.

[188] Rodrygo LT Santos, Craig Macdonald, and Iadh Ounis. Exploiting query reformulations for web search result diversification. In *Proceedings of the 19th international conference on World wide web*, pages 881–890, 2010.

[189] Rodrygo LT Santos, Craig Macdonald, Iadh Ounis, et al. Search result diversification. *Foundations and Trends® in Information Retrieval*, 9(1):1–90, 2015.

[190] Harrisen Scells, Shengyao Zhuang, and Guido Zuccon. Reduce, reuse, recycle: Green information retrieval research. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022*, pages 2825–2837. ACM, 2022.

[191] Falk Scholer, Hugh E. Williams, John Yiannis, and Justin Zobel. Compression of inverted indexes for fast query evaluation. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 11-15, 2002, Tampere, Finland*, pages 222–229. ACM, 2002.

[192] Adi Simhi, Itay Itzhak, Fazl Barez, Gabriel Stanovsky, and Yonatan Belinkov. Trust me, i'm wrong: Llms hallucinate with certainty despite knowing the answer. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 14665–14688, 2025.

[193] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mpnet: Masked and permuted pre-training for language understanding. In *Proceedings of the Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[194] Yu Sun, Shuohuan Wang, Yu-Kun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. ERNIE 2.0: A continual pre-training framework for language understanding. In *Proceedings of The 34th AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8968–8975. AAAI Press, 2020.

[195] RS Sutton and AG Barto. Reinforcement learning: An introduction. *IEEE Transactions on Neural Networks*, 9(5):1054–1054, 1998.

[196] Tao Tao and ChengXiang Zhai. Regularized estimation of mixture models for robust pseudo-relevance feedback. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Seattle, Washington, USA, August 6-11, 2006*, pages 162–169. ACM, 2006.

[197] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *arXiv preprint arXiv:2009.06732*, 2020.

[198] Gemma Team. Gemma 3, 2025. URL `https://goo.gle/Gemma3Report`.

[199] Qwen Team. Qwen2.5: A party of foundation models, September 2024. URL `https://qwenlm.github.io/blog/qwen2.5/`.

[200] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021.

[201] Nicola Tonellotto, Craig Macdonald, and Iadh Ounis. Efficient and effective retrieval using selective pruning. In *Proceedings of the 6th ACM International Conference on Web Search and Data Mining, WSDM 2013, Rome, Italy, February 4-8, 2013*, pages 63–72. ACM, 2013.

[202] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. doi: 10.48550/ARXIV.2302.13971.

[203] Johanne R. Trippas and J. Shane Culpepper. Report from the fourth strategic workshop on information retrieval in lorne (swirl 2025). *SIGIR Forum*, 2025.

[204] Rekha Vaidyanathan, Sujoy Das, and Namita Srivastava. A study on retrieval models and query expansion using prf. *International Journal of Scientific & Engineering Research*, 6(2):13–18, 2015.

[205] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.

[206] Junmei Wang, Min Pan, Tingting He, Xiang Huang, Xueyan Wang, and Xinhui Tu. A pseudo-relevance feedback framework combining relevance matching and semantic matching for information retrieval. *Information Processing and Management*, 57(6):102342, 2020.

[207] Le Wang, Ze Luo, Canjia Li, Ben He, Le Sun, Hao Yu, and Yingfei Sun. An end-to-end pseudo relevance feedback framework for neural document retrieval. *Information Processing and Management*, 57(2):102182, 2020.

[208] Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Simlm: Pre-training with representation bottleneck for dense passage retrieval. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 2244–2258. Association for Computational Linguistics, 2023.

[209] Liang Wang, Nan Yang, and Furu Wei. Query2doc: Query expansion with large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 9414–9423. Association for Computational Linguistics, 2023.

[210] Shuai Wang, Hang Li, Harrisen Scells, Daniel Locke, and Guido Zuccon. Mesh term suggestion for systematic review literature search. In *Proceedings of the 25th Australasian Document Computing Symposium, Virtual Event, Australia, 9 December 2021*, pages 8:1–8:8. ACM, 2021.

[211] Shuai Wang, Shengyao Zhuang, and Guido Zuccon. Bert-based dense retrievers require interpolation with BM25 for effective passage retrieval. In *Proceedings of the 2021 ACM SIGIR International Conference on the Theory of Information Retrieval, Virtual Event, Canada, July 11, 2021*, pages 317–324. ACM, 2021.

[212] Shuai Wang, Hang Li, and Guido Zuccon. Mesh suggester: A library and system for mesh term suggestion for systematic review boolean query construction. In *Proceedings of the 16th ACM International Conference on Web Search and Data Mining, WSDM 2023, Singapore, 27 February 2023 - 3 March 2023*, pages 1176–1179. ACM, 2023.

[213] Xiao Wang, Craig Macdonald, and Iadh Ounis. Deep reinforced query reformulation for information retrieval. *arXiv preprint arXiv:2007.07987*, 2020.

[214] Xiao Wang, Craig Macdonald, Nicola Tonellotto, and Iadh Ounis. Pseudo-relevance feedback for multiple representation dense retrieval. In *Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval*, pages 297–306, 2021.

[215] Xiao Wang, Craig Macdonald, and Iadh Ounis. Improving zero-shot retrieval using dense external expansion. *Information Processing and Management*, 59(5):103026, 2022.

[216] Xiao Wang, Sean MacAvaney, Craig Macdonald, and Iadh Ounis. Effective contrastive weighting for dense query expansion. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 12688–12704. Association for Computational Linguistics, 2023.

[217] Xiao Wang, Sean MacAvaney, Craig Macdonald, and Iadh Ounis. Generative query reformulation for effective adhoc search. *arXiv preprint arXiv:2308.00415*, 2023.

[218] Xiao Wang, Craig MacDonald, Nicola Tonellotto, and Iadh Ounis. Colbert-prf: Semantic pseudo-relevance feedback for dense passage and document retrieval. *ACM Transactions on the Web*, 17(1):3:1–3:39, 2023.

[219] Xuwen Wang, Qiang Zhang, Xiaojie Wang, and Yueping Sun. LDA based PSEUDO relevance feedback for cross language information retrieval. In *Proceedings of the 2nd IEEE International Conference on Cloud Computing and Intelligence Systems, CCIS 2012, Hangzhou, China, October 30 - November 1, 2012*, pages 1511–1516. IEEE, 2012.

