

Question Answering through Retrieval-Augmented Large Language Models: an Experimental Evaluation

Claudia Diamantini¹, Alex Mircoli², Alessia Pagnotta², Domenico Potena¹,
Maria Cristina Recchioni² and Emanuele Storti¹

¹*Dipartimento di Ingegneria dell'Informazione, Università Politecnica delle Marche*

²*Dipartimento di Scienze Economiche e Sociali, Università Politecnica delle Marche*

Abstract

Large Language Models (LLMs) show impressive performance in many natural language processing (NLP) tasks, including code generation and question answering. However, they suffer from various limitations, such as the generation of hallucinated content and reliance on outdated internal knowledge. To address these challenges, Retrieval-Augmented Large Language Models (RA-LLMs) have emerged as a promising solution, enhancing response accuracy by dynamically incorporating external knowledge through retrieval-augmented generation (RAG). Despite the advantages of RA-LLMs, the implementation of effective retrieval-augmented pipelines remains a complex task, since various techniques exist for document chunking, text similarity evaluation, and model selection, each influencing the overall system performance. In this work, we propose a general methodology for designing a RA-LLM pipeline, outlining key approaches for each phase and evaluating the effectiveness of different configurations. We also perform an experimental evaluation of the pipeline's performance using real-world data, analyzing the impact of different techniques at each stage. The findings of this study offer insights into optimizing RA-LLM architectures for enhanced information retrieval and response generation.

1. Introduction

In recent years, Large Language Models (LLMs) have demonstrated impressive capabilities in understanding and generating text [1]. In the field of natural language processing (NLP), they are currently used in a wide range of applications, such as code generation [2], conversational AI [3], and emotion recognition [4], due to their ability to model complex semantic relations. However, they still struggle with certain limitations, including the tendency to produce hallucinated content [5] and the reliance on outdated internal knowledge. The latter problem is mainly a consequence of the huge training costs of these models, which limit the possibility of updating the model with fresh data. In this context, Retrieval-Augmented Large Language Models (RA-LLMs) have gained popularity due to their ability to generate accurate responses by considering information extracted from external knowledge sources through retrieval-augmented generation (RAG) [6]. This approach enables LLMs to access up-to-date and contextually relevant information, thus enhancing the accuracy and reliability of their responses by integrating internal knowledge with dynamically retrieved data.

Retrieval-augmented generation can be performed in several different ways, i.e. by defining different pipelines or by using different techniques to implement a step of the same pipeline. For example, many techniques have been proposed in the literature to split documents into smaller and more manageable text chunks. Likewise, various approaches for evaluating text similarity during the retrieval phase can be adopted. Moreover, many LLMs are available online, each with different characteristics and performance. As a consequence, effectively combining the existing approaches and tools can be challenging. However, the appropriate choice of techniques to be used at each stage of the pipeline could significantly increase the performance of the system in terms of relevance of the retrieved documents and quality of the generated response, as demonstrated by several works (e.g., [7]). The present work aims to propose a general methodology for RA-LLM and outline the possible approaches that can be used for each step of the pipeline, as well as to evaluate the effectiveness of different combinations of techniques.

ITADATA2025: The 4th Italian Conference on Big Data and Data Science, September 9–11, 2025, Turin, Italy

✉ c.diamantini@univpm.it (C. Diamantini); a.mircoli@univpm.it (A. Mircoli); S1112907@studenti.univpm.it (A. Pagnotta); d.potena@univpm.it (D. Potena); c.recchioni@univpm.it (M. C. Recchioni); e.storti@univpm.it (E. Storti)



© 2025 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

The main contributions of the paper are:

- the definition of a pipeline for RA-LLM to generate relevant and complete answers to questions asked by users by retrieving information from an external knowledge base;
- the experimental evaluation of the pipeline performance by means of real-world data and, for each phase of the pipeline, of the impact of the use of a specific technique, chosen among those most commonly adopted in the literature.

The rest of the paper is structured as follows: the next Section presents some relevant related works on Large Language Models and Retrieval-Augmented Generation. The methodology for the creation of the RA-LLM is proposed in Section 3, while Section 4 discusses the results of an experimental evaluation of the three phases of the methodology. Finally, Section 5 draws conclusions and discusses future works.

