RestoRecommender

June, 2024

Hannane Habibi
hannane.habibi@studenti.unipd.it

Nila Akbari
nila.akbari@studenti.unipd.it

Mahshad Golafshan
mahshad.golafshan@studenti.unipd.it

Mehran Farajinegarestan
mehran.farajinegarestan@studenti.unipd.it

Shabnam Sattar
Shabnam.sattar@studenti.unipd.it

Abstract

This project aims to develop a Retrieval-Augmented Generation (RAG) based Large Language Model (LLM) system designed to integrate a restaurant review dataset to answer user queries with updated and relevant data. By using LangChain, a framework for building RAG systems, we will implement a system that enhances the capabilities of LLMs by incorporating external data retrieval mechanisms. The project will involve a comprehensive literature review on existing RAG systems, followed by the collection and preparation of a review dataset. A vector database will be created to store this dataset, facilitating efficient retrieval operations. The LLM will utilize the retrieved data as contextual input, thereby improving the quality of generated responses. The performance of the RAG system will be evaluated using established metrics, and an interface for user interaction will be developed as an additional goal. This project aims to demonstrate the effectiveness of RAG systems in improving LLM responses by providing contextually enriched information from a dedicated review dataset.

1. Introduction

Large language models (LLMs) have achieved remarkable success in natural language processing (NLP) tasks, yet they face significant limitations, particularly in domain-specific or knowledge-intensive scenarios [1]. A major challenge is the generation of "hallucinations" [2]—incorrect or fabricated information—when handling queries that extend beyond their training data or require current information. To address this, Retrieval-Augmented Generation (RAG) enhances LLMs by integrating external knowledge retrieval processes.

RAG employs semantic similarity calculations to retrieve relevant document chunks from external knowledge bases, providing LLMs with necessary contextual information to generate accurate responses as shown in Figure 1. This integration reduces hallucinations and improves factual correctness, making RAG a key technology for advancing chatbots and other LLM applications, enhancing their practicality in real-world scenarios.

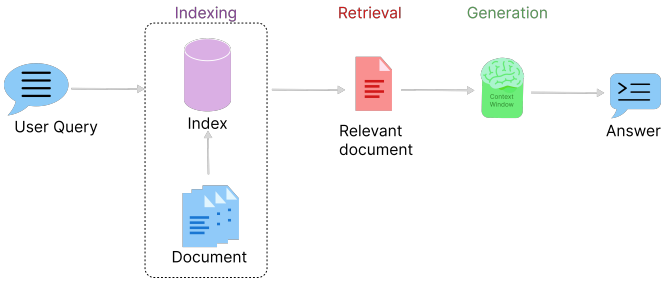

Figure 1: Visual representation of RAG system workflow

RAG technology has rapidly evolved, marked by distinct stages aligned with advancements in model architectures. Initially, RAG's development coincided with the rise of the Transformer architecture, focusing on incorporating external knowledge through pre-training models (PTMs) [3] [4] [5]. The advent of ChatGPT highlighted powerful in-context learning (ICL) capabilities [6], shifting RAG research towards optimizing information retrieval during inference to tackle more complex tasks. As research progressed, the focus expanded to include fine-tuning techniques, further enhancing RAG integration with LLMs.

A typical RAG application involves querying LLM about the latest news. Given LLM's reliance on pre-training data, it may initially lack the capacity to provide updates on recent developments. RAG bridges this gap by sourcing and incorporating relevant information from external databases, enabling LLMs to generate well-informed responses. [7]

RAG's interplay with other LLM optimization methods, such as fine-tuning (FT) [8] and prompt engineering, highlights its versatility and effectiveness. Prompt engineering leverages a model's inherent capabilities with minimal external knowledge, while FT allows deep customization of a model's behavior and style. RAG excels in dynamic environments by providing real-time knowledge updates, making it superior for scenarios requiring precise information retrieval. [9]

The efficiency of RAG depends on factors like retrieval sources, data granularity, and optimization of indexing and querying processes. Addressing these elements enhances the retrieval and generation phases, ensuring relevant and accurate responses. As RAG evolves, integrating fine-tuning and iterative retrieval strategies will further advance LLM capabilities, improving their handling of complex and dynamic information.