[220] Xueru Wen, Xiaoyang Chen, Xuanang Chen, Ben He, and Le Sun. Offline pseudo relevance feedback for efficient and effective single-pass dense retrieval. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2023, Taipei, Taiwan, July 23-27, 2023*, pages 2209–2214. ACM, 2023.

[221] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, EMNLP 2020 - Demos, Online, November 16-20, 2020*, pages 38–45. Association for Computational Linguistics, 2020.

[222] Shitao Xiao, Zheng Liu, Yingxia Shao, and Zhao Cao. Retromae: Pre-training retrieval-oriented language models via masked auto-encoder. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 538–548. Association for Computational Linguistics, 2022.

[223] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *Proceedings of the 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[224] Jinxi Xu and W. Bruce Croft. Query expansion using local and global document analysis. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR'96, August 18-22, 1996, Zurich, Switzerland (Special Issue of the SIGIR Forum)*, pages 4–11. ACM, 1996.

[225] Jinxi Xu and W. Bruce Croft. Improving the effectiveness of information retrieval with local context analysis. *ACM Transactions on Information Systems (TOIS)*, 18(1):79–112, 2000.

[226] Yang Xu, Gareth J. F. Jones, and Bin Wang. Query dependent pseudo-relevance feedback based on wikipedia. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2009, Boston, MA, USA, July 19-23, 2009*, pages 59–66. ACM, 2009.

[227] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[228] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[229] Peilin Yang, Hui Fang, and Jimmy Lin. Anserini: Reproducible ranking baselines using lucene. *ACM Journal of Data and Information Quality*, 10(4):16:1–16:20, 2018.

[230] Wei Yang, Haotian Zhang, and Jimmy Lin. Simple applications of BERT for ad hoc document retrieval. *arXiv preprint arXiv:1903.10972*, 2019.

[231] Zhijun Yang, Yang Wang, Jianhou Gan, Hang Li, and Ning Lei. Design and research of intelligent question-answering(q&a) system based on high school course knowledge graph. *Mobile Networks Application*, 26(5):1884–1890, 2021.

[232] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Proceedings of the Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 5754–5764, 2019.

[233] HongChien Yu, Zhuyun Dai, and Jamie Callan. PGT: pseudo relevance feedback using a graph-based transformer. In *Proceedings of the 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28 - April 1, 2021, Proceedings, Part II*, volume 12657 of *Lecture Notes in Computer Science*, pages 440–447. Springer, 2021.

[234] HongChien Yu, Chenyan Xiong, and Jamie Callan. Improving query representations for dense retrieval with pseudo relevance feedback. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021*, pages 3592–3596. ACM, 2021.

[235] Hamed Zamani, Javid Dadashkarimi, Azadeh Shakery, and W. Bruce Croft. Pseudo-relevance feedback based on matrix factorization. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016*, pages 1483–1492. ACM, 2016.

[236] Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 334–342, 2001.

[237] ChengXiang Zhai and John D. Lafferty. Model-based feedback in the language modeling approach to information retrieval. In *Proceedings of the 2001 ACM CIKM International Conference on Information and Knowledge Management, Atlanta, Georgia, USA, November 5-10, 2001*, pages 403–410. ACM, 2001.

[238] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. Repbert: Contextualized text embeddings for first-stage retrieval. *arXiv preprint arXiv:2006.15498*, 2020.

[239] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. Optimizing dense retrieval model training with hard negatives. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, pages 1503–1512. ACM, 2021.

[240] Hang Zhang, Yeyun Gong, Yelong Shen, Jiancheng Lv, Nan Duan, and Weizhu Chen. Adversarial retriever-ranker for dense text retrieval. In *Proceedings of the 10th International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[241] Le Zhao. *Predicting and Solving Term Mismatch for Full-Text Retrieval*. PhD thesis, Carnegie Mellon University, 2012.

[242] Le Zhao and Jamie Callan. Automatic term mismatch diagnosis for selective query expansion. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 515–524, 2012.

[243] Wayne Xin Zhao, Jing Liu, Ruiyang Ren, and Ji-Rong Wen. Dense text retrieval based on pretrained language models: A survey. *ACM Transactions on Information Systems*, 42(4): 89:1–89:60, 2024.

[244] Zhi Zheng, Kai Hui, Ben He, Xianpei Han, Le Sun, and Andrew Yates. BERT-QE: contextualized query expansion for document re-ranking. In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 4718–4728. Association for Computational Linguistics, 2020.

[245] Yutao Zhu, Huaying Yuan, Shuting Wang, Jiongnan Liu, Wenhan Liu, Chenlong Deng, Zhicheng Dou, and Ji-Rong Wen. Large language models for information retrieval: A survey. *arXiv preprint arXiv:2308.07107*, 2023.

[246] Shengyao Zhuang and Guido Zuccon. TILDE: term independent likelihood model for passage re-ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, pages 1483–1492. ACM, 2021.

[247] Shengyao Zhuang, Hang Li, Shuai Wang, and Guido Zuccon. IELAB for TREC deep learning track (dl) 2021. In *Proceedings of the Thirtieth Text REtrieval Conference, TREC 2021, online, November 15-19, 2021*, volume 500-335 of *NIST Special Publication*. National Institute of Standards and Technology (NIST), 2021.

[248] Shengyao Zhuang, Hang Li, and Guido Zuccon. Deep query likelihood model for information retrieval. In *Advances in Information Retrieval - 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28 - April 1, 2021, Proceedings, Part II*, volume 12657 of *Lecture Notes in Computer Science*, pages 463–470. Springer, 2021.

[249] Shengyao Zhuang, Hang Li, and Guido Zuccon. Implicit feedback for dense passage retrieval: A counterfactual approach. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022*, pages 18–28. ACM, 2022.

[250] Shengyao Zhuang, Xueguang Ma, Bevan Koopman, Jimmy Lin, and Guido Zuccon. Promptreps: Prompting large language models to generate dense and sparse representations for zero-shot document retrieval. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 4375–4391. Association for Computational Linguistics, 2024.

[251] Liron Zighelnic and Oren Kurland. Query-drift prevention for robust query expansion. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2008, Singapore, July 20-24, 2008*, pages 825–826. ACM, 2008.

[252] Ingrid Zukerman, Bhavani Raskutti, and Yingying Wen. Query expansion and query reduction in document retrieval. In *Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2003), 3-5 November 2003, Sacramento, California, USA*, pages 552–559. IEEE Computer Society, 2003.

