Information Retrieval Using Large Language Models on a Restricted Access Environment

Diego Costa, Gabriel Matos, Gilson Russo, Leon Barroso, and Erick Bezerra

SIDIA
Manaus - AM, Brazil

Abstract. Information retrieval plays an important role in everyday tasks, especially when it comes to documentation. Retrieving information about private documentation used to build other software is very challenging due to its absence on the internet, meaning there is no information about it beyond its own documentation. Due to concerns about confidential data, using external proprietary systems is prohibited. Motivated by this, in this study, we present Mycroft, a retrieval system that leverages the Retrieval Augmented Generation technique to find a feasible approach that improves search and information retrieval requested by users about the documentation. To implement this system, a dataset of questions and answers about the documentation was generated for evaluation. The system was developed on-premise using open-source Large Language Models and evaluated using Natural Language Processing metrics and human evaluation to validate the generated answers. Following an extensive evaluation of the results, the proposed retrieval system demonstrated satisfactory performance in addressing user queries and achieved favorable outcomes in human evaluation, indicating its utility.

Keywords: Retrieval Augmented Generation, Large Language Models, Software Documentation.

1 Introduction

This paper is an extension of work originally presented at the 11th International Conference on Computer Science, Engineering and Applications (CSEA 2025) [1]. Information Retrieval has developed an important role in day-to-day tasks. It has become part of daily routines for most people due to the need for rapid information and decision-making [2, 3]. It is present in many forms, such as internet browsing, virtual assistants, and chatbots [2]. With the advance of deep learning algorithms, information retrieval systems have been significantly improved [4], especially with the rise of Large Language Models (LLMs), which have been revealed as suitable for question-answering applications [5].

With the capacity of LLMs in question answering, it is possible to reduce both time and effort in information retrieval [4]. However, using cloud computing tools and commercial LLM raises security concerns due to the possibility of data leaks. As an alternative, open-source LLMs can be deployed on-premise server [4].

Large Language Models require prior knowledge acquired from training about a subject to answer a query. When unaware of the subject, it hallucinates [6]. An approach for building LLM applications without training them is the Retrieval-Augmented Generation (RAG) [4], which addresses this limitation by using an external source of information as context for response generation [7, 8].

RAG systems are considered a good choice for handling private information, as they leverage the LLMs' Natural Language Processing (NLP) capabilities without requiring the computational power needed to train an LLM model [4]. RAG systems are being implemented in various fields such as health, law, software, biology, and finance [9–13].

These systems are built on proprietary organizational documents that were not part of the LLM training. These documents are extensive and vital for private and governmental organizations, sometimes making it a time-consuming task to search and consult this information in daily routines [4].

DOI: 10.5121/ijnlc.2026.15202

In this work, Mycroft (a RAG system) is proposed to answer user questions about a private Software Development Kit (SDK) used internally in the organization. For this purpose, the documentation is used as a source of information by indexing it in a vector store database and retrieve it for the LLM context to answer the user's query.

This paper is organized as follows: Section 2 presents the background and related works, listing similar contributions and results achieved. Section 3 outlines the methodology, describing the approaches used to achieve the results. Section 4 reports the results, Section 5 presents the lessons learned and threats to validity, while Section 6 concludes by discussing the experimental outcomes.

2 Background

LLMs are trained with massive databases from different areas of knowledge, enabling them to answer a wide range of questions across various subjects [14]. However, when it comes to private data, they have a high probability of hallucinating in their responses [6], as they have never encountered such data. One of the commonly used approaches to address this issue it the RAG [7].

The RAG approach consists of a pipeline that first breaks down documents into small pieces, then indexes these pieces in a vector database. This database is consulted when a query about related documentation is made. The information relevant to the query is retrieved and passed to the LLM as context to answer the user's question [7]. To enhance the LLM response, various strategies are employed in the RAG pipeline. These include breaking down documents into different sizes, indexing information as metadata in the database, utilizing entity graphs, and leveraging other LLMs to retrieve more refined data for the LLM context. All these strategies are detailed in the Related Works subsection.