In this paper, we provide a comprehensive exploration of RAG technology. Section 2 reviews related work on RAG applications and recent methodological advancements. Section 3 discusses the core concepts and current paradigms of RAG. In Section 4, we outline our project methodologies and provide a detailed discussion of our approach. Section 5 presents our observations and experimental results. Section 6 explores potential future developments and enhancements for our model. Finally, Section 7 concludes the paper and summarizes our findings.

2. Related Works

The paper "Active Retrieval Augmented Generation" by Zhengbao Jiang et al. [10] introduces a novel framework to enhance Retrieval-Augmented Generation (RAG) systems called FLARE (Feedback Loop with Active REtrieval). Unlike traditional RAG models that use a static corpus, FLARE dynamically interacts with the retrieval process, allowing the generation model to iteratively refine its queries and retrieve more relevant information. This active learning approach enables the model to update its retrieval strategy based on evolving context and partial responses, significantly improving the coherence and accuracy of the generated text. Experimental results show that FLARE outperforms traditional static retrieval methods, setting a new standard for RAG systems. This advancement holds promise for applications needing precise and contextually nuanced text generation, such as conversational agents, question-answering systems, and personalized content creation.

The paper "Corrective Retrieval Augmented Generation" by Shi-Qi Yan et al. [11] introduces a novel framework designed to address inaccuracies in traditional RAG systems by incorporating a corrective retrieval mechanism. The authors propose an iterative process where initial retrieval results and generated responses are evaluated, and corrective queries are formulated to refine the retrieval of information. This approach aims to enhance the precision and relevance of the retrieved documents, thereby improving the quality and accuracy of the generated content. The corrective retrieval process involves dynamically adjusting queries based on feedback from intermediate outputs, ensuring a more adaptive and responsive retrieval generation cycle. Experimental results demonstrate that this method significantly reduces errors and increases the factual consistency of the generated text compared to existing RAG models. The paper highlights the potential of corrective retrieval to advance applications requiring high accuracy and reliability, such as detailed question-answering systems, expert advice generation, and critical information synthesis.

Another notable paper is "RQ-RAG: Learning to Refine Queries for Retrieval Augmented Generation" by Chi-Min Chan et al. [12], which introduces an innovative approach to enhance RAG systems by focusing on query refinement. The RQ-RAG framework leverages a learning-based mechanism to iteratively refine the queries used in the retrieval process, aiming to obtain more relevant and contextually appropriate information. This method addresses the limitation of static query strategies in traditional RAG models, which often fail to adapt to the nuances of complex queries and diverse information needs. By incorporating a query refinement process that dynamically adjusts based on feedback from intermediate retrieval results, RQ-RAG significantly improves the quality and relevance of the generated content. The authors demonstrate through extensive experiments that RQ-RAG outperforms existing RAG methods, highlighting its potential for applications in areas such as knowledge-intensive tasks, complex question-answering, and interactive dialogue systems.

3. Retrieval-Augmented Generation (RAG)

The Retrieval-Augmented Generation (RAG) research paradigm has evolved into three distinct stages: Naive RAG, Advanced RAG, and Modular RAG. Each stage was developed to address the shortcomings of its predecessor.

3.1. Naive RAG

The Naive RAG stage emerged with the rise of models like ChatGPT. It follows a "Retrieve-Read" framework that involves [7]:

1. **Indexing:** Cleaning and extracting raw data from formats like PDF, HTML, Word, and Markdown, converting it into plain text, segmenting it into smaller chunks, and encoding these chunks into vector representations using an embedding model. These vectors are stored in a database for efficient similarity searches.

2. **Retrieval:** Upon receiving a user query, the system encodes the query into a vector and computes similarity scores against the stored vectors. The top K similar chunks are retrieved.

3. **Generation:** The retrieved chunks are used as context to generate responses. The model combines the query and chunks into a prompt to formulate an answer, which may rely on the model's inherent knowledge or the retrieved context.

