Optimizing RAG: Classifying Queries for Dynamic Processing

Kabir Olawore[1] Michael McTear[1] Yaxin Bi[1] David Griol[2]

[1] School of Computing, Ulster University, UK
[2] Departamento de Lenguajes y Sistemas Informáticos, University of Granada, Spain
{olawore-b, mf.mctear, y.bi}@ulster.ac.uk, dgriol@ugr.es

Abstract

In Retrieval-Augmented Generation (RAG) systems efficient information retrieval is crucial for enhancing user experience and satisfaction, as response times and computational demands significantly impact performance. RAG can be unnecessarily resource-intensive for frequently asked questions (FAQs) and simple questions. In this paper we introduce an approach in which we categorize user questions into simple queries that do not require RAG processing. Evaluation results show that our proposal reduces latency and improves response efficiency compared to systems relying solely on RAG.

1 Introduction

Since the launch of ChatGPT in November 2022, conversational systems powered by Large Language Models (LLMs) have gained widespread adoption, allowing users to ask questions with the expectation of receiving accurate, factual answers (McTear and Ashurkina 2024; Mohamadi et al. 2023; Skjuve et al. 2024).

However, the responses of LLMs are not always accurate or even up-to-date. Although LLMs are trained on vast datasets, they may lack access to domain specific information, such as data from a company's internal database. The knowledge of an LLM is limited to the training data's cut-off date, resulting in potential obsolescence. Moreover, unlike traditional retrieval systems, where knowledge is stored explicitly in structures such as knowledge graphs, LLMs encode knowledge implicitly within their model parameters, making information retrieval less transparent and potentially less reliable (Yang et al. 2024; Zhu et al. 2024).

Retrieval-Augmented Generation (RAG) has been developed as a method to address these limitations by combining the generative capabilities of LLMs with real-time information retrieval from external sources (Lewis et al. 2021). In RAG, external documents are embedded into vector representations and stored in a specialized vector database. When a user submits a query, it is similarly vectorized and used to retrieve relevant documents. These documents are then integrated with the query and sent to the LLM for inference, ensuring that the generated response is based exclusively on the retrieved information. This hybrid approach enables RAG systems to deliver accurate, up-to-date, and context-specific answers (Gao et al. 2023; Huang and Huang, 2024).

The effectiveness of RAG systems has been demonstrated across various domains. Kharitonova et al. (2024) evaluated a RAG-based question-answering system for mental health support by embedding documents containing clinical practice guidelines. Their results highlighted the system's ability to deliver answers that were coherent, accurate, and supported by scientific evidence. Similarly, Olawore et al. (2025) described a RAG-based system designed to provide information about university fees, departments, facilities and other administrative details. Their findings showed that the system retrieved relevant and accurate information more effectively than standalone LLMs. Furthermore, the system enabled transparency and accountability by allowing users to trace each response back to its original source within the university dataset.

One significant drawback of RAG is that it is computationally expensive, particularly at the retrieval and inference stages. Processing frequently asked questions through the entire RAG workflow is both inefficient and costly. A more effective approach involves using a semantic cache, capable of handling variations and paraphrases of queries while returning consistent responses. On receiving a new request, the system first checks if a similar request has been processed previously. If so, it retrieves the stored response from the cache, bypassing the need to re-execute the complete RAG workflow (Alake et al. 2024; Mortro 2025; Siriwardhana et al. 2023). This approach reduces redundant computations and can also minimize end-to-end latency. For instance, Jin et al. (2024) introduced and evaluated a cache-based system called RAGCache across various models and workloads, demonstrating a 4x reduction in time to first token generation.

Zhao et al. (2024) proposed a four-level query classification system based on data requirements and reasoning complexity, encompassing explicit fact queries, implicit fact queries, and interpretable and hidden rationale queries. They introduce different methods for integrating external data with queries at each classification level. Explicit fact queries can be answered directly using the provided data, while the other types of queries require additional processing and access to external resources.

In this paper, we argue that explicit fact queries can be treated similarly to frequently asked questions, thereby bypassing the RAG workflow. On receiving a new query, the system first determines if the query has been asked before. If it has, the stored response is retrieved. If not, the system checks whether the query qualifies as an explicit fact query and retrieves the corresponding answer. For other query types, the RAG workflow is invoked. Additionally, queries of any type that are asked and resolved a certain number of times can also be added to the semantic cache for frequently asked questions. Our approach significantly reduces computational costs and latency in question-answering systems. In the following sections, we present a preliminary investigation into these concepts, offer experimental results addressing latency reduction, and conclude with recommendations for future work.