The RAG has been applied in various domain-specific areas and has demonstrated significant efficiency with adjustments in the pipeline for specific tasks in knowledge domains. Some authors modify the chunking strategy, the embedding model, the LLM's prompt, and the LLM model, as seen in the papers studied in the survey [15]. One advantage of RAG is that it is easier to keep the LLM model response up-to-date, as it only requires updating the external source of information. Consequently, the RAG system will use this updated version to query and answer queries [16].

To evaluate the generated answers against ground truth, we employed Bilingual Evaluation Understudy (BLEU) [17] and Recall-Oriented Understudy for Gisting Evaluation (ROUGE) [18]. Human evaluation assessed the proposed RAG answers by verifying their responses based on the method outlined in paper [11], focusing on Adequacy, Usefulness, and Relevance. These metrics are widely adopted in related works cited in this paper.

The BLEU metric evaluates n-grams matches between a generated and reference sentence [17]. N-grams are sequences of contiguous words. The longer the n-grams matched sequence, the better the score. While the BLEU metric is effective for evaluating word order and precision, it is limited in contextual and semantic understanding. Words with similar meanings but differing from the reference may reduce the score.

ROUGUE metric calculates the proportion of matched n-grams between the generated and reference sentences [18]. The score reflects the proportion of matched n-grams relative to reference n-grams. Unlike BLEU metric, ROUGUE emphasizes how much information from the reference appears in the generated sentence.

Additionally, we evaluated semantic similarity using the RAGAS[1] framework and BERTScore [19]. These metrics convert candidate and reference texts into numeric vector

[1] https://docs.ragas.io/

representations, measuring their semantic proximity using cosine similarity [20]. By capturing the semantic meaning of sentences, they ensure that different sentences with the same meaning are not penalized.

2.1 Related Works

Recently, different types of RAG architectures have been applied in various domains, aiming to achieve the best balance between answer generation and computational resources. For this purpose, small open-source LLMs with 1, 3, 7, and 8 billion parameters have been widely used across a variety of research fields, as described in the works [21, 10, 11, 4, 13] where different small open-source LLMs are employed to generate answers. These models differ in the specific domain knowledge applied by RAG and the pipeline used for information retrieval and answer generation. To achieve better results, various adjustments are made.

In the work [21] the authors used 12 different metrics to evaluate the LLM models' responses. Mistral:7B achieved the best score in 10 of them, although the authors used Mistral:7B to generate answers and questions for experiments, which may have biased the results.

The work [10] utilized Llama3.2:3B for chunk relevance checking and query refinement and employed Mixtral8x7B, Llama3.1:8B, and Gemma2:9B for answer generation. The authors compared the results using human evaluation and cosine similarity. Llama3.1:8B secured the best scores in both evaluation types.

The study [11] the LLM models Gemma2:2B, Llama3.2:1B and Phi3-mini:3.8B were utilized. The authors evaluated the context retrieval phase by calculating context recall using the RAGAS framework and an LLM as a judge, comparing retrieved chunks with the ground truth reference. The chosen LLM for this task was Llama3:8B, and to evaluate generated answers, they used Natural Language Processing metrics such as Bilingual Evaluation Understudy(BLEU), Recall-Oriented Understudy for Gisting Evaluation (ROUGE), and Metric for Evaluation of Translation with Explicit Ordering (METEOR). The best embedding model was stella_v5, and the best LLM for generating answers was Phi3-mini.

The work [4] proposed an entity tree, which is searched alongside relevant document retrieval from the database. If a user query mentions an entity, it is searched in the tree and added to the LLM context along with the retrieved documents. They experimented with the Llama2:7B base model and a fine-tuned version answered 21 questions correctly. However, the author also used Llama2:7B to generate questions and answers for validation.

The work [13] focused on how document chunks were split. They utilized a sentence transformer trained on over 256M questions and answers to split documents into sizes of 128, 256, and 512 tokens. Texts smaller than 2048 characters were concatenated until this limit was reached or a title or table was encountered in the document. They evaluated retrieval accuracy using ROUGE and BLEU metrics by comparing retrieved chunks with ground truth paragraphs. A chunk size of 512 yielded better results for answer generation, as it captures more context. However, in some cases, it adds irrelevant information to the context and misses important details.