However, Naive RAG has several drawbacks:

- Retrieval Challenges: Struggles with precision and recall, often retrieving irrelevant or misaligned chunks.
- Generation Difficulties: Risks of hallucination, irrelevance, toxicity, or bias in generated responses.
- Augmentation Hurdles: Difficulty in integrating retrieved information with tasks, leading to redundancy or incoherence [7].

3.2. Advanced RAG

Advanced RAG aims to overcome the limitations of Naive RAG by focusing on:

1. **Enhanced Indexing:** Using techniques like sliding windows, fine-grained segmentation, and metadata incorporation to improve indexing quality [13].

2. **Pre-Retrieval Optimization:** Optimizing the indexing structure and refining the original query through methods like query rewriting and expansion [14] [15].

3. **Post-Retrieval Optimization:** Improving integration of retrieved context with the query, using re-ranking and context compression to ensure the most relevant information is highlighted [7].

3.3. Modular RAG

Modular RAG introduces even greater adaptability and versatility by integrating various specialized modules:

- Search Module: Adapts specific scenarios for direct searches across different data sources using LLM-generated queries [16].
- Memory Module: Leverages LLM memory to guide retrieval, creating an unbounded memory pool [17] [18].
- Routing Module: Selects optimal pathways for queries, navigating through diverse data sources [19].
- Predict Module: Reduces redundancy and noise by generating context directly through the LLM [20].
- Task Adapter Module: Tailors RAG to different downstream tasks, automating prompt retrieval and creating task-specific retrievers [21] [22]

Modular RAG also supports flexible module arrangements and interactions, allowing dynamic adjustments to improve retrieval relevance and task performance. Innovations such as the Rewrite-Retrieve-Read model and hybrid retrieval strategies enhance the system's ability to handle complex queries.

The progression from Naive RAG to Modular RAG reflects ongoing efforts to refine and enhance the efficiency, relevance, and accuracy of RAG systems. Each stage builds upon the previous one, incorporating new strategies and components to better address the complexities of information retrieval and generation in response to diverse user queries.

4. Methodology

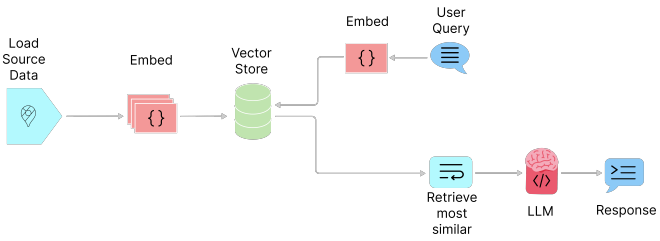

Figure 2: RestoRecommender RAG structure

In this project, a Retrieval-Augmented Generation (RAG) based Large Language Model (LLM) was developed, integrating a database of restaurant reviews obtained from the Google API to provide users with personalized restaurant recommendations and information. The primary components of this system are a vector database for efficient review retrieval and a large language model to generate coherent and contextually relevant responses. Figure 2 demonstrates our RAG system architecture.

4.1. Data Collection

Restaurant reviews and details were obtained using the Google Maps API, ensuring a rich dataset for providing recommendations. The dataset consists of 513 restaurants, bars and cafes in Padova. Overall, 2565 reviews were collected. With regards to each restaurant information such as the place name, address, opening hours, rating, etc. was gathered.

In the reviews dataset, other than the review details (the review itself, the date and time it was published and it's rating), a column that would link each review to the relevant restaurant was added.

4.2. Embedding Model

To embed restaurant reviews for efficient retrieval, Google's Text Embedding model is utilized via an API. This removes the need for a powerful system to run the code. The embeddings are cached locally to improve performance and reduce repeated computational overhead. The model selection process considered several criteria:

- Efficiency: The model should be lightweight for efficient use on various hardware. Google's Text Embedding model is accessible through an API, eliminating the need for a strong local CPU or GPU.
- Embedding Size: The model outputs 768-dimensional vectors, which is sufficient for our needs of capturing semantic relationships within reviews
- Input Length: The model can handle text up to 2,048 tokens, which is well-suited for most restaurant review lengths.