2. Related work

Large Language Models (LLMs) have revolutionized the field of Natural Language Processing (NLP) by demonstrating remarkable capabilities in language understanding and text generation. The training process of an LLM involves exposing the model to vast amounts of textual data, enabling it to learn patterns, semantics, and contextual relationships. However, the paradigm presents key challenges: (1) once trained, the model’s knowledge remains static, reflecting only the data available up to that point, and (2) the LLMs are prone to hallucinations [8] with domain-specific queries, since training data primarily consist of general-purpose texts. Hence, given the expansive computational cost of training LLMs from scratch, or retraining them with new information, integrating custom knowledge into these models is a nontrivial challenge. Several strategies have been explored in the literature to address these issues, such as fine-tuning, i.e., training the LLM in a task-specific dataset to refine its knowledge [9, 10, 11, 12], prompt engineering, i.e., supporting the model by providing a structured prompt with detailed instructions for the task [13, 14, 15], and Retrieval-Augmented Generation (RAG), i.e., enriching the prompt with information extracted from external knowledge bases [16]. In particular, RAG is emerging as a leading solution, allowing for updates to the LLM’s knowledge base without the need to re-train the model. This is achieved by dynamically querying external knowledge sources, extracting relevant domain-specific information, and incorporating it into the model’s output. Simpler approaches to RAG consist in a pipeline starting with an indexing phase, in which documents are processed, segmented and embedded for storage in vector databases. A following stage involves retrieval, which can be sparse, i.e. word-based and applied mostly in text retrieval, or dense, embedding queries and external knowledge into vector spaces. The pipeline is concluded by a generation stage [17].

Within the taxonomy of foundational RAG approaches proposed in [18], *query-based* RAGs are among the most widely studied in the literature. These models aim to integrate the user’s query with retrieved information, which is then used for the initial stage of the generator’s input. As such, the combined content, including the original query and the retrieved information, is processed by the generator as a unified input, allowing to produce a contextually richer response. For instance, REALM [19] selects the top-k most relevant article snippets based on the query and passes each snippet along with the question to the LLMs to generate k responses. These responses are then merged to produce the final answer.

Lewis et al. [16] employed pre-trained parametric and non-parametric memory for language generation, using BART as the generator to significantly improve the generation process. In a similar way, In-Context RALM [20] leverages BM25 for document retrieval, and trains a predictive reranker which is aimed at reordering and integration of the top-ranked documents. In some cases, the retrieval process itself can be considered optional, as in [21], tailoring the behavior to diverse task requirements through self-reflection.

Beyond the query-answering RAGs, alternative RAG strategies include incorporating retrieved information into generative models as latent representations (e.g., FID [22]), or generative models integrating retrieval information through logits during the decoding process (e.g., kNN-LM [23]). Additionally, a further approach applicable to data sequences allows the use of retrieved information

directly as responses, instead of relying only on generation, aiming to save resources and accelerate response time (e.g., REST [24]).

Since performances of classical RAG architectures, in terms of relevance, precision and recall, vary considerably, more recent approaches introduce specific improvements focusing on enhancing retrieval quality and reducing latency through pre-retrieval and post-retrieval strategies. Among the former, improvements include query rewriting and transformation (e.g., [25]), while reranking, summarization and fusion are among the latter. Aspects which can be optimized during the indexing phase include the chunking strategy, possible enrichment of chunks with metadata, and definition of proper indexing structures. Chunk optimization methods are employed by [26], in which text chunks are broken down into finer atomic statements to achieve higher recall and improved results, while in RAPTOR [27], a tree structure is produced by recursive embedding, clustering, and summarization of text chunks to address the lack of contextual information. Since redundant information can interfere with the final generation of LLM, the retrieved content is typically processed. Among others, solutions include reranking, i.e. ordering of retrieved documents to highlight the most relevant ones [28], or compression of the context (e.g., [29]). On the other hand, alternative and effective methods to enhance the retrieval phase include Knowledge Graphs, where entities are connected by relations, serving as a pre-built index for retrieval [30].

3. Methodology

In this section, we describe the proposed methodology for building Retrieval-Augmented Large Language Models. The methodology is depicted in Figure 1 and consists of three main phases: *Indexing*, *Retrieval & Reranking*, and *Text Generation*. A detailed description of each methodological step is presented in the following subsections, along with the discussion of the main issues of each phase.

