Document-Based Question Answering Using Retrieval-Augmented Generation with Open-Source Language Model

Ramkishor Pondreti[1], Yashwanth Yarabati[2], Chandini Panga[2], Deepthi Korla[2], Rahul Perla[2], Harish Barakala[2]

[1]*B. Tech. Students, Department of Computer Science & Engineering, Aditya Institute of Technology and Management, Tekkali-532201, India.*

[2]*Department of Computer Science & Engineering, Aditya Institute of Technology and Management, Tekkali-532201, India.*

Abstract: **Our project aims to address the issue of Extracting important information from a large collection of PDF documents by creating a smart system that combines two powerful methods: Large Language Models (LLMs) and Retrieval Augmented Generation (RAG). PDF files and other types of documents often hold vast amounts of text, such as user manuals, legal papers, academic articles, and technical guides. Manually searching through these documents to find specific answers can be very time-consuming and difficult. Our system begins with the user's question and then searches for relevant information from various external sources. By leveraging these external sources, RAG enhances the capabilities of pre-trained LLMs. While LLMs have revolutionized natural language processing, their responses are still limited to the data they were trained on. By adding external information, RAG can significantly improve the accuracy and relevance of LLM responses. This process enriches the language model's answers by combining the user's query with the most recent available data, ensuring that the responses are not only relevant and specific but also up-to-date and contextually accurate. This approach greatly enhances the quality of responses for a wide range of applications, from chatbots to information retrieval systems. By making use of RAG, our system can provide better, more accurate, and more contextually aware responses, addressing the challenge of extracting meaningful content from large collections of PDF documents and other texts.**

Keywords: **Retrieval Augmented Generation, Question Answering, Large Language Models, Information Retrieval**

1.INTRODUCTION

1.1 Large Language Models

A large language model (LLM) is typically an artificial intelligence system trained on a large amount of text data to understand and generate human-like language responses. LLMs are typically trained on large amounts of text data to be used for various natural language processing (NLP) tasks, such as text generation, machine translation, sentiment analysis, and question answering. LLMs employ deep learning architectures, particularly transformer networks, which are great at managing and interpreting sequential data. During training, these models learn to predict subsequent words in sentences, enabling them to grasp complex linguistic patterns, context, and nuances. This training equips LLMs to perform a range of NLP tasks.[1]

Despite their impressive capabilities, LLMs have some limitations because of the knowledge base. The LLM's knowledge is based on what it was exposed to during the training phase, which might not cover specific enterprise-related information, thus limits their response to such queries. As they are trained on the dataset that only covers information only till some point in time, LLMs often don't have access to latest information. These models are trained to be general-purpose and provide more generic answers, when we need to model to answer a question related a specific enterprise, it fails to provide an answer. To create a model, we need to train it from scratch with that enterprise specific data, which is even more troublesome.[2]

One of the solutions to make a pre-trained model answer questions related to our enterprise is to finetune the model with our enterprise specific data. Instead of training a new model from scratch for a specific purpose, fine-tuning allows you to build upon this pre-trained base model. In this process we train the model with our data by tuning the

parameters of the base model. Backpropagation algorithm adjusts internal parameters by back-propagating the errors, helping the model learn about the new training data. Although finetuning can train the model with domain specific data, it takes in a lot of training time, data and hardware requirements which a small or medium enterprise cannot afford. Additionally, in finetuning we alter the model parameters which may degrade the model's general performance and effectiveness. To make the model train again on new enterprise specific dataset, it must be retrained again, which is an expensive job.

As finetuning doesn't seem to be a great choice for small enterprises, we needed another way to train the model our own data. Instead of training the model we could just use In-Context Learning [3], which is a technique used in prompt engineering. In-context prompting is a technique where we provide a paragraph or any information as context and ask questions based on that contextual paragraph. Manually providing a contextual paragraph for each user query is cumbersome. Therefore, we require an architecture capable of being trained on extensive datasets to automatically retrieve relevant contextual paragraphs based on user queries, which can be fed to an LLM model for generation of final response.

1.2 Retrieval Augmented Generation

Retrieval-Augmented-Generation (RAG) architecture provides the service of retrieving such contextually relevant paragraphs from a large dataset without any need to provide the context every time a question is to be asked. RAG is an emerging technique in natural language processing that aims to enhance the capabilities of large language models by integrating retrieval mechanisms into the generative process. RAG retrieves relevant information from external knowledge sources, which is then used to inform the generation process.[4] This approach is particularly valuable in domains such as question answering, content generation, and knowledge-intensive NLP tasks. The primary objective of this project is to implement a RAG system that can improve the accuracy and relevance of generated content, particularly in areas where existing models may fall short. By leveraging both retrieval and generation, this project seeks to create a more robust and reliable NLP system.