2 Methodology

Our proposal approach enhances RAG chatbot capabilities through question classification and a routing mechanism, optimized to process queries of varying complexity. The main objective is to significantly improve computational efficiency and latency compared to traditional RAG-based conversational agents that uniformly process all queries through the entire pipeline. At the core of this system lies a classifier that determines whether to bypass the retrieval stage for straightforward queries or engage the full retrieval pipeline for complex questions requiring additional factual support.

As Figure 1 shows, the proposed hybrid architecture comprises three distinct stages: classification, retrieval, and generation. In the classification stage, incoming queries are analysed to determine their complexity and information requirements. Simple queries that can be addressed directly proceed immediately to the generation stage, while complex queries that necessitate additional context are routed through the retrieval pipeline. This selective engagement of the retrieval mechanism represents a key optimization in our design, substantially reducing the computational overhead associated with unnecessary document retrieval and processing.

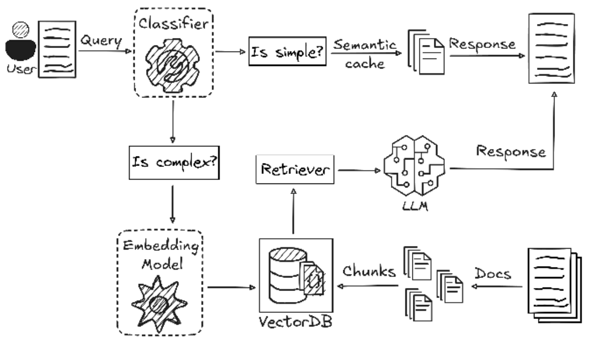

Figure 1: Optimized RAG chatbot architecture, classifying queries as simple (predefined responses) or complex (retrieval and LLM-generated responses).

2.1 Data Preparation

The dataset is a curated dataset QP comprising predefined key-value pairs of 100 questions and answers together with an unstructured dataset for the RAG pipeline. The unstructured data contains information about courses at the Faculty of Computing, Engineering, and the Built Environment (CEBE) at Ulster University. The key-value pairs consists of questions and answers related to CEBE, which were generated using an LLM and manually selected based on their semantic simplicity and brevity, ensuring they address straightforward queries efficiently. The selection process employs metrics such as semantic complexity and query length to classify a question as "simple". Each pair in QP undergoes preprocessing to standardize formats and optimize retrieval:

$$QP_{processed} = Standardize(QP)$$

This standardized dataset serves as a lightweight response mechanism for simple queries, bypassing the computational overhead associated with RAG-based inference.

2.2 Question Classification Framework

Users interact with the system through a chatbot interface. During query processing, each incoming user query Q undergoes an initial complexity assessment to determine its appropriate response strategy. The classification mechanism evaluates Q across multiple dimensions, such as semantic complexity, query length, and contextual requirements.

$$C(Q) = Classify\ (Q \mid features)$$

A machine learning-based classification model, trained on an annotated dataset of questions using logistic regression, serves as the backbone of this routing system. The model differentiates between simple questions, which can be directly resolved using predefined answers, and complex questions that necessitate retrieval and generative reasoning steps. For simple queries, the predefined response is retrieved.

For complex queries, the model invokes the RAG pipeline to produce an informed response. This dual-response strategy reduces computational overhead by leveraging predefined answers when possible, while ensuring nuanced processing for more intricate queries. The dynamic classification and routing approach ensures optimal performance and adaptability in handling a diverse range of user queries.

2.3 State Management

To ensure optimal system performance and mitigate latency across the hybrid architecture, the predefined key-value question-answer pairs are designed to enhance computational efficiency. When a query Q arrives, it is first transformed into a vectorized embedding (Q), which is stored in the system state:

$$E(Q) = Embed\ (Q)$$

The classification model processes (Q) to predict the query type, determining whether it aligns with predefined responses or requires retrieval-augmented generation (RAG). If the classifier identifies Q as likely resolvable via the predefined dataset, the system searches for a semantically similar question within the stored embeddings (D). The best-ranked candidate is retrieved and evaluated against a predefined similarity threshold τ:

$$Match(Q) = ArgMax\left(Similarity\big(E(Q), E(D)\big)\right) \geq \tau$$

If the similarity score satisfies the threshold, the corresponding predefined response is returned. Otherwise, the RAG pipeline is invoked to process the query. As more queries are processed, repeated patterns are identified and dynamically added to the predefined question-answer management system. This iterative process ensures continuous improvement of the system's predefined state, reducing the reliance on real-time retrieval for frequently encountered queries. By maintaining a balance between the predefined response mechanism and the RAG pipeline, the system sustains responsiveness and minimizes computational overhead.