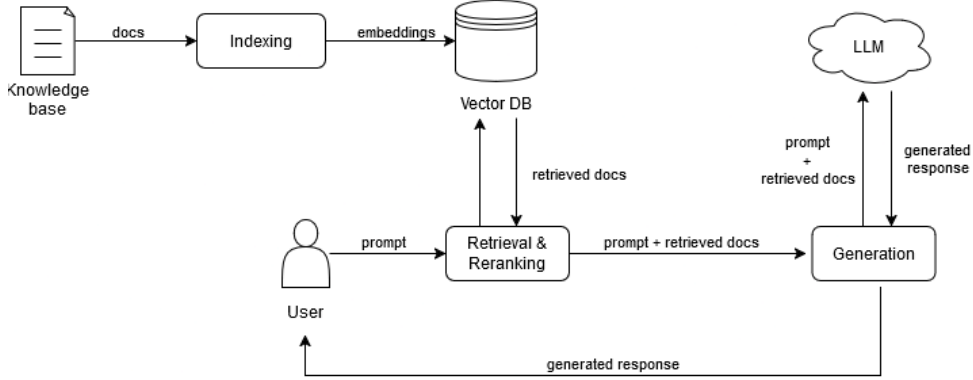


Figure 1: Description of the proposed methodology for RAG-based text generation.

3.1. Indexing

The goal of the *Indexing* phase is to process and store data belonging to an external knowledge base so that they are optimized for efficient data retrieval during the following phase (i.e., *Retrieval & Reranking*). The input of this phase is a set of text documents which contain information relating to the domain of interest, while the output is a set of sentence embeddings, along with metadata, stored in a vector database. The main steps of the *Indexing* phase are:

- chunking: texts are split into smaller chunks in order to store them in a more efficient way and allow searches to be performed at a higher level of granularity;
- embedding generation: text chunks are transformed into vector embeddings;

- storage: vector embeddings are stored and indexed into a vector database, which ensures optimized access to this type of data.

Among the existing approaches for chunking, the two most popular ones - which will be evaluated in section 4 - are the *recursive character chunker* (RCC) and the *semantic chunker* (SC). The RCC iteratively splits text into chunks, going through different levels of delimiters until it reaches a division that complies with the established criteria. In practice, the chunker divides the text into segments of a predetermined size, known as chunk size, with the help of delimiter characters. Although these characters can be customized, the default ones are: ["\n\n", "\n", " "], while the chunk size is determined by a number of characters chosen according to specific needs. The process starts by looking for the first delimiter (e.g., "\n\n"), which indicates the end of a paragraph. If the text between two "\n\n" is too long to fit within the chunk size limit, the system does not break the text, but moves on to the next delimiter, namely "\n", which separates sentences. If this division also does not respect the size limit, it tries again with the next delimiter (" " for words) until a segmentation that respects the size constraint is obtained.

The SC is based on the semantic similarity between groups of sentences, in order to create chunks that preserve the semantics of the text. It is particularly advantageous in the analysis of complex, long documents, such as technical manuals, scientific papers and reports, where a simple split based on static delimiters could compromise the overall understanding of the text. The SC splits the document into groups of three sentences, combining each sentence with the previous and the next one through an overlap mechanism. The groups of three sentences are then processed sequentially to generate vector embeddings representing each group. At this point, the cosine similarity between each current embedding and the next one is calculated and the groups of sentences are split when the cosine similarity is less than a given threshold.

For what concerns vector embedding, they are generated to improve the Retrieval phase, allowing a search based on semantic similarity between the considered texts (i.e., the user prompt and the documents in the knowledge base). Embeddings are usually generated using pre-trained Transformer models, such as Google BERT or OpenAI text embedding models. In the following, embeddings will be generated through OpenAI API¹.

3.2. Retrieval & Reranking

The *Retrieval & Reranking* phase aims at finding the most relevant set of texts from the knowledge base stored in the vector database. In this phase, user prompts are transformed into vector embeddings and are then compared to those in the vector database. Several different strategies can be used to determine the most relevant texts to be retrieved; among them, the most widely adopted are: semantic search, cosine similarity, and hybrid search. The first approach evaluates the similarity between two text embeddings by means of the Euclidean distance, while the second approach exploits the cosine similarity. The hybrid search combines keyword search and semantic search. For keyword search, it uses the Okapi BM25 ranking function, which is defined as:

$$BM25(q, d) = \sum_{i=1}^n \frac{IDF(q_i) \cdot f(q_i, d) \cdot (k_1 + 1)}{f(q_i, d) + k_1 \cdot (1 - b + b \cdot \frac{|d|}{avg_doclength})} \quad (1)$$

where:

- $f(q_i, d)$ is the frequency of term q_i in document d
- $|d|$ is the length of the document d
- $avg_doclength$ is the average document length in the corpus

The results can also be improved by coupling one of those retrieval strategies with a reranker. In fact, in many cases the initial retrieval phase may not identify the most relevant documents to the user's question. In these situations, the reranker allows to reorganize the selected candidates and filter out less relevant texts by applying further selection criteria.