2.LITERATURE SURVEY

The concept of Retrieval-Augmented Generation (RAG) has been extensively explored in recent literature, laying a strong foundation for its application in enhancing language models. Recent research underscores the significant potential of Retrieval-Augmented Generation (RAG) for enhancing large language models (LLMs), particularly for small and medium enterprises (SMEs) looking to train these models on their proprietary data. Traditional instruction tuning often falls short in specialized domains, as highlighted by Ghosh et al. (2024) , which reveals that such models, trained on broad datasets, struggle to provide the necessary depth for specific tasks and fail to generate enterprise specific responses. Tang et al. (2022) [5] discuss how fine-tuning can serve as a potential solution for enhancing natural language generation; however, they highlight that this approach may be prohibitively costly for small and medium enterprises. In contrast, RAG allows models to access verified external information that will be injected into this application, later retrieved from the store and augmented with the user query to provide the context needed from the LLM to operate on, thereby improving both the accuracy and reliability of outputs, as discussed by Huang and Yizheng and Zhao et al.

Additionally, Mao et al. (2020) illustrate RAG's effectiveness in open-domain question answering, making it a valuable tool for SMEs needing precise customer support. The quality of retrieved information remains crucial, with Salemi and Zamani (2024) [6] emphasizing the importance of robust evaluation methods to ensure relevance. Practical guidelines provided by VM et al. (2024) [7] further support SMEs in implementing tailored solutions, while Soudani et al. (2024) [8] point out ongoing challenges related to computational demands and data relevance of this system. Collectively, this body of work establishes a strong foundation for developing user-friendly RAG applications that empower SMEs to effectively harness LLMs for their enterprise specific needs.

3.METHODOLOGY

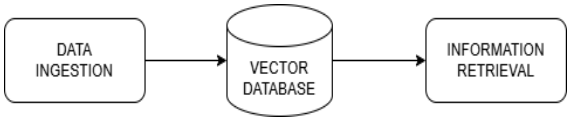

Figure 1: The Phases in RAG

This project employs a structured methodology comprising two primary phases: Data Ingestion and Information Retrieval as shown in the above figure. Each phase is designed to optimize the handling of textual data, ensuring that the large language model can generate accurate, contextually relevant responses. The Data Ingestion phase focuses on preparing the data, while the Information Retrieval phase emphasizes efficient and effective retrieval of relevant information. This section details the processes, models, and techniques employed in each phase, illustrating how they collectively contribute to the project's objectives.

3.1 Data Ingestion Phase

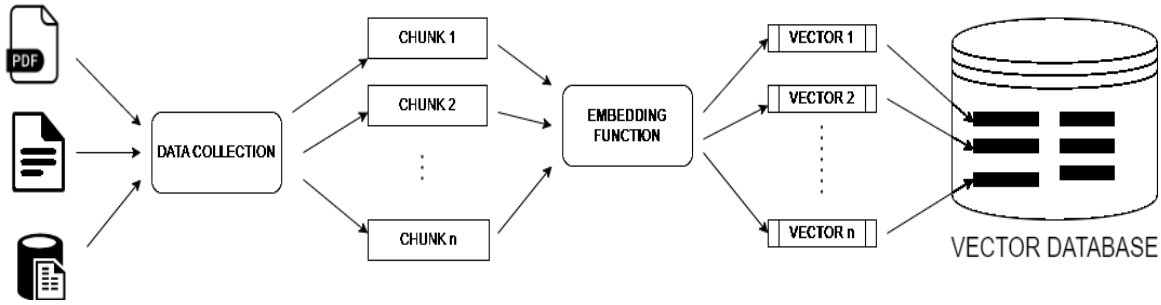

Figure 2: Steps involved in Data Ingestion Phase

3.1.1 Chunking of Documents

The above figure describes about data ingestion phase steps, the enterprise specific data is collected from different sources and compiled into a PDF file for processing. The data ingestion process begins with chunking, where large documents are divided into smaller, manageable pieces. This step is critical for facilitating efficient processing and retrieval, ensuring that even lengthy texts can be handled without performance issues. Each chunk represents a logical unit of the document, such as a paragraph or sentence, which allows for more precise and context-aware retrieval in subsequent phases. The context window of any open-sourced LLM is also limited to accept only a certain number of tokens (usually 1024 tokens), hence chunking of the document is necessary to design the augmented prompt.