Appendix A

Appendix

Table A.1: Full results of the feedback signal analysis on TREC DL 2019 for BM25 and BM25+Rocchio, comparing signals derived from first-stage retrieval (Baseline) against ground-truth signals from QRels (Comparator). {(S), (M), (W)} means {Strong, Moderate, Weak} signal levels, (B) means signals from first-stage retrieval, (C) means signals from QRels files.

Model	Depth	MAP	Δ (%)	RR	Δ (%)	R@1000	Δ (%)	nDCG@1	Δ (%)	nDCG@3	Δ (%)	nDCG@10	Δ (%)	nDCG@100	Δ (%)
BM25	-	0.2697	-	0.7044	-	0.7687	-	0.5972	-	0.5298	-	0.4971	-	0.4945	-
BM25+Rocchio (B)	1	0.2682	-0.56	0.6977	-0.95	0.7819	1.72	0.5463	-8.52	0.5366	1.28	0.4817	-3.10	0.4454	-9.93
BM25+Rocchio (B)	3	0.2350	-12.87	0.6328	-10.16	0.8044	4.64	0.5000	-16.28	0.4963	-6.32	0.4483	-9.82	0.4318	-12.68
BM25+Rocchio (B) (S)	1	0.3249	20.47	0.8854	25.70	0.8058	4.83	0.6999	17.20	0.6368	20.20	0.5607	12.79	0.5157	4.29
BM25+Rocchio (B) (S)	3	0.3706	37.41	0.8334	18.31	0.8412	9.43	0.6505	8.92	0.6536	23.37	0.6076	22.23	0.5578	12.80
BM25+Rocchio (B) (M)	1	0.2173	-19.43	0.3930	-44.21	0.7732	0.59	0.3853	-35.48	0.3964	-25.18	0.4074	-18.04	0.4358	-11.87
BM25+Rocchio (B) (M)	3	0.2641	-2.08	0.5738	-18.54	0.7979	3.80	0.4842	-18.92	0.4899	-7.53	0.4871	-2.01	0.4975	0.61
BM25+Rocchio (B) (W)	1	0.1917	-28.92	0.3739	-46.92	0.7408	-3.63	0.1806	-69.76	0.2480	-53.19	0.2906	-41.54	0.3592	-27.36
BM25+Rocchio (B) (W)	3	0.1957	-27.44	0.3917	-44.39	0.7641	-0.60	0.2118	-64.53	0.2518	-52.47	0.2902	-41.62	0.3643	-26.33
BM25+Rocchio (C) (S)	1	0.3296	22.21	0.8504	20.73	0.8209	6.79	0.6807	13.98	0.6256	18.08	0.5609	12.83	0.5217	5.50
BM25+Rocchio (C) (S)	3	0.3761	39.45	0.8307	17.93	0.8653	12.57	0.6644	11.25	0.6510	22.88	0.6075	22.21	0.5628	13.81
BM25+Rocchio (C) (M)	1	0.2248	-16.65	0.4311	-38.80	0.7918	3.01	0.3850	-35.53	0.4062	-23.33	0.4175	-16.01	0.4425	-10.52
BM25+Rocchio (C) (M)	3	0.2704	0.26	0.5782	-17.92	0.8298	7.95	0.4873	-18.40	0.5127	-3.23	0.5014	0.87	0.4975	0.61
BM25+Rocchio (C) (W)	1	0.2239	-16.98	0.4856	-31.06	0.7497	-2.47	0.3029	-49.28	0.3464	-34.62	0.3684	-25.89	0.4075	-17.59
BM25+Rocchio (C) (W)	3	0.2425	-10.09	0.5813	-17.48	0.7700	0.17	0.3912	-34.49	0.4058	-23.41	0.4027	-18.99	0.4300	-13.04

Table A.2: Full results of the feedback signal analysis on TREC DL 2019 for ANCE, ANCE+VPRF-Rocchio, and ANCE-PRF, comparing signals derived from first-stage retrieval (Baseline) against ground-truth signals from QRels (Comparator). {(S), (M), (W)} means {Strong, Moderate, Weak} signal levels, (B) means signals from first-stage retrieval, (C) means signals from QRels files.

Model	Depth	MAP	Δ (%)	RR	Δ (%)	R@1000	Δ (%)	nDCG@1	Δ (%)	nDCG@3	Δ (%)	nDCG@10	Δ (%)	nDCG@100	Δ (%)
ANCE	-	0.3908	-	0.8501	-	0.8031	-	0.7222	-	0.7022	-	0.6767	-	0.5860	-
ANCE+Rocchio (B)	1	0.4284	9.62	0.8437	-0.75	0.8253	2.76	0.7222	0.00	0.6922	-1.42	0.6887	1.77	0.6185	5.55
ANCE+Rocchio (B)	3	0.4300	10.03	0.8177	-3.81	0.8179	1.84	0.7037	-2.56	0.7023	0.01	0.6790	0.34	0.6202	5.84
ANCE-PRF (B)	3	0.4423	13.18	0.8721	2.59	0.8293	3.26	0.7361	1.92	0.7204	2.59	0.7074	4.54	0.6270	7.00
ANCE+Rocchio (B) (S)	1	0.4988	27.64	0.9198	8.20	0.8454	5.27	0.7986	10.58	0.7754	10.42	0.7499	10.82	0.6651	13.50
ANCE+Rocchio (B) (S)	3	0.5119	30.99	0.8816	3.71	0.8531	6.23	0.7708	6.73	0.7690	9.51	0.7511	10.99	0.6749	15.17
ANCE-PRF (B) (S)	3	0.4907	25.56	0.9060	6.58	0.8388	4.45	0.7986	10.58	0.7722	9.97	0.7484	10.60	0.6583	12.34
ANCE+Rocchio (B) (M)	1	0.4025	2.99	0.7521	-11.53	0.8317	3.56	0.6703	-7.19	0.6603	-5.97	0.6558	-3.09	0.6176	5.39
ANCE+Rocchio (B) (M)	3	0.4365	11.69	0.8278	-2.62	0.8477	5.55	0.7161	-0.84	0.6951	-1.01	0.6845	1.15	0.6444	9.97
ANCE-PRF (B) (M)	3	0.4369	11.80	0.7740	-8.95	0.8324	3.65	0.6620	-8.34	0.6905	-1.67	0.6936	2.50	0.6385	8.96
ANCE+Rocchio (B) (W)	1	0.3583	-8.32	0.7182	-15.52	0.7980	-0.64	0.5657	-21.67	0.5780	-17.69	0.5778	-14.62	0.5480	-6.48
ANCE+Rocchio (B) (W)	3	0.3915	0.18	0.7886	-7.23	0.8130	1.23	0.6713	-7.05	0.6526	-7.06	0.6480	-4.24	0.5836	-0.41
ANCE-PRF (B) (W)	3	0.3929	0.54	0.7121	-16.23	0.8159	1.59	0.5232	-27.55	0.5733	-18.36	0.5906	-12.72	0.5791	-1.18
ANCE+Rocchio (C) (S)	1	0.5078	29.94	0.9182	8.01	0.8614	7.26	0.7921	9.68	0.7744	10.28	0.7513	11.02	0.6713	14.56
ANCE+Rocchio (C) (S)	3	0.5154	31.88	0.8790	3.40	0.8725	8.64	0.7685	6.41	0.7542	7.41	0.7444	10.00	0.6772	15.56
ANCE-PRF (C) (S)	3	0.4983	27.51	0.9054	6.51	0.8444	5.14	0.7724	6.95	0.7664	9.14	0.7471	10.40	0.6597	12.58
ANCE+Rocchio (C) (M)	1	0.4254	8.85	0.7932	-6.69	0.8413	4.76	0.7022	-2.77	0.6957	-0.93	0.6783	0.24	0.6308	7.65
ANCE+Rocchio (C) (M)	3	0.4533	15.99	0.8359	-1.67	0.8559	6.57	0.7315	1.29	0.7095	1.04	0.6986	3.24	0.6519	11.25
ANCE-PRF (C) (M)	3	0.4276	9.42	0.8207	-3.46	0.8353	4.01	0.7049	-2.40	0.6986	-0.51	0.6926	2.35	0.6317	7.80
ANCE+Rocchio (C) (W)	1	0.3838	-1.79	0.7736	-9.00	0.7954	-0.96	0.6385	-11.59	0.6376	-9.20	0.6174	-8.76	0.5587	-4.66
ANCE+Rocchio (C) (W)	3	0.4159	6.42	0.8371	-1.53	0.8193	2.02	0.7164	-0.80	0.6949	-1.04	0.6779	0.18	0.5979	2.03
ANCE-PRF (C) (W)	3	0.3934	0.67	0.7714	-9.26	0.8031	0.00	0.6343	-12.17	0.6297	-10.32	0.6364	-5.96	0.5766	-1.60

