Evalugator — Rapid, Agile Development and Evaluation of Retrieval Augmented Generation Systems Without Labels

Bevan Koopman[1,2], Hang Li[1], Shuai Wang[1], and Guido Zuccon[1]

[1] The University of Queensland, Brisbane, Australia
[2] CSIRO, Brisbane, Australia
{b.koopman, hang.li, shuai.wang2, g.zuccon}@uq.edu.au

Abstract. Evaluating complex Retrieval Augmented Generation (RAG) systems in real-world settings is challenging. There is often a lack of fine-grained labelled data and the absence of comprehensive evaluation tools that can assess individual components of a pipeline. This hinders rapid, rigorous development, particularly for agentic RAG systems. We was our experienced at GuideStream.AI, a startup developing an AI for clinical guideline recommendation. To address this gap, we developed Evalugator, a suite of agentic components to support agile development and evaluation. Evalugator features: (1) generation of synthetic queries, relevance assessments, answers and evaluation criteria for training and evaluation in new domains; (2) LLM-based judging agents; and (3) simple UI and API tools to launch experiments and analyse results. This paper uses Evalugator as a case study to demonstrate how a principled, agent-based evaluation framework can support the rapid development of complex RAG systems in a startup environment.

Keywords: Retrieval augmented generation · Evaluation

1 Introduction

GuideStream.AI is a startup incubated within The University of Queensland focused on building a specialised retrieval system that offers medical professionals personalised, real-time, highly effective access to clinical guidelines relevant to their patient. Core to our product is a complex agentic Retrieval Augmented Generation (RAG) pipeline, that has undergone extensive training. A core principle at GuideStream.AI is grounding the product on rigorous evaluation practices. While RAG has proven effective in research settings, we encountered the following challenges that hampered our aim of rapid development and integrated experimentation workflow:

1. Often developers have no training or evaluation data related to their setting to develop their RAG system, especially data that allows evaluation to span multiple dimensions and preferences, e.g., relevance, factuality, quality, layout/format, etc.

2. Even when evaluation data is available, it might be difficult for developers not experienced with IR evaluation practices to run evaluation experiments.
3. It is difficult to evaluate individual components of the RAG system (e.g., retriever vs generator effectiveness) — this makes it much harder to diagnose issues and focus development efforts.

We recognised the tension between being able to rapidly develop a prototype in a startup environment and maintaining some scientific rigour in model development. To try and manage this balance we developed a series of components — collectively dubbed Evalugator— to aid us in rapid and agile development and evaluation. The main components were:

– QuestionFisher, providing the ability to generate synthetic queries, document-level relevance assessment and answers for domains where these do not exist.
– Seperate LLM-based judging agents [3]: RetrieverRater for document level judging for the retriever; GenRat for answer quality judging of the generator based on multiple criteria [4].
– Simple to use tools (UI and API) to launch evaluation experiments that utilise the judgement agents, allowing developers to get quick evaluation feedback. Flexible display of results so developers can dig into the results and understand the behaviour of the RAG systems.

Other tools have been proposed for RAG evaluation in fast-paced development environments. An example is RURAGE [2], which is limited in only focusing on the generation evaluation, ignoring the interplay with other components in the RAG pipeline, and it intentionally does not provide LLM-judge methods, which instead we believe are key in our settings for fine-grained evaluation of response preferences and dimensions.

Our presentation will use Evalugator as an example of how a principled approach to evaluation can support rapid development of RAG system in a startup environment.

2 Technical Overview

Figure 1 provides an overview of Evalugator. Key steps from the diagram are:

1. The QuestionFisher agent takes an individual document and uses an LLM to generate questions for which the contents of the document provide an answer, a short ground truth answer, a long ground truth answer, and the particular page number.[3] All this is stored as a QuestionSet in Evalugator's datastore. Multiple QuestionSet can be loaded and split up for training, validation, testing, etc.; all these can be visualised via Evalugator's UI.
2. Users launch experiments with a QuestionSet and specific RAG system settings (e.g., index, retrieval model, generator model). The user also specifies if RetrieverRater and GenRat should be run on the results. Evalugator pushes an entry for each question into a queue for the Live RAG system.