¹<https://platform.openai.com/docs/guides/embeddings>

3.3. Text Generation

The goal of the last phase (*Text Generation*) is to generate a response expressed in natural language. The response is produced by giving as input to an LLM both the user prompt and the retrieved documents, which are used to extend the internal knowledge of the LLM through few-shot learning. Given the same number of documents provided as input, the quality of the output of this phase depends on the performance of the chosen LLM.

4. Experiments

This section presents an empirical analysis designed to evaluate the performance of the RAG-based system. The evaluations are conducted on a knowledge base consisting of English documents focused on Italian cuisine and nutrition. The section describes the generation process of queries and expected outputs, the evaluation metrics used to compare the three phases of the tested methodology, and a critical discussion of the obtained results.

4.1. Experimental setup

4.1.1. Knowledge base

To assess the performance of the RA-LLM system, raw data in English are sourced from various online platforms, including Wikipedia² and arXiv³. The text is extracted from the gathered documents, available in formats such as PDF and HTML, to build a dataset of 74 entries with an average length of 9,240 characters. The data focus mainly on Italian cuisine, including recipes from different Italian regions and the history of local culinary traditions. In addition, several scientific papers focus on nutrition issues, with a special emphasis on elderly nutrition and the Mediterranean diet.

4.1.2. Generation of queries and expected outputs

For a comprehensive evaluation of the system's ability to respond to user's prompts, 320 questions and answers are generated with the support of the GPT-4o-mini language model. These are organized into three levels of complexity:

- Simple Questions (100): aim to verify the knowledge of basic facts explicitly present in the documents.
- Intermediate Questions (102): require contextual understanding and the integration of multiple pieces of information.
- Multi-Document Questions (112): require in-depth analysis, synthesis skills, and critical thinking.

An excerpt of the questions generated by the LLM is shown in Table 1.

The questions are used to test the RAG processes and therefore must be designed to challenge the system effectively. For questions classified as "simple" and "intermediate", the process begins by generating 5 questions for shorter documents and 10 for longer ones. Then a manual evaluation is performed to select the most appropriate questions to achieve the desired number of questions and answers. Specifically, the final selection associates a single question with each short document and multiple questions with each long document. This approach ensures a balanced distribution of questions based on the length and complexity of the documents.

Finally, "multi-document" questions are generated by using multiple texts as reference context. Groups of three documents are selected sequentially, and the model is required to create a question that involves at least two documents in the group, preferably all three. For each triad, six questions are initially generated and then manually reviewed to identify the best ones. It is essential that each

²<https://it.wikipedia.org>

³<https://arxiv.org/>

Table 1

Example questions generated by the LLM.

Simple questions
<ul style="list-style-type: none"> - What is Arrabbiata sauce and how is it used in Italian cooking? - What are some common ingredients in Lucanian cuisine? - Tell me about the history of agnolotti and its origin
Intermediate questions
<ul style="list-style-type: none"> - What are the specific differences in filling and preparation between agnolotti di magro and agnolotti di grasso, and how do these distinctions influence the overall flavor profile of the dishes? - How did the introduction of new ingredients from the Americas in the 18th century impact the development and diversity of Italian cuisine? - What are the key regional variations in the preparation of timballo, particularly regarding the types of ingredients and methods used in different Italian regions?
Multi-document questions
<ul style="list-style-type: none"> - How do the ingredient choices and preparation techniques in Beef Braciola and Pasta Alla Norma reflect the culinary traditions and regional influences of southern Italy, particularly in the use of local produce and meats? - How does the preparation method of ossobuco, with its detailed cooking process involving vegetables and wine, compare to the traditional cooking techniques used in the preparation of cotechino with lentils, which also involves specific vegetable accompaniments and cooking methods? - How do the traditional cooking methods and ingredient combinations in Roman-style roasted lamb and tagliatelle al ragù reflect the regional culinary practices of Lazio and Bologna, particularly in terms of the significance of local ingredients and historical influences on these dishes?

question combines knowledge or insights from the selected documents and establishes meaningful connections between their contents.

4.1.3. Performance metrics

In this analysis, we evaluate the results through the approach offered by the `deepeval`⁴ library, which allows advanced metrics calculation using different LLMs. Therefore, all the evaluation metrics used are based on the *LLM-as-a-judge* paradigm. In our case, we use the OpenAI GPT-4o-mini model with a temperature value of zero.