Table A.3: Full results of the feedback signal analysis on TREC DL 2019 for TCTV2, TCTV2+VPRF-Rocchio, and TCTV2-PRF, comparing signals derived from first-stage retrieval (Baseline) against ground-truth signals from QRels (Comparator). {(S), (M), (W)} means {Strong, Moderate, Weak} signal levels, (B) means signals from first-stage retrieval, (C) means signals from QRels files.

Model	Depth	MAP	Δ (%)	RR	Δ (%)	R@1000	Δ (%)	nDCG@1	Δ (%)	nDCG@3	Δ (%)	nDCG@10	Δ (%)	nDCG@100	Δ (%)
TCTV2	-	0.4676	-	0.8788	-	0.8794	-	0.8009	-	0.7488	-	0.7309	-	0.6631	-
TCTV2+Rocchio (B)	1	0.4980	6.50	0.8821	0.38	0.8991	2.24	0.8009	0.00	0.7621	1.78	0.7410	1.38	0.6874	3.66
TCTV2+Rocchio (B)	3	0.4949	5.84	0.8682	-1.21	0.8942	1.68	0.7917	-1.15	0.7464	-0.32	0.7406	1.33	0.6876	3.69
TCTV2-PRF (B)	3	0.4901	4.81	0.8615	-1.97	0.8888	1.07	0.7500	-6.36	0.7606	1.58	0.7456	2.01	0.6802	2.58
TCTV2+Rocchio (B) (S)	1	0.5503	17.69	0.9762	11.08	0.9153	4.08	0.8156	1.84	0.7924	5.82	0.7691	5.23	0.7132	7.56
TCTV2+Rocchio (B) (S)	3	0.5769	23.37	0.9136	3.96	0.9292	5.66	0.8002	-0.09	0.7957	6.26	0.7773	6.35	0.7366	11.08
TCTV2-PRF (B) (S)	3	0.5326	13.90	0.8807	0.22	0.9134	3.87	0.7654	-4.43	0.7641	2.04	0.7600	3.98	0.7102	7.10
TCTV2+Rocchio (B) (M)	1	0.4107	-12.17	0.6181	-29.67	0.8940	1.66	0.5485	-31.51	0.5952	-20.51	0.6331	-13.38	0.6417	-3.23
TCTV2+Rocchio (B) (M)	3	0.4675	-0.02	0.7999	-8.98	0.9088	3.34	0.7164	-10.55	0.7060	-5.72	0.7052	-3.52	0.6861	3.47
TCTV2-PRF (B) (M)	3	0.4841	3.53	0.8550	-2.71	0.8977	2.08	0.7558	-5.63	0.7403	-1.14	0.7338	0.40	0.6915	4.28
TCTV2+Rocchio (B) (W)	1	0.3629	-22.39	0.5752	-34.55	0.8686	-1.23	0.3439	-57.06	0.4437	-40.75	0.5188	-29.02	0.5601	-15.53
TCTV2+Rocchio (B) (W)	3	0.4408	-5.73	0.8036	-8.56	0.8875	0.92	0.6921	-13.58	0.6608	-11.75	0.6692	-8.44	0.6351	-4.22
TCTV2-PRF (B) (W)	3	0.4705	0.62	0.8487	-3.43	0.889	1.09	0.7315	-8.67	0.7110	-5.05	0.7053	-3.50	0.6582	-0.74
TCTV2+Rocchio (C) (S)	1	0.5489	17.39	0.9726	10.67	0.919	4.50	0.8102	1.16	0.7903	5.54	0.7646	4.61	0.7134	7.59
TCTV2+Rocchio (C) (S)	3	0.5782	23.65	0.8948	1.82	0.9427	7.20	0.7855	-1.92	0.7914	5.69	0.7775	6.38	0.7362	11.02
TCTV2-PRF (C) (S)	3	0.5359	14.61	0.8956	1.91	0.9217	4.81	0.7732	-3.46	0.7688	2.67	0.7556	3.38	0.7112	7.25
TCTV2+Rocchio (C) (M)	1	0.4219	-9.77	0.6507	-25.96	0.9000	2.34	0.5784	-27.78	0.6248	-16.56	0.6509	-10.95	0.6479	-2.29
TCTV2+Rocchio (C) (M)	3	0.4838	3.46	0.8206	-6.62	0.9225	4.90	0.7245	-9.54	0.7157	-4.42	0.7152	-2.15	0.6948	4.78
TCTV2-PRF (C) (M)	3	0.4926	5.35	0.8627	-1.83	0.9105	3.54	0.7570	-5.48	0.7349	-1.86	0.7377	0.93	0.6964	5.02
TCTV2+Rocchio (C) (W)	1	0.3896	-16.68	0.6698	-23.78	0.8548	-2.80	0.4780	-40.32	0.5300	-29.22	0.5702	-21.99	0.5786	-12.74
TCTV2+Rocchio (C) (W)	3	0.4784	2.31	0.8497	-3.31	0.8825	0.35	0.7431	-7.22	0.7202	-3.82	0.7055	-3.48	0.6645	0.21
TCTV2-PRF (C) (W)	3	0.4774	2.10	0.8819	0.35	0.8878	0.96	0.7662	-4.33	0.7444	-0.59	0.7160	-2.04	0.6658	0.41