[3] In our case this was because documents were PDFs so had specific pages. Chunks or offsets could be used for non-PDF documents.

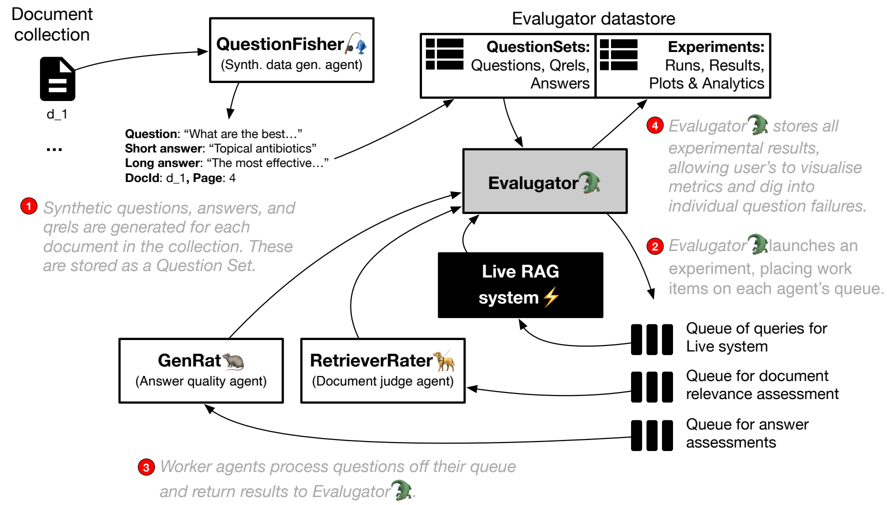

Fig. 1: Technical overview of Evaluagator and its associated agents.

3. The Live RAG system pulls questions off the queue and runs them. The results are placed on judging queues for RetrieverRater and GenRat, which run independently and in parallel to assess results. Multiple workers for each agent provide higher throughput. Queues are persisted so experiments gracefully resume if any part of the system goes down.
4. Experiment results appear in Evaluagator's datastore. This includes detailed results for each question as well as overall quality metrics (retrieval, answer quality and query latency). Developers can visualise these results via Evaluagator's simple UI or connect to the datastore via API to analyse results into their tool of choice.

Figure 2 shows experimental results in the Evaluagator UI. The table shows the list of experiments. The user has selected experiment `exp_002` and is provided with results for that experiment. Overall evaluation metrics (e.g., `Avg. correctness_score` and `Avg. clarity_score`) are shown, as well individual question metrics shown in plots. A table view (not shown) allows inspection of individual questions, answers, retrieved documents and metrics.

We employ two LLM judges within Evaluagator to assess retriever and generator effectiveness [1]. The Evaluagator's flexible architecture allows for integrating new judge agents; we already plan to add one for generating question-specific evaluation criteria. The RetrieverRater assesses a document using the user's question and both short and long ground truth answers. Different judging rubrics are provided but in general these assess if the document either fully or partially helps to answer the question. GenRat also uses question and ground truth answers but assesses the answer from the RAG system. Its judging rubric combines measures of correction / accuracy and measures of answer clarity. Both these agents can be used via API or as a simple web app (see Figure 3), encour-