3 Mycroft - Retrieval Augmented Generation for SDK Documentation

Coding scripts with an internally developed SDK tool is not always easy, even with its documentation, it can take some time to find answers to the questions that arise during the development phase.

In this regard, software engineers frequently consult technical documentation not only to create new scripts but also to maintain existing scripts with new features that are released and upgrade deprecations.

The software documentation referred here pertains to a private Python SDK hosted as web pages. It is organized into sections, each focusing on a class with a brief explanation of its purpose. Each class is divided into constants and methods. For every method, there is a section with a brief explanation, the function's parameters, and one or two code examples illustrating its usage.

The documentation includes two additional sections. The first section covers errors generated by the SDK, with descriptions for each error. The other section discusses the constants provided by the SDK, detailing their purposes and example code.

One concern is about confidential data [22], using external proprietary LLMs such as ChatGPT[2], Grok[3] and Claude[4] requires sending the data to external servers, which may facilitate data leaks [23]. As stated in the work [24], the data usage policy of closed-source LLM models allows them to utilize the provided data to retraining their models, making public the data that should remain private.

To mitigate this issue, open source LLMs on-premise can be used [25]. Another concern is about computational capacity, as LLMs increase in size they also demand more computational resources. Therefore, small models are more suitable for our resources.

The selected LLM models were open-source, with fewer than 8 billion parameters, which were the most recent available at the time. These models delivered reasonable performance in benchmark evaluations for coding, math, and general capabilities. The models included: qwen2.5:7b [26], qwen3:8b [27], llama3.1:8b [28], and mistral:7b [29]. Their lower hardware requirements allowed us to run them on a Dell PowerEdge R640 server with two Intel Xeon Gold 6154 CPU @ 3.00GHz and twelve 16GB 266MHz RAM.

The only exception among the small sized LLMs was the Qwen2.5:14B [26], which we used to assist in generating a question dataset for the experiments. This model also ran on the same server but was significantly slower than the small models due to its larger size. It was the largest model we were able to run due to our limited resources. The dataset generation process will be detailed in the following subsection.

This Section describes the methodology used to develop a RAG pipeline for retrieving information about a private Python SDK documentation used internally within an organization. It outlines the creation of a dataset for test experiments and the RAG approach, which includes documentation pre-processing, embedding generation, and evaluation metrics.

3.1 Dataset Creation

To validate RAG efficiency, a validation dataset is required. First, we extracted the documentation text from GitHub web pages. Therefore, we created a Python script to scrape the documentation by navigating recursively through each section until all documentation was saved in text files. Alongside each section, we included a header with the page URL and the corresponding class to ensure precise tracking and prevent retrieval errors. This process is illustrated in Figure 1.

After scrapping the documentation, it was divided into 2000-character chunks and passed to the LLM Qwen2.5:14B [26] with a prompt to generate relevant and concise

[2] https://openai.com/
[3] https://x.ai/grok
[4] https://claude.ai/

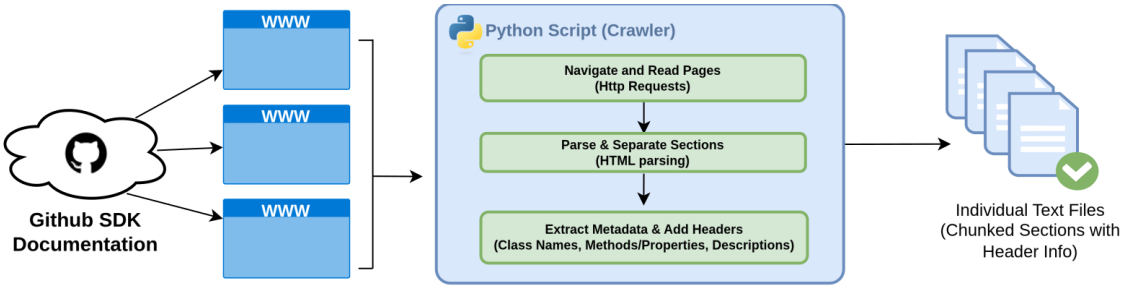

Fig. 1. Documentation Scrapper

questions with ground truth answers for each chunk. Care was taken to exclude the LLM used for generating the questions and answers dataset to avoid bias.