Table A.4: Full results of the feedback signal analysis on TREC DL 2019 for DistilBERT, DistilBERT+VPRF-Rocchio, and DistilBERT-PRF, comparing signals derived from first-stage retrieval (Baseline) against ground-truth signals from QRels (Comparator). {(S), (M), (W)} means {Strong, Moderate, Weak} signal levels, (B) means signals from first-stage retrieval, (C) means signals from QRels files.

Model	Depth	MAP	Δ (%)	RR	Δ (%)	R@1000	Δ (%)	nDCG@1	Δ (%)	nDCG@3	Δ (%)	nDCG@10	Δ (%)	nDCG@100	Δ (%)
DistilBERT	-	0.4832	-	0.8763	-	0.8905	-	0.7454	-	0.7466	-	0.7319	-	0.6698	-
DistilBERT+Rocchio (B)	1	0.5076	5.05	0.8670	-1.06	0.8859	-0.52	0.7454	0.00	0.7470	0.05	0.7247	-0.98	0.6831	1.99
DistilBERT+Rocchio (B)	3	0.5156	6.71	0.8606	-1.79	0.8928	0.26	0.7731	3.72	0.7411	-0.74	0.7387	0.93	0.6897	2.97
DistilBERT-PRF (B)	3	0.4996	3.39	0.8588	-2.00	0.8968	0.71	0.7546	1.23	0.7648	2.44	0.7386	0.92	0.6778	1.19
DistilBERT+Rocchio (B) (S)	1	0.5652	16.97	0.9726	10.99	0.9263	4.02	0.8021	7.61	0.7980	6.88	0.7719	5.47	0.7200	7.49
DistilBERT+Rocchio (B) (S)	3	0.5931	22.74	0.9410	7.38	0.9336	4.84	0.7967	6.88	0.8068	8.06	0.7971	8.91	0.7433	10.97
DistilBERT-PRF (B) (S)	3	0.5408	11.92	0.8954	2.18	0.9124	2.46	0.7963	6.58	0.7821	4.75	0.7630	4.25	0.7041	5.12
DistilBERT+Rocchio (B) (M)	1	0.4188	-13.33	0.5198	-40.68	0.9074	1.90	0.4510	-39.50	0.5622	-24.70	0.6219	-15.03	0.6429	-4.02
DistilBERT+Rocchio (B) (M)	3	0.4847	0.31	0.8143	-7.08	0.9193	3.23	0.7029	-5.70	0.7049	-5.59	0.7075	-3.33	0.6981	4.23
DistilBERT-PRF (B) (M)	3	0.5049	4.49	0.8779	0.18	0.9069	1.84	0.7755	4.04	0.7472	0.08	0.7369	0.68	0.6900	3.02
DistilBERT+Rocchio (B) (W)	1	0.3635	-24.77	0.5152	-41.21	0.8784	-1.36	0.2458	-67.02	0.3966	-46.88	0.5004	-31.63	0.5562	-16.96
DistilBERT+Rocchio (B) (W)	3	0.4441	-8.09	0.8022	-8.46	0.8970	0.73	0.6667	-10.56	0.6622	-11.30	0.6561	-10.36	0.6371	-4.88
DistilBERT-PRF (B) (W)	3	0.5047	4.45	0.8757	-0.07	0.9002	1.09	0.7639	2.48	0.7386	-1.07	0.7262	-0.78	0.6789	1.36
DistilBERT+Rocchio (C) (S)	1	0.5710	18.17	0.9676	10.42	0.9382	5.36	0.7975	6.99	0.7871	5.42	0.7680	4.93	0.7237	8.05
DistilBERT+Rocchio (C) (S)	3	0.6044	25.08	0.9394	7.20	0.9540	7.13	0.8148	9.31	0.8046	7.77	0.7939	8.47	0.7490	11.82
DistilBERT-PRF (C) (S)	3	0.5409	11.94	0.9084	3.66	0.9243	3.80	0.8241	10.56	0.7852	5.17	0.7621	4.13	0.7023	4.85
DistilBERT+Rocchio (C) (M)	1	0.4332	-10.35	0.6165	-29.65	0.9181	3.10	0.5341	-28.35	0.6100	-18.30	0.6419	-12.30	0.6517	-2.70
DistilBERT+Rocchio (C) (M)	3	0.5009	3.66	0.8246	-5.90	0.9303	4.47	0.7145	-4.15	0.7348	-1.58	0.7199	-1.64	0.7071	5.57
DistilBERT-PRF (C) (M)	3	0.5150	6.58	0.8649	-1.30	0.9103	2.22	0.7708	3.41	0.7602	1.82	0.7399	1.09	0.6954	3.82
DistilBERT+Rocchio (C) (W)	1	0.3954	-18.17	0.6506	-25.76	0.8810	-1.07	0.4313	-42.14	0.5118	-31.45	0.5591	-23.61	0.5790	-13.56
DistilBERT+Rocchio (C) (W)	3	0.4936	2.15	0.8650	-1.29	0.9058	1.72	0.7627	2.32	0.7409	-0.76	0.7171	-2.02	0.6751	0.79
DistilBERT+Rocchio (C) (W)	3	0.5074	5.01	0.8669	-1.07	0.9104	2.23	0.7523	0.93	0.7564	1.31	0.7305	-0.19	0.6832	2.00

Table A.5: Full results of the feedback signal analysis on TREC DL 2020 for BM25 and BM25+Rocchio, comparing signals derived from first-stage retrieval (Baseline) against ground-truth signals from QRels (Comparator). {(S), (M), (W)} means {Strong, Moderate, Weak} signal levels, (B) means signals from first-stage retrieval, (C) means signals from QRels files.