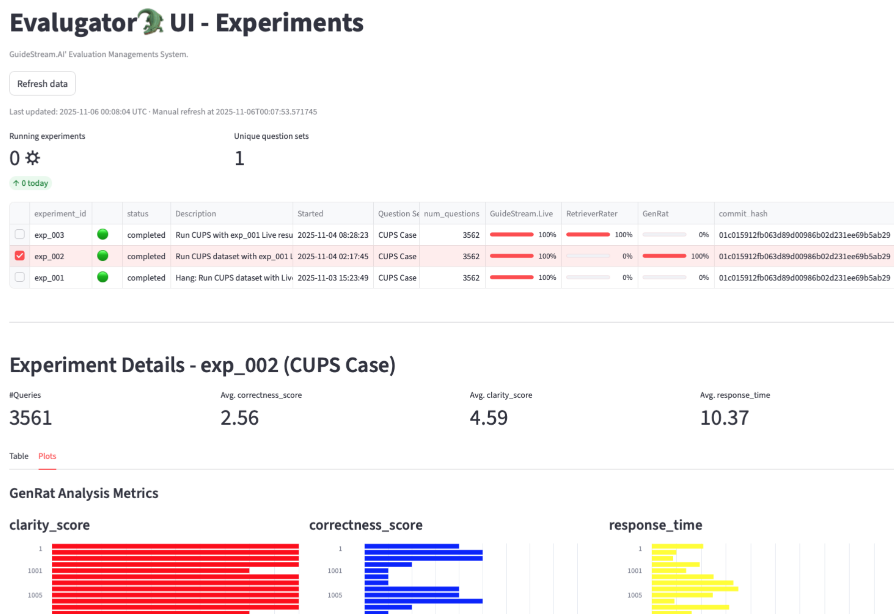

Fig. 2: Evalugator UI list of experiments and results for experiment "exp_002".

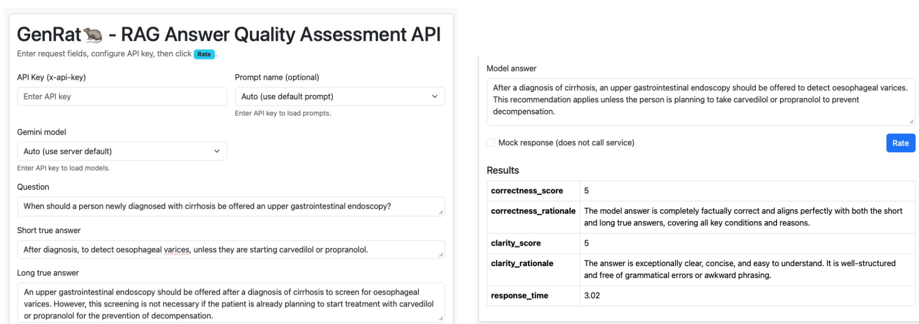

Fig. 3: GenRat provides judgements of the answers provided by a RAG system. While generally used via API, an simple UI also encourages develops to dig into individual questions to better analyse effectiveness.

aging developers to dig into the behaviour of individual questions. In addition to standard quality metrics, these agents also automatically categorise different failures types for each question (eg, hallucination vs no relevant documents retrieved).

3 Presenter Biography

Bevan Koopman is a co-founder of GuideStream.AI and an Associated Professor at The University of Queensland and CSIRO. His work focuses on applying information retrieval and natural language processing: making the myriad of health information more accessible to both clinicians and the public.

References

1. Balog, K., Metzler, D., Qin, Z.: Rankers, judges, and assistants: Towards understanding the interplay of LLMs in information retrieval evaluation. In: Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 3865–3875 (2025)
2. Krayko, N., Sidorov, I., Laputin, F., Panchenko, A., Galimzianova, D., Konovalov, V.: Rurage: Robust universal rag evaluator for fast and affordable qa performance testing. In: European Conference on Information Retrieval. pp. 135–145. Springer (2025)
3. Li, H., Dong, Q., Chen, J., Su, H., Zhou, Y., Ai, Q., Ye, Z., Liu, Y.: LLMs-as-judges: a comprehensive survey on LLM-based evaluation methods. arXiv preprint arXiv:2412.05579 (2024)
4. Yu, F., Seedat, N., Herrmannova, D., Schilder, F., Schwarz, J.R.: Beyond pointwise scores: Decomposed criteria-based evaluation of LLM responses. In: Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track. pp. 1931–1954 (2025)