The adopted approach allows examining the systems' performance in terms of contextual accuracy and relevance of responses. In particular, the considered metrics are:

1. Contextual Precision
2. Contextual Recall
3. Contextual Relevancy
4. Answer Relevancy
5. Faithfulness

The *Contextual Precision* measures how well the system prioritizes the most useful documents for a specific query. This metric is calculated as follows:

$$CP = \frac{1}{\text{Number of Relevant Nodes}} \sum_{k=1}^n \left(\frac{\text{Number of Relevant Nodes Up to Position } k}{k} \times r_k \right)$$

Where:

- k is the $(i + 1)$ -th chunk in the retrieval context.
- n is the length of the retrieval context.
- r_k is the binary relevance of the k -th chunk in the retrieval context. A value of $r_k = 1$ is assigned to relevant chunks, 0 otherwise.

⁴<https://github.com/confident-ai/deepeval>

The *Contextual Recall* evaluates the quality of the retrieval pipeline by measuring how much the retrieval context is aligned to the expected output. The indicator is defined by the following equation:

$$\text{Contextual Recall} = \frac{\text{Number of Attributable Statements}}{\text{Total Number of Statements}}$$

Specifically, the metric is determined using a language model to extract all statements in the expected output. The model then verifies how many of these claims can be assigned to the chunks present in the retrieval context, thus identifying the number of "attributable statements".

The *Contextual Relevancy* measures the quality of the retriever by evaluating the overall relevance of the information present in the retrieval context for a given input. The Contextual Relevancy score is determined by the following equation, using an approach similar to the previous metric:

$$\text{Contextual Relevancy} = \frac{\text{Number of Relevant Statements}}{\text{Total Number of Statements}}$$

The *Answer Relevancy* evaluates the quality of the response generation phase by measuring the degree of relevance of the actual output versus the input provided. The value of the Answer Relevancy is determined by the following equation:

$$\text{Answer Relevancy} = \frac{\text{Number of Relevant Statements}}{\text{Total Number of Statements}}$$

First, the calculation involves using a language model to extract statements from the actual output. Then, the model evaluates the relevance of each statement to the input, identifying the total number of "relevant statements".

Finally, the *Faithfulness* metric verifies how consistent the actual output is with the contents of the retrieval context. The value of the metric is defined by the following equation:

$$\text{Faithfulness} = \frac{\text{Number of Truthful Claims}}{\text{Total Number of Claims}}$$

As with the other metrics, a language model is used to extract all the statements in the actual output. Subsequently, the same model classifies an assertion as a "truthful claim" only if it is relevant to the retrieval context.

It is important to note that all three stages of the methodology, i.e. indexing, retrieval, and generation, will be tested. For the latter, the system is tested using the OpenAI GPT-4o-mini⁵ generative model and some open source alternatives, including Mistral-7B-Instruct-v0.2⁶, Mixtral-8x7B-Instruct v0.1⁷, and Llama-3.1-70B-Instruct⁸. In this way, it is also possible to assess the ability of open-source generative models.

4.2. Results

This section presents the results obtained during the evaluation of the proposed system based on RA-LLM. The main objective is to evaluate different variants with the purpose of determining the most effective approach in terms of generated responses. The outputs of the approaches are available at the following link: https://anonymous.4open.science/r/RAG_Text-CE8E.

4.2.1. Evaluation of the Indexing phase

For the indexing phase, three approaches are compared, namely the Naïve RAG, the Semantic Chunker and the OpenAI Large Embedding RAG. The characteristics of the three systems are shown in Table 2. The Naïve RAG implements the simplest approaches both for chunking and embedding and hence it is used as a baseline. The Semantic Chunker and the OpenAI Large Embedding RAG, instead, use more advanced techniques for, respectively, chunking and embedding.

⁵<https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/>

⁶<https://mistral.ai/news/announcing-mistral-7b/>

⁷<https://mistral.ai/news/mixtral-of-experts/>

⁸<https://www.llama.com/>

Table 2

Characteristics of the Naïve, Semantic Chunker, and OpenAI Large Embedding RAG.

System	Chunking strategy	Embedding
Naïve RAG	Recursive chunker	text-embedding-3-small
Semantic Chunker	Semantic chunker	text-embedding-3-small
OpenAI Large Embedding RAG	Recursive chunker	text-embedding-3-large

Table 3

Comparison among Naïve, Semantic Chunker, and OpenAI Large Embedding RAG.