4.3. Setting Up the Vector Database

Reviews, and some other details about the restaurant, including place address, place name, phone number, etc. were indexed in a FAISS(Facebook AI Similarity Search) database, which is a highly efficient similarity search library, to facilitate quick retrieval based on user queries.

The Cosine and Euclidean distances were used to calculate the similarity between two embedded vectors. However, as embedding vectors were normalized no matter which distance was used, there was not any significant difference in the result.

4.4. Prompt Engineering

A custom prompt was designed to instruct the LLM on how to utilize the retrieved reviews to generate useful responses. The prompt was constructed using sample templates as seen on other RAG projects; The prompt also included the context which was retrieved in previous sections and the user query that was given as input. The prompt was:

"Your job is to use Google Map restaurants and bars reviews to help people find best places to go for a meal or a drink. Use the following information and reviews to answer the questions. if the context is not about restaurants, then kindly tell the user that you can only provide assistance and answer questions related to restaurants. If you don't know an answer based on the context, say you don't know."

4.5. LLM Integration

After the prompt was generated, it was given to the LLM, and the model then generated a response. Gemini 1.5 Flash was used as the LLM.

This model has a 1,048,576 token limit for the input and can generate up to 8,192 tokens. This amount was enough for the scenario in which it was used.

5. Results and Observations

The implementation of the RAG-based LLM yielded several observations. Below are the detailed results:

5.1. Effectiveness in Diverse Queries

The system can handle a variety of queries, ranging from general questions about the best restaurants to more specific inquiries about dietary preferences (e.g., vegan, gluten-free) and amenities (e.g., outdoor seating, live music). It provides comprehensive and detailed answers, often listing multiple options along with relevant details about each.

5.2. Handling of Unavailable Information

When information is not available, the system appropriately indicates this, such as in responses to questions about 24-hour restaurants or keto-friendly options. This transparency helps manage user expectations.

5.3. Handling of Unrelated Questions

When the user asks unrelated questions, the system will successfully remain in the scope of its duty and respond that it can only provide answers related to restaurants based on the provided reviews.

5.4. Limitations in Database

The system's effectiveness is limited by the scope and completeness of the review database it draws from. For instance, it couldn't provide information for certain locations like Via Roma or specific new restaurants beyond what was available in the reviews.

5.5. Response Specificity and Detail

The system provides detailed responses, often mentioning ratings and specific elements from the reviews (e.g., waitstaff, menu items), which adds depth to the answers.

It is adept at summarizing user sentiments from reviews, such as describing customer service or the dining experience.

5.6. Quality and Consistency of Reviews

If the reviews in the database are inconsistent in quality or detail, the system may struggle to generate consistently reliable and comprehensive responses. For instance, restaurants with fewer reviews or less detailed feedback might be underrepresented.

5.7. Handling Negative Reviews

The system tends to focus on positive aspects and may not adequately address negative feedback or mixed reviews, which could give users a skewed perception of the options.

5.8. Content Moderation Constraints

One significant limitation was observed when processing specific types of queries. For instance, when the query "What are some romantic restaurants that are suitable for date-night?" was submitted, the LLM did not generate any results. To understand this behavior, the LangSmith tool was utilized, revealing that the retriever component had indeed gathered appropriate context. Despite this, the LLM refused to produce a response.

To further investigate, the same query was resubmitted with a modified prompt format; however, the outcome remained unchanged, with no results generated. Subsequently, the question was posed directly to the LLM, bypassing the retrieval component, and it still failed to generate a response.

As the next step in the investigation, the word "romantic" was removed from the query, which was then resubmitted to the RAG system. This time, the system successfully generated an answer. This behavior indicated that the LLM was likely restricted from responding to any query containing the word "romantic," suggesting the presence of content moderation or filtering mechanisms within the LLM that prevent it from addressing certain types of content.