Model	Depth	MAP	Δ (%)	RR	Δ (%)	R@1000	Δ (%)	nDCG@1	Δ (%)	nDCG@3	Δ (%)	nDCG@10	Δ (%)	nDCG@100	Δ (%)
BM25	-	0.2870	-	0.6531	-	0.7938	-	0.5595	-	0.5155	-	0.4959	-	0.4959	-
BM25+Rocchio (B)	1	0.2124	-25.99	0.5994	-8.22	0.7571	-4.62	0.5595	0	0.4483	-13.04	0.3956	-20.23	0.3952	-20.31
BM25+Rocchio (B)	3	0.1936	-32.54	0.5198	-20.41	0.7447	-6.19	0.4206	-24.83	0.4246	-17.63	0.3864	-22.08	0.3734	-24.70
BM25+Rocchio (B) (S)	1	0.3378	17.70	0.9002	37.83	0.7922	-0.20	0.7567	35.25	0.6663	29.25	0.5729	15.53	0.5008	0.99
BM25+Rocchio (B) (S)	3	0.3936	37.143	0.8859	35.65	0.8422	6.10	0.7695	37.53	0.7040	36.57	0.6411	29.28	0.5566	12.24
BM25+Rocchio (B) (M)	1	0.1951	-32.02	0.3618	-44.60	0.7703	-2.96	0.3824	-31.65	0.3911	-24.13	0.4033	-18.67	0.4198	-15.35
BM25+Rocchio (B) (M)	3	0.1960	-31.71	0.3855	-40.97	0.7956	0.23	0.374	-33.15	0.3821	-25.88	0.4039	-18.55	0.4358	-12.11
BM25+Rocchio (B) (W)	1	0.1810	-36.93	0.3228	-50.57	0.7366	-7.20	0.1379	-75.35	0.2256	-56.23	0.2888	-41.76	0.3504	-29.34
BM25+Rocchio (B) (W)	3	0.1839	-35.92	0.3712	-43.16	0.7433	-6.36	0.2153	-61.51	0.2389	-53.65	0.2866	-42.20	0.3506	-29.30
BM25+Rocchio (C) (S)	1	0.3497	21.84	0.8733	33.71	0.8242	3.82	0.7238	29.36	0.6583	27.70	0.5792	16.79	0.5142	3.69
BM25+Rocchio (C) (S)	3	0.3993	39.12	0.8714	33.42	0.8639	8.83	0.7348	31.33	0.6979	35.38	0.6356	28.17	0.5579	12.50
BM25+Rocchio (C) (M)	1	0.2042	-28.85	0.4160	-36.30	0.7805	-1.67	0.3985	-28.77	0.4096	-20.54	0.4124	-16.83	0.4272	-13.85
BM25+Rocchio (C) (M)	3	0.2136	-25.57	0.4763	-27.07	0.8134	2.46	0.4553	-18.62	0.4523	-12.25	0.4404	-11.19	0.4601	-7.21
BM25+Rocchio (C) (W)	1	0.2108	-26.55	0.4337	-33.59	0.7591	-4.37	0.2741	-51.00	0.3192	-38.07	0.3457	-30.28	0.3780	-23.77
BM25+Rocchio (C) (W)	3	0.2501	-12.85	0.5431	-16.84	0.7874	-0.80	0.3988	-28.72	0.4155	-19.39	0.4135	-16.61	0.4257	-14.15

Table A.6: Full results of the feedback signal analysis on TREC DL 2020 for ANCE, ANCE+VPRF-Rocchio, and ANCE-PRF, comparing signals derived from first-stage retrieval (Baseline) against ground-truth signals from QRels (Comparator). {(S), (M), (W)} means {Strong, Moderate, Weak} signal levels, (B) means signals from first-stage retrieval, (C) means signals from QRels files.

Model	Depth	MAP	Δ (%)	RR	Δ (%)	R@1000	Δ (%)	nDCG@1	Δ (%)	nDCG@3	Δ (%)	nDCG@10	Δ (%)	nDCG@100	Δ (%)
ANCE	-	0.4047	-	0.8275	-	0.7804	-	0.7619	-	0.7479	-	0.6806	-	0.5670	-
ANCE+Rocchio (B)	1	0.4249	4.99	0.8118	-1.89	0.8135	4.24	0.7619	0	0.7455	-0.32	0.6875	1.01	0.5810	2.46
ANCE+Rocchio (B)	3	0.4220	4.27	0.8377	1.23	0.7958	1.97	0.7540	-1.03	0.7472	-0.09	0.6841	0.51	0.5760	1.58
ANCE-PRF (B)	3	0.4340	7.23	0.8881	7.32	0.8286	6.17	0.8571	12.49	0.7792	4.18	0.7275	6.89	0.5897	4.00
ANCE+Rocchio (B) (S)	1	0.4871	20.36	0.9547	15.37	0.8297	6.32	0.8102	6.34	0.7920	5.90	0.7255	6.60	0.6085	7.32
ANCE+Rocchio (B) (S)	3	0.5259	29.95	0.9164	10.74	0.8377	7.34	0.8132	6.73	0.8158	9.08	0.7670	12.69	0.6398	12.84
ANCE-PRF (B) (S)	3	0.4798	18.56	0.8771	5.99	0.8241	5.60	0.7798	2.35	0.7685	2.75	0.7249	6.51	0.6078	7.20
ANCE+Rocchio (B) (M)	1	0.3277	-19.03	0.5363	-35.19	0.8063	3.32	0.4863	-36.17	0.5555	-25.73	0.5777	-15.12	0.5471	-3.51
ANCE+Rocchio (B) (M)	3	0.3541	-12.50	0.6839	-17.35	0.8174	4.74	0.6177	-18.93	0.6286	-15.95	0.6091	-10.51	0.5738	1.20
ANCE-PRF (B) (M)	3	0.3703	-8.50	0.6882	-16.83	0.8143	4.34	0.6359	-16.54	0.6409	-14.31	0.6302	-7.41	0.5627	-0.76
ANCE+Rocchio (B) (W)	1	0.2753	-31.97	0.4823	-41.72	0.7688	-1.49	0.2493	-67.28	0.3732	-50.10	0.4292	-36.94	0.4499	-20.65
ANCE+Rocchio (B) (W)	3	0.3180	-21.42	0.6949	-16.02	0.7832	0.36	0.5575	-26.83	0.5609	-25.00	0.5250	-22.86	0.4968	-12.38
ANCE-PRF (B) (W)	3	0.3594	-11.19	0.7185	-13.17	0.7911	1.37	0.6181	-18.87	0.6142	-17.88	0.5935	-12.80	0.5280	-6.88
ANCE+Rocchio (C) (S)	1	0.4851	19.87	0.9213	11.34	0.8517	9.14	0.7775	2.05	0.7690	2.82	0.7133	4.80	0.6041	6.54
ANCE+Rocchio (C) (S)	3	0.5311	31.23	0.8954	8.21	0.8781	12.52	0.7811	2.52	0.7967	6.52	0.7542	10.81	0.6434	13.47
ANCE-PRF (C) (S)	3	0.4843	19.67	0.8581	3.70	0.8555	9.62	0.7837	2.86	0.7865	5.16	0.7342	7.88	0.6165	8.73
ANCE+Rocchio (C) (M)	1	0.3345	-17.35	0.6009	-27.38	0.8103	3.83	0.5390	-29.26	0.5816	-22.24	0.5836	-14.25	0.5447	-3.93
ANCE+Rocchio (C) (M)	3	0.3644	-9.96	0.7134	-13.79	0.8306	6.43	0.6171	-19.01	0.6403	-14.39	0.6159	-9.51	0.5755	1.50
ANCE-PRF (C) (M)	3	0.3920	-3.14	0.7188	-13.14	0.8133	4.22	0.6528	-14.32	0.6598	-11.78	0.6336	-6.91	0.5679	0.16
ANCE+Rocchio (C) (W)	1	0.2769	-31.58	0.5721	-30.86	0.7461	-4.40	0.3882	-49.05	0.4443	-40.59	0.4504	-33.82	0.4255	-24.96
ANCE+Rocchio (C) (W)	3	0.3464	-14.41	0.7622	-7.89	0.7691	-1.45	0.6319	-17.06	0.6260	-16.30	0.5723	-15.91	0.4879	-13.95
ANCE-PRF (C) (W)	3	0.3722	-8.03	0.7127	-13.87	0.7752	-0.67	0.5942	-22.01	0.6239	-16.58	0.5909	-13.18	0.5202	-8.25