Metrics	Naïve RAG	Semantic Chunker RAG	OpenAI Large Embedding RAG
ContextualPrecisionScore	0.826	0.878 (+6.28%)	0.884 (+7.02%)
ContextualRecallScore	0.931	0.960 (+3.09%)	0.958 (+2.86%)
ContextualRelevancyScore	0.266	0.256 (-3.68%)	0.276 (+3.69%)
AnswerRelevancyScore	0.826	0.865 (+4.65%)	0.921 (+11.51%)
FaithfulnessScore	0.954	0.946 (-0.84%)	0.942 (-1.25%)

As regards the indexing phase, the comparison between the text-embedding-3-large model and the text-embedding-3-small model seems to be interesting. This highlights improvements in key metrics (e.g., AnswerRelevancyScore) in favor of the large model. In fact, text-embedding-3-large allows better prioritization of the most relevant documents compared to the less significant ones, providing greater completeness in retrieving the relevant documents and giving more aligned responses with the initial input. In general, as also shown in the following comparisons, the adoption of text-embedding-3-large leads to better performance.

4.2.2. Evaluation of the Retrieval phase

For what concerns the Retrieval phase, the results of the experiments are summarized in table 4, highlighting the differences between the Naïve RAG, which uses the semantic search for the Retrieval phase, and the tested variants, namely the Cosine Similarity RAG, the Reranker RAG, and the Hybrid Search RAG.

Table 4

Comparison of Metrics among Naïve, Cosine Similarity, Reranker, and Hybrid Search RAG.

Metrics	Naïve RAG	Cosine Similarity	Reranker	Hybrid Search
ContextualPrecision	0.826	0.814 (-1.54%)	0.882 (+6.78%)	0.836 (+1.21%)
ContextualRecall	0.931	0.941 (+0.99%)	0.965 (+3.59%)	0.956 (+2.67%)
ContextualRelevancy	0.266	0.273 (+2.77%)	0.271 (+2.04%)	0.292 (+9.69%)
AnswerRelevancy	0.826	0.839 (+1.52%)	0.860 (+4.11%)	0.860 (+4.07%)
Faithfulness	0.954	0.952 (-0.19%)	0.947 (-0.72%)	0.941 (-1.36%)

The best approaches for the retrieval phase seem to be the Reranker and the Hybrid Search. In fact, the latter shows the highest Contextual Relevancy Score, equal to 0.292. This result is consistent with expectations, since the technique adopted excludes, among the selected chunks, those not strictly relevant to the query. This approach also allows a significant increase in the Answer Relevancy (+4.07%). However, there is a small reduction in the Faithfulness (-1.36%), which is the only metric where a reduction in performance is noted compared to the baseline.

It should also be noted that the use of cosine distance, compared to Euclidean distance, does not result in significant differences in the overall system performance.

4.2.3. Evaluation of the Generation phase

Regarding the generative phase, we carried out experiments on both GPT-4o-mini and open source models, namely Mistral 7B-Instruct-v0.2, Mixtral-8x7B-Instruct-v0.1 and Llama-3.1-70B-Instruct. Table

5 shows the average values of the metrics calculated for each considered model.

Table 5

Comparison of Metrics among GPT-4o-mini, Mistral 7B, Mistral 7x8B, and Llama 70B RAG.

Metrics	GPT-4o-mini	Mistral 7B RAG	Mistral 7x8B RAG	Llama 70B RAG
ContextualPrecisionScore	0.826	0.839 (+1.50%)	0.827 (+0.04%)	0.825 (-0.11%)
ContextualRecallScore	0.931	0.953 (+2.37%)	0.952 (+2.23%)	0.947 (+1.70%)
ContextualRelevancyScore	0.266	0.284 (+6.90%)	0.266 (-0.16%)	0.269 (+1.02%)
AnswerRelevancyScore	0.826	0.697 (-15.59%)	0.510 (-38.33%)	0.751 (-9.13%)
FaithfulnessScore	0.954	0.895 (-6.13%)	0.901 (-5.51%)	0.901 (-5.55%)

The results suggest that GPT-4o-mini outperforms the open source models. In particular, the model shows superior performance in identifying relevant information and generating complete, accurate answers. In contrast, the other models analyzed struggle with understanding the provided context or responding to complex queries, often returning incomplete or null responses. Furthermore, while it was expected that Mistral 7B would perform the worst among open-source models, as it is the model with the lowest number of parameters, it was actually Mistral 7x8B that recorded the lowest results in terms of response relevance to queries. In general, Llama 70B shows better performance than Mistral, although this is understandable considering that it is the open source model with the highest number of parameters among those tested.