3.1.2 Embedding Function

A. Frequency-Based Models:
Frequency-based models, such as TF-IDF, One-Hot Encoding, Count Vectorizer, and n-grams, are powerful techniques for text representation that capture word occurrence patterns in a corpus. TF-IDF (Term Frequency-Inverse Document Frequency) highlights important words by balancing their frequency within a document against their prevalence across all documents, making it useful for identifying key terms while downplaying common ones. One-Hot Encoding represents each unique word as a binary vector, indicating its presence or absence but ignoring frequency. Count Vectorizer builds on this by representing text as a matrix of token counts, capturing frequency while maintaining a sparse structure. N-grams, which group sequences of words, further enhance text representation by considering word combinations, thereby preserving context.

All these frequency-based models have several inherent limitations. A key drawback is their inability to capture semantic relationships between words, as they treat each word or phrase in isolation without considering meaning or context. The generated embeddings will have a sparse representation, which may slow down the training process.

B. Prediction-based models:
Prediction-based models capture the semantic and contextual relationships between words by predicting the likelihood of a word given its surrounding context. Word2Vec is a shallow neural network model that operates using two main approaches: Skip-Gram and Continuous Bag of Words (CBOW). The Skip-Gram model aims to predict the surrounding context words given a target word, making it effective for learning representations in smaller datasets or when context is more informative. On the other hand, CBOW predicts the target word based on its surrounding context words, typically performing better in larger datasets where context provides a stronger signal.

These models are very good at capturing word meaning and semantics, enabling them to understand complex relationships. For instance, words like

"king" and "man" will have similar vectors with subtle differences that reflect their association, while the word like "queen" is exactly opposite of "king", hence it has the vector representation in opposite direction. Word2Vec and related models also allow for word analogies, showcasing their capacity for deeper contextual understanding. Additionally, prediction-based models can generalize better across unseen data and are computationally efficient due to their low-dimensional, dense representations of the given input tokens.

3.2 Vector Databases

A vector database stores high-dimensional input tokens as embedding vectors, enabling efficient searching and retrieval. In RAG systems, it facilitates quick access to relevant information through similarity searches. Unlike relational databases, vector databases do not require a fixed schema, and data can only be accessed via similarity searches that return lists of similar vectors.

3.2.1 Representation of Vectors

Vector representations, such as those seen in the figure, are critical components in the structure of a vector database. These vectors, depicted as multi-dimensional points in space, are typically generated from text chunks using models like Word2Vec. Each token will be mapped to an embedding vector in this vector space where similar tokens are represented by similar vectors. Each vector encodes semantic relationships and contextual meaning, and when stored in a vector database for efficient querying and retrieval.

A vector database is specifically designed to handle these high-dimensional vectors, organizing them for fast similarity search operations. The proximity between vectors in the database, represented by the angles and distances in the diagram, is used to measure similarity between the query vector and stored vectors, often using metrics like cosine similarity. Indexing these vectors efficiently is crucial to enable fast searches. Popular indexing techniques, such as Approximate Nearest Neighbors (ANN), allow for quick retrieval by focusing the search on the most relevant sections of the vector space, rather than exhaustively comparing every vector.

The structure of a vector database must support not only fast search but also scalability, as the number of vectors and dimensions can grow significantly. Additionally, optimizing the storage of vectors and the efficiency of the indexing process ensures that the system can handle large volumes of data. This kind of structure is essential for applications in recommendation systems, information retrieval, and AI-driven search engines, where contextually relevant data needs to be accessed quickly based on vector similarities as shown in the below figure.