Table A.7: Full results of the feedback signal analysis on TREC DL 2020 for TCTV2, TCTV2+VPRF-Rocchio, and TCTV2-PRF, comparing signals derived from first-stage retrieval (Baseline) against ground-truth signals from QRels (Comparator). {(S), (M), (W)} means {Strong, Moderate, Weak} signal levels, (B) means signals from first-stage retrieval, (C) means signals from QRels files.

Model	Depth	MAP	Δ (%)	RR	Δ (%)	R@1000	Δ (%)	nDCG@1	Δ (%)	nDCG@3	Δ (%)	nDCG@10	Δ (%)	nDCG@100	Δ (%)
TCTV2	-	0.4047	-	0.8275	-	0.7804	-	0.7619	-	0.7479	-	0.6806	-	0.5670	-
TCTV2+Rocchio (B)	1	0.4744	17.22	0.8402	1.53	0.8708	11.58	0.7976	4.68	0.7565	1.14	0.7079	4.01	0.6205	9.43
TCTV2+Rocchio (B)	3	0.4904	21.17	0.8321	0.55	0.8655	10.90	0.7817	2.59	0.7592	1.51	0.7144	4.96	0.6274	10.65
TCTV2-PRF (B)	3	0.4864	20.18	0.8774	6.03	0.8562	9.71	0.8254	8.33	0.8038	7.47	0.7331	7.71	0.6252	10.26
TCTV2+Rocchio (B) (S)	1	0.5415	33.80	0.9722	17.49	0.8905	14.11	0.8094	6.23	0.7969	6.55	0.7535	10.71	0.6536	15.27
TCTV2+Rocchio (B) (S)	3	0.6018	48.70	0.9484	14.61	0.9142	17.15	0.8323	9.24	0.8227	10.00	0.7925	16.44	0.6918	22.01
TCTV2-PRF (B) (S)	3	0.5348	32.15	0.9220	11.42	0.8780	12.51	0.8307	9.03	0.8223	9.95	0.7683	12.89	0.6512	14.85
TCTV2+Rocchio (B) (M)	1	0.3431	-15.22	0.4156	-49.78	0.8628	10.56	0.4004	-47.45	0.4922	-34.19	0.5511	-19.03	0.5698	0.49
TCTV2+Rocchio (B) (M)	3	0.3754	-7.24	0.5763	-30.36	0.8830	13.15	0.5321	-30.16	0.5665	-24.25	0.5792	-14.90	0.6124	8.01
TCTV2-PRF (B) (M)	3	0.4684	15.74	0.8540	3.20	0.8583	9.98	0.7791	2.26	0.7545	0.88	0.7113	4.51	0.6355	12.08
TCTV2+Rocchio (B) (W)	1	0.2691	-33.51	0.3678	-55.55	0.8152	4.46	0.1025	-86.55	0.2727	-63.54	0.3726	-45.25	0.4407	-22.28
TCTV2+Rocchio (B) (W)	3	0.3440	-15.00	0.5926	-28.39	0.8352	7.02	0.4484	-41.15	0.4785	-36.02	0.4792	-29.59	0.5031	-11.27
TCTV2-PRF (B) (W)	3	0.4703	16.21	0.8827	6.67	0.8551	9.57	0.7748	1.69	0.7488	0.12	0.7001	2.87	0.6079	7.21
TCTV2+Rocchio (C) (S)	1	0.5437	34.35	0.9668	16.83	0.9022	15.61	0.7972	4.63	0.7733	3.40	0.7383	8.48	0.6545	15.43
TCTV2+Rocchio (C) (S)	3	0.6213	53.52	0.9253	11.82	0.9333	19.59	0.8066	5.87	0.8089	8.16	0.7902	16.10	0.7041	24.18
TCTV2-PRF (C) (S)	3	0.5376	32.84	0.9266	11.98	0.8886	13.86	0.8294	8.86	0.8267	10.54	0.7612	11.84	0.6544	15.41
TCTV2+Rocchio (C) (M)	1	0.3411	-15.72	0.4405	-46.77	0.8626	10.53	0.4256	-44.14	0.4975	-33.48	0.5480	-19.48	0.5628	-0.74
TCTV2+Rocchio (C) (M)	3	0.3977	-1.73	0.6544	-20.92	0.8907	14.13	0.5883	-22.79	0.6121	-18.16	0.6113	-10.18	0.6273	10.63
TCTV2-PRF (C) (M)	3	0.4831	19.37	0.8704	5.18	0.8705	11.55	0.7708	1.17	0.7708	3.06	0.7182	5.52	0.6435	13.49
TCTV2+Rocchio (C) (W)	1	0.2750	-32.05	0.4257	-48.56	0.7995	2.45	0.1746	-77.08	0.3134	-58.10	0.3985	-41.45	0.4261	-24.85
TCTV2+Rocchio (C) (W)	3	0.4134	2.15	0.7619	-7.93	0.8471	8.55	0.6081	-20.19	0.6354	-15.04	0.6076	-10.73	0.5489	-3.19
TCTV2-PRF (C) (W)	3	0.4792	18.41	0.8897	7.52	0.8615	10.39	0.7877	3.39	0.7614	1.81	0.7165	5.27	0.6171	8.84