5. Conclusion

In this work, we defined a pipeline for developing a question answering system based on Retrieval-Augmented Large Language Models and we experimentally evaluated several techniques for each of its building blocks. The analysis was conducted by using a real-world dataset on Italian food as the knowledge base and generating a set of questions at various levels of complexity. The generated responses were evaluated from different perspectives by means of five metrics. For what concerns the Retrieval phase, the experiments suggest that the use of a semantic chunker and a large embedding model lead to better results in terms of precision, recall and relevancy. As regards the Retrieval phase, the use of a reranker or the hybrid search seems to improve the relevancy of the generated responses, thanks to better retrieval of relevant information from the knowledge base. Finally, as regards the generation phase, the considered commercial LLM outperforms open source LLMs, in particular in terms of answer relevancy and faithfulness.

In future work, we plan to extend the experimentation by considering a larger set of documents in the knowledge base, from which it will be possible to define a greater number of questions. Moreover, we plan to generate the new dataset using a distinct LLM since questions and related answers are currently generated by one of the LLMs that are also evaluated in Section 4.2, which may potentially favor its performance by generating questions more similar to its own reasoning patterns, phrasing, or knowledge scope. We are also interested in evaluating other commercial and open-source LLMs, such as Anthropic Claude⁹ and DeepSeek [31]. Finally, we want to investigate the impact of some key parameters, such as chunk size and chunk overlap between different source types, since their optimization could potentially increase the performance of the Indexing and Retrieval phases.

Acknowledgments

Alex Mircoli and Maria Cristina Recchioni have received funding from the project Vitality – Project Code ECS00000041, CUP I33C22001330007 - funded under the National Recovery and Resilience Plan (NRRP), Mission 4 Component 2 Investment 1.5 - 'Creation and strengthening of innovation ecosystems,' construction of 'territorial leaders in R&D' – Innovation Ecosystems - Project 'Innovation, digitalization

⁹<https://www.anthropic.com/claude>

and sustainability for the diffused economy in Central Italy – VITALITY’ Call for tender No. 3277 of 30/12/2021, and Concession Decree No. 0001057.23-06-2022 of Italian Ministry of University funded by the European Union – NextGenerationEU.

Declaration on Generative AI

No GenAI tool was used during the preparation of this work.

References

- [1] S. K. Dam, C. S. Hong, Y. Qiao, C. Zhang, A complete survey on llm-based ai chatbots, arXiv preprint arXiv:2406.16937 (2024).
- [2] M. Nejjar, L. Zacharias, F. Stiehle, I. Weber, Llms for science: Usage for code generation and data analysis, *Journal of Software: Evolution and Process* (2023) e2723.
- [3] D. Banerjee, P. Singh, A. Avadhanam, S. Srivastava, Benchmarking llm powered chatbots: methods and metrics, arXiv preprint arXiv:2308.04624 (2023).
- [4] A. Chiorrini, C. Diamantini, A. Mircoli, D. Potena, E. Storti, Emotionalberto: Emotion recognition of italian social media texts through bert, in: 2022 26th International Conference on Pattern Recognition (ICPR), IEEE, 2022, pp. 1706–1711.
- [5] G. Perković, A. Drobnjak, I. Botički, Hallucinations in llms: Understanding and addressing challenges, in: 2024 47th MIPRO ICT and Electronics Convention (MIPRO), IEEE, 2024, pp. 2084–2088.
- [6] G. Izacard, P. Lewis, M. Lomeli, L. Hosseini, F. Petroni, T. Schick, J. Dwivedi-Yu, A. Joulin, S. Riedel, E. Grave, Atlas: Few-shot learning with retrieval augmented language models, *Journal of Machine Learning Research* 24 (2023) 1–43.
- [7] H. Yu, A. Gan, K. Zhang, S. Tong, Q. Liu, Z. Liu, Evaluation of retrieval-augmented generation: A survey, in: W. Zhu, H. Xiong, X. Cheng, L. Cui, Z. Dou, J. Dong, S. Pang, L. Wang, L. Kong, Z. Chen (Eds.), *Big Data*, Springer Nature Singapore, Singapore, 2025, pp. 102–120.
- [8] L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin, et al., A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions, *ACM Transactions on Information Systems* (2023).
- [9] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, LoRA: Low-rank adaptation of large language models, in: *International Conference on Learning Representations*, 2022. URL: <https://openreview.net/forum?id=nZeVKeeFYf9>.
- [10] O. Honovich, T. Scialom, O. Levy, T. Schick, Unnatural instructions: Tuning language models with (almost) no human labor, in: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2023, pp. 14409–14428.
- [11] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al., Training language models to follow instructions with human feedback, *Advances in neural information processing systems* 35 (2022) 27730–27744.
- [12] R. Rafailov, A. Sharma, E. Mitchell, C. D. Manning, S. Ermon, C. Finn, Direct preference optimization: Your language model is secretly a reward model, *Advances in Neural Information Processing Systems* 36 (2024).
- [13] M. Jia, L. Tang, B.-C. Chen, C. Cardie, S. Belongie, B. Hariharan, S.-N. Lim, Visual prompt tuning, in: *European Conference on Computer Vision*, Springer, 2022, pp. 709–727.
- [14] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al., Chain-of-thought prompting elicits reasoning in large language models, *Advances in neural information processing systems* 35 (2022) 24824–24837.
- [15] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, Y. Cao, React: Synergizing reasoning and acting in language models, in: *International Conference on Learning Representations (ICLR)*, 2023.