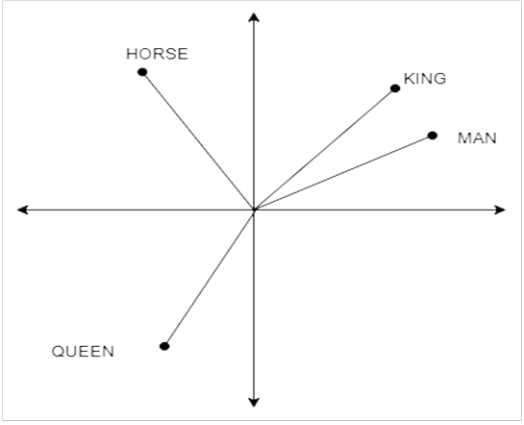

Figure 3: Vectors Representation

3.2.2 Similarity Search

Similarity search is an operation provided within the vector database that is central to the retrieval phase. When a query is presented, it gets converted into a vector using the same embedding function, then the system searches for vector representations in the database that are closest to the query vector, indicating a high degree of similarity. The speed and accuracy of this search process are vital for the overall performance of the system. Most vector databases use the cosine similarity for the similarity search operation as it offers better search results while being quick and easy to interpret.

Cosine similarity is a measure used to assess how similar two vectors (or sets of data) are. Instead of looking at their actual lengths, it focuses on the direction in which they point. Imagine vectors as arrows: if they point in the same direction, the cosine similarity is high (close to 1); if they're perpendicular, it's 0; and if they point in opposite directions, it's low (close to -1). The cosine similarity formula is:

$$\cos(A, B) = \frac{A.B}{||A||.||B||}$$

Given two embedding vectors, A and B, the cosine similarity between them is calculated using this formula. (A.B) represents the dot product (also known as the inner product) of vectors A and B. ||A|| and ||B|| denote the Euclidean norms of vectors A and B, respectively. The numerator measures how much the vectors align in the same direction. If they are perfectly aligned (pointing in the same direction), the dot product is maximized. The denominator, (||A||.||B||) normalizes the result by dividing the dot product by the product of the vector magnitudes. This ensures that the similarity score remains within the range of -1 to 1. If the vectors are identical, the cosine similarity is 1. If the vectors are orthogonal (perpendicular), the cosine similarity is 0. If the vectors point in opposite directions, the cosine similarity is -1.[9]

3.3 Information Retrieval Phase

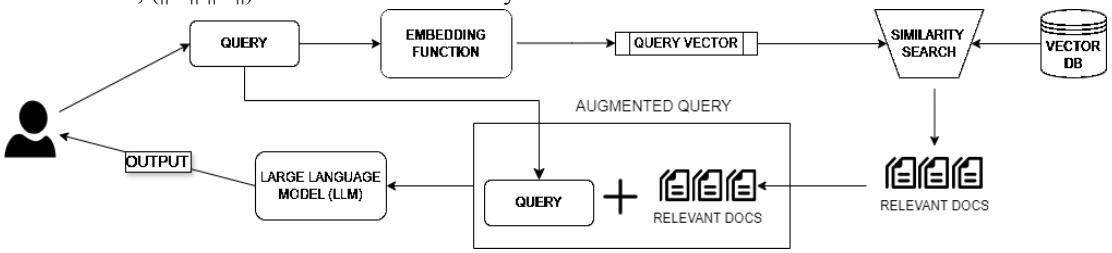

Figure 4: Steps involved in Information Retrieval Phase

3.3.1 Retrieval

In the above figure, the retrieval phase, the system retrieves relevant documents from the vector database in response to a user's query. This process begins with transforming the query into a vector representation using the same embedding function that was used in data ingestion phase. The vector database contains pre-processed chunks of text, each also represented as vectors they correspond. By calculating the cosine similarity between the query vector and these stored vectors, the system identifies the chunks that are most contextually similar to the query. This operation may return a lot of documents, we can control the number of relevant chunks we want by taking the Top K results from the set of chunks. The closer the cosine similarity score is to 1, the more relevant the chunk is to the query. This phase is critical as it ensures that only the most relevant information is retrieved, setting a strong foundation for the subsequent steps. The effectiveness of this retrieval process directly influences the quality of the generated responses, as the chosen chunks provide the context necessary for the next phase.[10]

3.3.2 Augmentation

Once relevant chunks of text are retrieved, the augmentation phase combines these chunks with the user's query to create a contextual prompt along with some instructions to the model on how to generate the output using the context. The retrieved chunks provide valuable information that is then combined with the query to form a augmented prompt for the language model. By using the context, the language model is now equipped to understand the query and generate responses that is both informative and contextually appropriate. The quality of this phase relies heavily on how well the retrieved information is integrated with the query, as it directly impacts the coherence and relevance of the final output.