2.4 RAG Pipeline

The RAG pipeline manages queries that are considered complex or infrequent. The RAG module is made up of two major components: the retriever and the generator. The retriever used a dense embedding model to locate relevant documents within the prepared corpus, ensuring that the most semantically similar content was selected. The generator on the other hand is powered by a transformer-based model, generating a coherent and contextually relevant response using the retrieved documents (Olawore et al. 2025).

2.5 Performance Evaluation

We have completed a preliminary evaluation comparing our RAG_HYBRID proposal with a RAG-only approach. We have used classic metrics such as accuracy, precision, recall, and F1-score to assess the relevance and precision of the chatbot's responses. These metrics provide a robust framework for evaluating the alignment of the chatbot's outputs with expected answers, ensuring the system's ability to deliver accurate and contextually appropriate responses.

To assess performance, we have measured latency. Latency was determined by recording the time elapsed between the submission of a query and the chatbot's final response. This analysis demonstrated the hybrid system's efficiency in reducing response times, highlighting its potential for improving user experience in real-time applications.

Additionally, the RAG_HYBRID's CPU utilization was evaluated and compared with the RAG-only solution. CPU usage was measured by logging the average processor consumption during query processing and the generation of the final response. This analysis provided insights into the computational efficiency of the hybrid architecture, emphasizing its ability to manage resource utilization while maintaining responsiveness.

3 Results

The evaluation result shows the effectiveness of the RAG-hybrid chatbot. Figure 2 shows the plot of the latency comparison between RAG-hybrid an RAG-only.

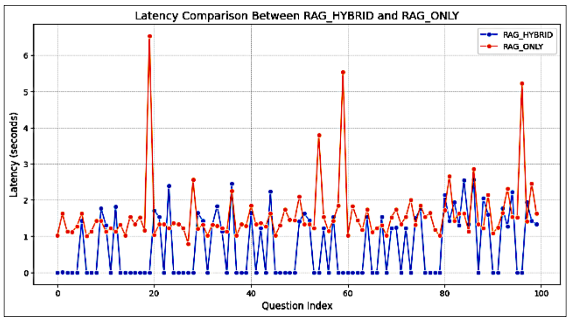

Figure. 2: Latency comparison between rag_hybrid and rag_only.

The plot illustrates a clear latency advantage of the RAG_HYBRID system over the RAG_ONLY system across 100 queries, consisting of 57 simple questions and 43 complex questions. RAG_HYBRID consistently demonstrates lower response times with minimal fluctuations, while RAG_ONLY exhibits significant spikes, exceeding 6 seconds for some queries. These results highlight the efficiency of the RAG_HYBRID system in leveraging predefined answers to maintain low latency and reduce computational overhead.

Also, in terms of processing needs, Figure 3 demonstrates a notable difference in CPU usage between the RAG_HYBRID and RAG_ONLY systems across the 100 test queries. RAG_HYBRID consistently exhibits lower CPU utilization, maintaining efficiency and avoiding significant spikes, while RAG_ONLY shows pronounced peaks, with usage exceeding 3.5% for certain queries. These results highlight the computational efficiency of the RAG_HYBRID approach, which leverages predefined answers to reduce the processing burden, compared to the RAG_ONLY system that relies on resource-intensive retrieval and generation processes. The edge cases in RAG_HYBRID are situations where the chatbot had to respond to users' queries with RAG.

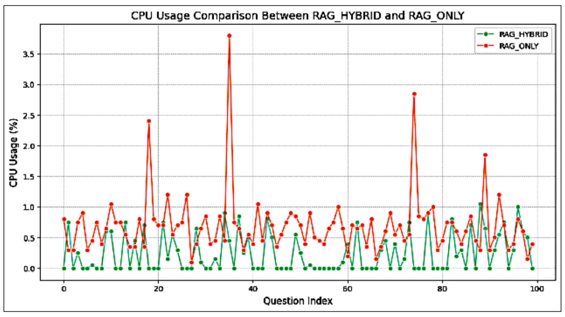

Figure. 3: Comparison of CPU usage between rag_hybrid and rag_only.

Finally, Table 1 shows that the RAG-hybrid chatbot achieves outstanding performance, with 98% accuracy and recall, a perfect precision of 1.00, and an F1-score of 0.99. These results highlight its reliability and effectiveness in delivering accurate and relevant responses.

	Accuracy	Precision	Recall	F1score
RAG-Hybrid	0.98	1.00	0.98	0.99

Table 1: Performance metrics of the RAG hybrid model.

4 Conclusions and Future Work

This study introduced a hybrid RAG chatbot architecture that efficiently combines predefined question-answer pairs with retrieval-augmented generation, demonstrating notable improvements in latency, CPU usage, and overall accuracy compared to RAG-only solutions. These results highlight the system's efficiency and scalability for real-time conversational AI.