- [16] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al., Retrieval-augmented generation for knowledge-intensive nlp tasks, *Advances in Neural Information Processing Systems* 33 (2020) 9459–9474.
- [17] W. Fan, Y. Ding, L. Ning, S. Wang, H. Li, D. Yin, T.-S. Chua, Q. Li, A survey on rag meeting llms: Towards retrieval-augmented large language models, in: *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD '24*, Association for Computing Machinery, New York, NY, USA, 2024, p. 6491–6501. URL: <https://doi.org/10.1145/3637528.3671470>. doi:10.1145/3637528.3671470.
- [18] P. Zhao, H. Zhang, Q. Yu, Z. Wang, Y. Geng, F. Fu, L. Yang, W. Zhang, B. Cui, Retrieval-augmented generation for ai-generated content: A survey, *arXiv preprint arXiv:2402.19473* (2024).
- [19] K. Guu, K. Lee, Z. Tung, P. Pasupat, M.-W. Chang, Realm: retrieval-augmented language model pre-training, in: *Proceedings of the 37th International Conference on Machine Learning*, 2020, pp. 3929–3938.
- [20] O. Ram, Y. Levine, I. Dalmedigos, D. Muhlgay, A. Shashua, K. Leyton-Brown, Y. Shoham, In-context retrieval-augmented language models, *Transactions of the Association for Computational Linguistics* 11 (2023) 1316–1331.
- [21] A. Asai, Z. Wu, Y. Wang, A. Sil, H. Hajishirzi, Self-rag: Learning to retrieve, generate, and critique through self-reflection, in: *The Twelfth International Conference on Learning Representations*, 2023.
- [22] G. Izacard, E. Grave, Leveraging passage retrieval with generative models for open domain question answering, in: *EACL 2021-16th Conference of the European Chapter of the Association for Computational Linguistics*, Association for Computational Linguistics, 2021, pp. 874–880.
- [23] U. Khandelwal, O. Levy, D. Jurafsky, L. Zettlemoyer, M. Lewis, Generalization through memorization: Nearest neighbor language models, *arXiv preprint arXiv:1911.00172* (2019).
- [24] Z. He, Z. Zhong, T. Cai, J. Lee, D. He, Rest: Retrieval-based speculative decoding, in: *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 2024, pp. 1582–1595.
- [25] X. Ma, Y. Gong, P. He, H. Zhao, N. Duan, Query rewriting in retrieval-augmented large language models, in: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023, pp. 5303–5315.
- [26] V. Raina, M. Gales, Question-based retrieval using atomic units for enterprise rag, *arXiv preprint arXiv:2405.12363* (2024).
- [27] P. Sarthi, S. Abdullah, A. Tuli, S. Khanna, A. Goldie, C. D. Manning, Raptor: Recursive abstractive processing for tree-organized retrieval, in: *The Twelfth International Conference on Learning Representations*, 2024.
- [28] S. Zhuang, B. Liu, B. Koopman, G. Zuccon, Open-source large language models are strong zero-shot query likelihood models for document ranking, in: *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.
- [29] S. Hofstätter, J. Chen, K. Raman, H. Zamani, Fid-light: Efficient and effective retrieval-augmented text generation, in: *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2023, pp. 1437–1447.
- [30] B. Peng, Y. Zhu, Y. Liu, X. Bo, H. Shi, C. Hong, Y. Zhang, S. Tang, Graph retrieval-augmented generation: A survey, *arXiv preprint arXiv:2408.08921* (2024).
- [31] X. Bi, D. Chen, G. Chen, et al., Deepseek LLM: Scaling open-source language models with longtermism, 2024. URL: <https://arxiv.org/abs/2401.02954>. arXiv:2401.02954.