3.3.3 Generation

In the generation phase, the augmented prompt is processed by a large language model (LLM)to produce a coherent and contextually relevant response. LLMs, such as GPT and LLaMA, are designed to understand and generate human-like text based on the context provided. They leverage extensive training on diverse datasets to understand language patterns, enabling them to generate text that aligns with the input they receive. The class of LLaMA models is open source, while the class of GPT models is closed source. During this phase, the language model uses the contextual prompt comprising both the query and the retrieved information to craft a response. The integration of retrieved chunks ensures that the model's output is specific, relevant data rather than relying on its pre-existing knowledge base. This approach enhances the response's accuracy and relevance, providing users with high-quality information tailored to their query. The effectiveness of the generation phase is dependent upon both the quality of the retrieval and augmentation phases, as they set the stage for the language model's response generation.[11,12].

4.RESULT

The implementation of the Retrieval-Augmented Generation (RAG) system for small-scale enterprises has yielded promising outcomes. This project aimed to provide these enterprises with a powerful tool, allowing them to leverage a language model trained specifically on their data, thus enabling them to generate responses to their clients that are more aligned with their unique needs and context.

4.1 Evaluation of the RAG System

The performance of some open-source models on Hugging Face was assessed using four key metrics: context precision, which measures the accuracy of relevant information retrieved; faithfulness, which verifies alignment with retrieved content; and response relevancy, which verifies that answers match questions.

Model	Context Precision	Context Recall	Response Relevancy	Faithfulness
Mistral-7B-Instruct-v0.2	0.65	0.68	0.62	0.60
Roberta-base-squad2	0.70	0.72	0.66	0.63
Meta-Llama-3.1-8B-Instruct	0.60	0.63	0.58	0.55
Falcon-7b-instruct	0.62	0.65	0.60	0.57

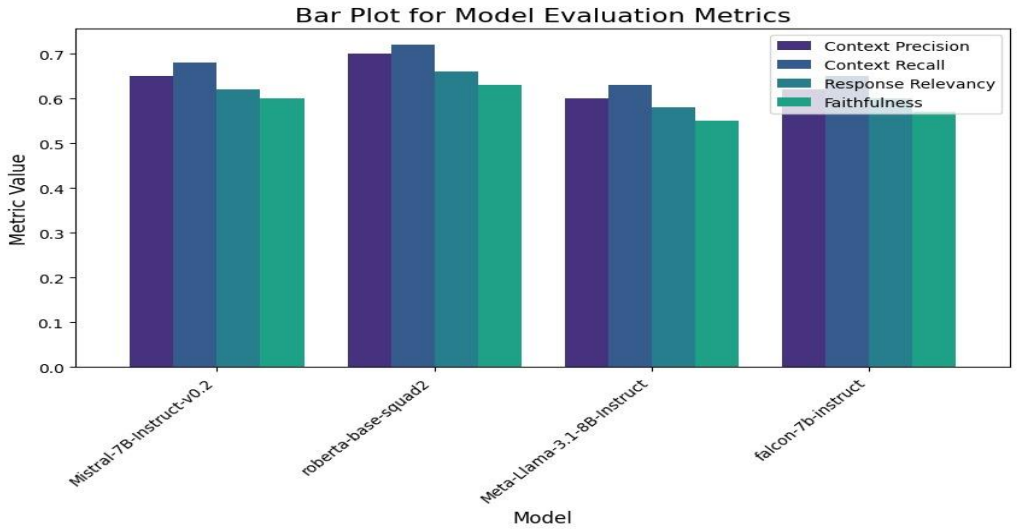

Figure 5: comparison of performance evaluation metrics for various models on testing sets