Future efforts will focus on enhancing the classification model to adapt to evolving query patterns and integrating advanced language models to handle complex queries more effectively. We will also explore other methods of mitigating computational expense. An extended evaluation of our proposal in real-world scenarios and the incorporation of user experience metrics will also contribute to further evidence of its practical utility.

Acknowledgments

This paper has been supported by funding from the EU H2020 project CRYSTAL: https://project-crystal.eu/

References

Richmond Alake and Apoorva Joshi. 2024. Adding Semantic Caching and Memory to Your RAG Application Using MongoDB and LangChain. https://www.mongodb.com/developer/products/atlas/advanced-rag-langchain-mongodb/

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2024. Retrieval-augmented generation for large language models: A survey. https://doi.org/10.48550/arXiv.2312.10997

Yizheng Huang and Jimmy Huang. 2024. A Survey on Retrieval-Augmented Text Generation for Large Language Models. https://arxiv.org/abs/2404.10981

Chao Jin, Zili Zhang, Xuanlin Jiang, Fangyue Liu, Xin Liu, Xuanzhe Liu, and Xin Jin. 2024. RAGCache: Efficient Knowledge Caching for Retrieval-Augmented Generation. https://arxiv.org/abs/2404.12457

Ksenia Kharitonova, David Pérez-Fernández, Javier Gutiérrez-Hernando, Asier Gutiérrez-Fandiño, Zoraida Callejas, and David Griol. 2024. Incorporating evidence into mental health Q&A: a novel method to use generative language models for validated clinical content extraction, Behaviour & Information Technology. https://doi.org/10.1080/0144929X.2024.2321959

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. https://arxiv.org/abs/2005.11401

Michael McTear and Marina Ashurkina. 2024. Transforming Conversational AI: Exploring the Power of Large Language Models in Interactive Conversational Agents. Apress Berkeley, CA. https://doi.org/10.1007/979-8-8688-0110-5

Salman Mohamadi, Ghulam Mujtaba, Ngan Le, Gianfranco Doretto, and Donald A. Adjeroh. 2023. ChatGPT in the age of generative AI and large language models: a concise survey. https://arxiv.org/abs/2307.04251

Pere Mortro, 2025. Implementing semantic cache to improve a RAG system with FAISS. Hugging Face Open-Source AI Cookbook. https://huggingface.co/learn/cookbook/en/semantic_cache_chroma_vector_database

Kabir Olawore, Michael McTear, and Yaxin Bi. 2025. Development and Evaluation of a University Chatbot Using Deep Learning: A RAG-Based Approach. In: Asbjørn Følstad, Symeon Papadopoulos, Theo Araujo, Effie L.-C. Law, Ewa Luger, Sebastian Hobert, and Petter Bae Brandtzaeg (eds.) Chatbots and Human-Centered AI: 8th International Workshop, CONVERSATIONS 2024, Thessaloniki, Greece December 4-5, 2024, Revised Selected Papers. Springer Cham. https://doi.org/10.1007/978-3-031-88045-2

Shamane Siriwardhana, Rivindu Weerasekera, Elliott Wen, Tharindu Kaluarachchi, Rajib Rana, and Suranga Nanayakkara. 2023. Improving the Domain Adaptation of Retrieval Augmented Generation (RAG) Models for Open Domain Question Answering. Transactions of the Association for Computational Linguistics, 11:1–17. https://doi.org/ 10.1162/tacl_a_00530

Marita Skjuve, Petter Bae Brandtzaeg, and Asbjørn Følstad. 2024. Why do people use ChatGPT? Exploring user motivations for generative conversational AI. First Monday, 29(1). https://doi.org/10.5210/fm.v29i1. 13541

Jingfeng Yang, Hongye Jin, Ruixiang Tang, Xiaotian Han, Qizhang Feng, Haoming Jiang, Shaochen Zhong, Bing Yin, and Xia Hu. 2024. Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond. ACM Trans. Knowl. Discov. Data 18, 6, Article 160 (July 2024), 32 pages. https://doi.org/10.1145/3649506

Siyun Zhao, Yuqing Yang, Zilong Wang, Zhiyuan He, Luna K. Qiu, and Lili Qiu. 2024. Retrieval Augmented Generation (RAG) and Beyond: A Comprehensive Survey on How to make your LLMs use External Data More Wisely. https://arxiv.org/abs/2409.14924

Yutao Zhu, Huaying Yuan, Shuting Wang, Jiongnan Liu, Wenhan Liu, Chenlong Deng, Haonan Chen, Zhicheng Dou, and Ji-Rong Wen. 2024. Large Language Models for Information Retrieval: A Survey. https://arxiv.org/html/2308.07107v3