Overall, the RAG LLM system appears well-implemented for the task of retrieving and summarizing restaurant reviews, providing users with detailed, varied, and contextually relevant recommendations. However, its performance is inherently tied to the comprehensiveness of the review data it has access to. So, it is vital the database is constantly maintained with high-quality and up-to-date data.

Another deciding factor of the quality of the response, is the wording of the user query. The specific keywords mentioned in the question can affect the retrieval of reviews. To bypass this issue, it is proposed that the RAG system be changed to a multi-query RAG, in which the user query is first given as an input to an LLM, and as an output the model gives several queries with the same core context but with different wording. These queries are then given to the retrieval system and after retrieving all the reviews, a unique union is assembled and given to the prompt along with the original user query.

6. Future Works

To further enhance our RAG system, future work will focus on implementing and exploring additional evaluation metrics to comprehensively assess the model's performance. Currently, we evaluate the model by asking different types of questions to test its performance across various scenarios. Moving forward, we will incorporate user-centric evaluations, such as satisfaction surveys and real-world testing with end-users, to gather valuable feedback on the practical usability and relevance of our recommendations.

Additionally, we will explore A/B testing to compare different versions of the model, helping us fine-tune our approach. By integrating these advanced evaluation techniques, we aim to achieve a more holistic understanding of the model's capabilities and identify areas for improvement, ultimately enhancing its effectiveness in providing personalized and accurate restaurant recommendations.

7. Conclusion

This project demonstrates the efficacy of a Retrieval-Augmented Generation (RAG) system in enhancing the performance of Large Language Models (LLMs) by integrating a review dataset to provide up-to-date and contextually relevant responses. By leveraging the capabilities of LangChain, the project successfully implemented a system that combines retrieval mechanisms with generative models, thus addressing common issues such as hallucinations and the lack of current information in LLM outputs.

The developed system, tailored to handle restaurant reviews, utilized a vector database and Google's Text Embedding model to efficiently retrieve relevant information. The integration of this external data into the LLM's generative process significantly improved the quality of responses, especially in domain-specific scenarios like restaurant recommendations. This improvement was evident in the system's ability to handle diverse queries, provide detailed responses, and transparently communicate when information was unavailable.

Despite its success, the system's performance is intrinsically linked to the quality and completeness of the review database. The findings indicate that consistent and comprehensive data is crucial for maintaining high-quality outputs. Additionally, the project highlighted the importance of handling sensitive content moderation constraints within the LLM, as demonstrated by the system's inability to respond to queries containing certain keywords such as "romantic".

In conclusion, the RAG-based LLM system developed in this project showcases the potential of combining retrieval mechanisms with generative models to enhance response accuracy and relevance. This integration is pivotal for deploying LLMs in real-world applications where current and specific information is paramount. Further advancements in dataset quality, multi-query strategies, and content moderation will continue to push the boundaries of what RAG systems can achieve in improving LLM capabilities.

References

[1] N. Kandpal, H. Deng, A. Roberts, E. Wallace, and C. Raffel. "large language models struggle to learn long-tail knowledge". 2023. in International Conference on Machine Learning. PMLR, pp. 15 696– 15 707.

[2] Y. Zhang, Y. Li, L. Cui, D. Cai, L. Liu, T. Fu, X. Huang, E. Zhao, Y. Zhang, Y. Chen, and et al. Siren's song in the ai ocean: A survey on hallucination in large language models. 2023. arXiv preprint arXiv:2309.01219.

[3] D. Arora, A. Kini, S. R. Chowdhury, N. Natarajan, G. Sinha, and A. Sharma. Gar-meets-rag paradigm for zero-shot information retrieval. 2023. arXiv preprint arXiv:2310.20158.

[4] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. K¨uttler, M. Lewis, W.-t. Yih, T. Rockt¨aschel, and et al. Retrievalaugmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

[5] S. Borgeaud, A. Mensch, J. Hoffmann, T. Cai, E. Rutherford, K. Millican, G. B. Van Den Driessche, J.-B. Lespiau, B. Damoc, A. Clark , and et al. Improving language models by retrieving from trillions of tokens. *in International conference on machine learning. PMLR*, pages 2206–2240, 2022.