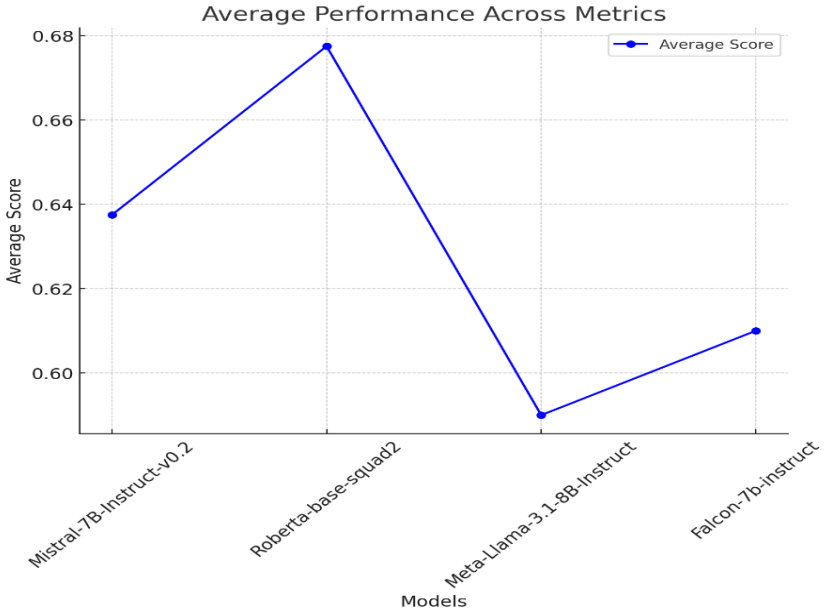

Figure 6: plotting the average performance of various models across metrics

The evaluation metrics highlighted the RAG model's ability in providing faithful and relevant information,by comparing performance evaluation metrices as shown in the above figures, the roberta-base-squad2 model demonstrates good performance and is well-suited for enterprise applications. However, there are many more models available that warrant exploration to further enhance the RAG system's capabilities.

4.2 Limitations and Challenges

Despite the substantial advantages of Retrieval-Augmented Generation (RAG) systems, several technical limitations and challenges persist. A critical challenge is the system's dependency on the quality and breadth of the retrieval database. The effectiveness of a RAG model is linked to its ability to retrieve relevant documents. If the database is incomplete, outdated, or lacks coverage of multimodal data (e.g., images, audio, video), the representation of multimodal data like images, videos and audio as vector could be challenging as it requires advanced techniques for data retrieval and management. Ensuring that the retrieval system can effectively handle and integrate multiple data types without degrading performance is a significant technical hurdle.

While these improvements can lead to more coherent and contextually accurate outputs, they also come with significant computational costs, increasing memory consumption and processing time. The trade-off between the quality of the generated content and the computational resources required is a key challenge in the design and optimization of RAG systems.

5.CONCLUSION

In conclusion, the implementation of the RAG system has shown considerable promise in enhancing language model performance for small-scale enterprises. This project has notably facilitated the custom training of models, allowing small enterprises to tailor AI solutions to their specific needs with greater ease. The system has improved accuracy and relevance, making it a valuable asset for businesses aiming to integrate AI into their operations effectively. However, the current implementation faces limitations in terms of multimodal capabilities and scalability. The lack of support for diverse data types and the constraints on expanding to larger datasets hinder the system's potential. Addressing these challenges through ongoing research and development will be crucial for unlocking the full benefits of the RAG system. With advancements in these areas, the RAG system could become a pivotal tool for small businesses, enabling them to remain competitive in an increasingly data-driven environment.

REFERENCES

[1] Naveed, Humza, et al. "A comprehensive overview of large language models (2023).

[2] Hadi, Muhammad Usman, et al. "A survey on large language models: Applications, challenges, limitations, and practical usage." (2023).

[3] Huang, Yizheng, and Jimmy Huang. "A Survey on Retrieval-Augmented Text Generation for Large Language Models (2024).

[4] Tang, Tianyi, et al. "Context-tuning: Learning contextualized prompts for natural language generation." (2022).

[5] Salemi, Alireza, and Hamed Zamani. "Evaluating Retrieval Quality in Retrieval-Augmented Generation." *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. (2024).

[6] VM, Kushala, Harikrishna Warrier, and Yogesh Gupta. "Fine Tuning LLM for Enterprise: Practical Guidelines and Recommendations." (2024).

[7] Soudani, Heydar, Evangelos Kanoulas, and Faegheh Hasibi. "Fine Tuning vs. Retrieval Augmented Generation for Less Popular Knowledge." (2024).

[8] Lakatos, Robert, et al. "Investigating the performance of Retrieval-Augmented Generation and fine-tuning for the development of AI-driven knowledge-based systems." (2024).

[9] Han, Yikun, Chunjiang Liu, and Pengfei Wang. "A comprehensive survey on vector database: Storage and retrieval technique, challenge." (2023).

[10] Rubin, Ohad, Jonathan Herzig, and Jonathan Berant. "Learning to retrieve prompts for in-context learning." (2021).

[11] Gao, Yunfan, et al. "Retrieval-augmented generation for large language models: A survey." (2023).

[12] Zhao, Penghao, et al. "Retrieval-augmented generation for ai-generated content: A survey." (2024).