Table A.8: Full results of the feedback signal analysis on TREC DL 2020 for DistilBERT, DistilBERT+VPRF-Rocchio, and DistilBERT-PRF, comparing signals derived from first-stage retrieval (Baseline) against ground-truth signals from QRels (Comparator). {(S), (M), (W)} means {Strong, Moderate, Weak} signal levels, (B) means signals from first-stage retrieval, (C) means signals from QRels files.

Model	Depth	MAP	Δ (%)	RR	Δ (%)	R@1000	Δ (%)	nDCG@1	Δ (%)	nDCG@3	Δ (%)	nDCG@10	Δ (%)	nDCG@100	Δ (%)
DistilBERT	-	0.4742	-	0.8677	-	0.8770	-	0.7778	-	0.7887	-	0.7207	-	0.6382	-
DistilBERT+Rocchio (B)	1	0.4896	3.24	0.8610	-0.77	0.8916	1.66	0.7778	0	0.7972	1.07	0.7487	3.88	0.6462	1.25
DistilBERT+Rocchio (B)	3	0.4974	4.89	0.8899	2.55	0.9101	3.77	0.8135	4.58	0.7973	1.09	0.7513	4.24	0.6535	2.39
DistilBERT-PRF (B)	3	0.4860	2.48	0.8810	1.53	0.8777	0.07	0.7976	2.54	0.7803	-1.06	0.7306	1.37	0.6293	-1.39
DistilBERT+Rocchio (B) (S)	1	0.5169	9.00	0.9924	14.37	0.9003	2.66	0.8222	5.71	0.7745	-1.80	0.7242	0.49	0.6404	0.34
DistilBERT+Rocchio (B) (S)	3	0.6022	26.99	0.9738	12.23	0.9255	5.53	0.8462	8.79	0.8260	4.73	0.7955	10.38	0.7030	10.15
DistilBERT-PRF (B) (S)	3	0.5264	11.01	0.9082	4.67	0.8915	1.65	0.8185	5.23	0.8084	2.50	0.7542	4.65	0.6561	2.80
DistilBERT+Rocchio (B) (M)	1	0.2748	-42.05	0.3113	-64.12	0.8538	-2.65	0.3621	-53.45	0.4073	-48.36	0.4670	-35.20	0.5193	-18.63
DistilBERT+Rocchio (B) (M)	3	0.4075	-14.07	0.6412	-26.10	0.8889	1.36	0.5747	-26.11	0.6140	-22.15	0.6440	-10.64	0.6315	-1.05
DistilBERT-PRF (B) (M)	3	0.4701	-0.86	0.8215	-5.32	0.8816	0.52	0.7229	-7.06	0.7499	-4.92	0.7018	-2.62	0.6340	-0.66
DistilBERT+Rocchio (B) (W)	1	0.1986	-58.12	0.2527	-70.88	0.7983	-8.97	0.0126	-98.38	0.1607	-79.62	0.2658	-63.12	0.3693	-42.13
DistilBERT+Rocchio (B) (W)	3	0.3904	-17.67	0.7441	-14.24	0.8603	-1.90	0.5903	-24.11	0.5873	-25.54	0.5702	-20.88	0.5555	-12.96
DistilBERT-PRF (B) (W)	3	0.4746	0.08	0.8626	-0.59	0.8747	-0.26	0.7474	-3.91	0.7675	-2.69	0.6991	-3.00	0.6213	-2.65
DistilBERT+Rocchio (C) (S)	1	0.5240	10.50	0.9893	14.01	0.9135	4.16	0.8192	5.32	0.7647	-3.04	0.7182	-0.35	0.6420	0.60
DistilBERT+Rocchio (C) (S)	3	0.6260	32.01	0.9628	10.96	0.9491	8.22	0.8386	7.82	0.8364	6.05	0.7992	10.89	0.7149	12.02
DistilBERT-PRF (C) (S)	3	0.5249	10.69	0.8671	-0.07	0.8974	2.33	0.7735	-0.55	0.7800	-1.10	0.7428	3.07	0.6558	2.76
DistilBERT+Rocchio (C) (M)	1	0.2734	-42.35	0.3263	-62.39	0.8592	-2.03	0.3746	-51.84	0.4179	-47.01	0.4622	-35.87	0.5147	-19.35
DistilBERT+Rocchio (C) (M)	3	0.4193	-11.58	0.7156	-17.53	0.9063	3.34	0.6210	-20.16	0.6360	-19.36	0.6519	-9.55	0.6465	1.30
DistilBERT-PRF (C) (M)	3	0.4809	1.41	0.8414	-3.03	0.8832	0.71	0.7242	-6.89	0.7537	-4.44	0.7082	-1.73	0.6391	0.14
DistilBERT+Rocchio (C) (W)	1	0.1719	-63.75	0.2562	-70.47	0.7678	-12.45	0.0463	-94.05	0.1499	-80.99	0.2313	-67.91	0.3068	-51.93
DistilBERT+Rocchio (C) (W)	3	0.4523	-4.62	0.8120	-6.42	0.8780	0.11	0.6776	-12.88	0.6996	-11.30	0.6578	-8.73	0.5906	-7.46
DistilBERT-PRF (C) (W)	3	0.4884	2.99	0.8598	-0.91	0.8874	1.19	0.7550	-2.93	0.7726	-2.04	0.7189	-0.25	0.6348	-0.53

Live long and prosper.

S’chn T’gai Spock,
"Amok Time"
September 15, 1967