[6] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, and et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27 730–27 744, 2022.

[7] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. 2024. arXiv preprint arXiv:2312.10997v5.

[8] X. V. Lin, X. Chen, M. Chen, W. Shi, M. Lomeli, R. James, P. Rodriguez, J. Kahn, G. Szilvasy, M. Lewis , and et al. Radit: Retrievalaugmented dual instruction tuning. 2023. arXiv preprint arXiv:2310.01352.

[9] O. Ovadia, M. Brief, M. Mishaeli, and O. Elisha. Fine-tuning or retrieval? comparing knowledge injection in llms. 2023. arXiv preprint arXiv:2312.05934.

[10] Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. *Conference on Empirical Methods in Natural Language Processing*, page 7969–7992, 2023.

[11] Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. Corrective retrieval augmented generation. 2024. arXiv:2401.15884v2.

[12] Chi-Min Chan, Chunpu Xu, Ruibin Yuan, Hongyin Luo, Wei Xue, Yike Guo, and Jie Fu. Rq-rag: Learning to refine queries for retrieval augmented generation. 2024. arXiv:2404.00610.

[13] I. ILIN. Advanced rag techniques: an illustrated overview. 2023.

[14] X. Ma, Y. Gong, P. He, H. Zhao, and N. Duan. Query rewriting for retrieval-augmented large language models. 2023. arXiv preprint arXiv:2305.14283.

[15] W. Peng,, G. Li, Y. Jiang, Z. Wang, D. Ou, X. Zeng, E. Chen, and et al. Large language model based long-tail query rewriting in taobao search. 2023. arXiv preprint arXiv:2311.03758.

[16] X. Wang, Q. Yang, Y. Qiu, J. Liang, Q. He, Z. Gu, Y. Xiao, and W. Wang. Knowledgpt: Enhancing large language models with retrieval and storage access on knowledge bases. 2023. arXiv preprint arXiv:2308.11761.

[17] X. Cheng, D. Luo, X. Chen, L. Liu, D. Zhao, and R. Yan. Lift yourself up: Retrieval-augmented text generation with self-memory. 2023. arXiv preprint arXiv:2305.02437.

[18] S. Wang, Y. Xu, Y. Fang, Y. Liu, S. Sun, R. Xu, C. Zhu, and M. Zeng. Training data is more valuable than you think: A simple and effective method by retrieving from training data. 2022. arXiv preprint arXiv:2203.08773.

[19] X. Li, E. Nie, S. Liang, and D. Cai. From classification to generation: Insights into crosslingual retrieval augmented icl. 2023. arXiv preprint arXiv:22311.06595.

[20] W. Yu, D. Iter, S. Wang, M. Ju, S. Sanyal, C. Zhu, M. Zeng, and M. Jiang. Generate rather than retrieve: Large language models are strong context generators. 2022. arXiv preprint arXiv:2209.10063.

[21] Z. Dai, Y. Zhao, J. Ma, Y. Luan, J. Ni, J. Lu, A. Bakalov, K. Guu, K. B. Hall, and M.-W. Chang. Promptagator: Few-shot dense retrieval from 8 examples. 2022. arXiv preprint arXiv:2209.11755.

[22] Z. Sun, X. Wang, Y. Tay, Y. Yang, and D. Zhou. Recitation-augmented language models. 2022. arXiv preprint arXiv:2210.01296.

Table 1: Task Distribution

Name	Literature Review	Implementation	Interface	Report (+ Sections)	Presentation	Total
Nila Akbari	15h	13h	2h	15h (Sect. 4, 5)	5h	50h
Mehran Faraji	15h	30h	3h	5h(Sect. 4, 5)	5h	58h
Mahshad Golafshan	15h	15h	5h	7h(Sect. 0, 7)	5h	47h
Hannane Habibi	25h	10h	0h	15h(Sect.1,2)	5h	55h
Shabnam Sattar	15h	15h	2h	10h(Sect. 3, 6)	5h	47h