A human evaluation process assessed the correctness and relevance of the questions and answers. The team improved irrelevant or incorrect items by adding relevant details, ensuring more complete context for the LLM to regenerate the question and answer.

A second human evaluation followed, ensuring no errors remained and classifying the questions into three difficulty levels: easy, medium, and hard. Easy-level questions had straightforward answers in the documentation, medium-level questions required combining two different chunks for a complete answer, and hard-level questions needed more than two relevant chunks and involved combining retrieved information with user-provided details. By the end of this phase, 302 questions with verified ground truth answers were generated: 29 hard, 84 medium, and 189 easy.

3.2 RAG methodology

The objective of this study is to facilitate the process of retrieving information about SDK documentation to assist software engineers in their development and maintenance tasks. Therefore, it explored the combination of different techniques to extract relevant information and used 4 LLMs to generate answers for domain-specific questions. We defined 3 research questions:

- RQ1: Which combination of documentation chunking and embedding model contributed to the most accurate responses?
- RQ2: Which Large Language Model provided the best responses?
- RQ3: How good are the responses according to users' opinion?

Documentation chunking. To enhance context information for generative LLMs, two approaches for document chunking were tested. For both methods, we tracked headers created during the scraping phase and stored them with the content. The first step involved dividing and identifying text and Python code within each document section, then separating them into text and code sections.

One chunk approach used a text recursive splitter to break down the section's text into chunks of 500 characters with 100-character overlaps. Then, we used a Python recursive splitter with the same number of characters as the text chunks to segment the Python code, as illustrated in Figure 2.

The other approach involves directly using the documentation sections that contain the class, its description, and code as a single chunk. This method keeps the entire section's information together within one chunk, while another chunk contains information about a different section. This approach is illustrated in Figure 2.

Segmenting the documentation into chunks improves contextual relevancy and prevents the LLM context window from overflowing with large documents to process [30]. All chunks are less than 1300 tokens, and all LLMs used in the experiments have context windows up to 8k tokens. The objective of these two chunk approaches is to determine which type (smaller or large) has a greater impact on the LLM answer.

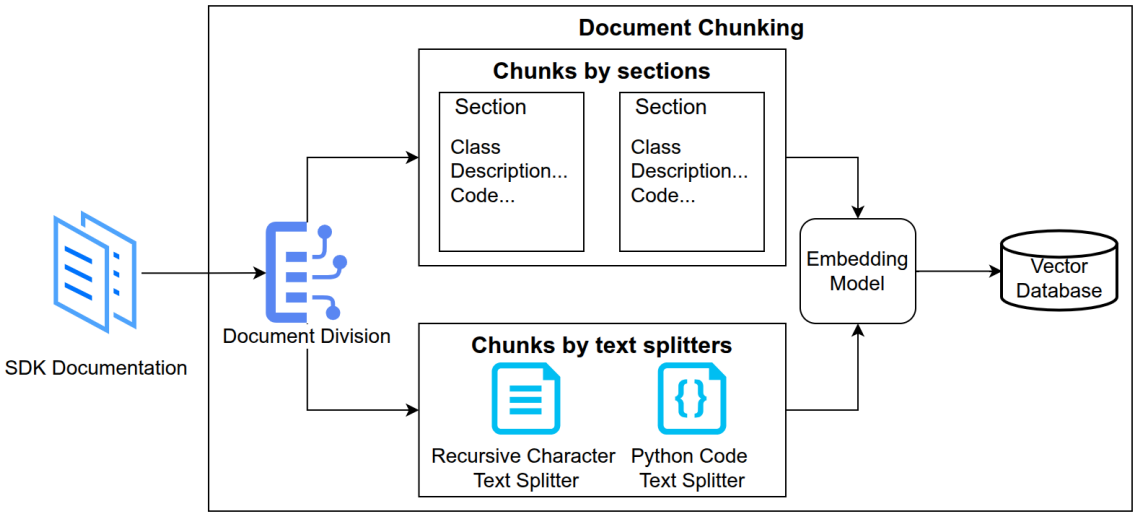

Fig. 2. Chunk strategies

Indexing. In this phase, the chunks extracted from the documentation undergo a process to transform them into numeric vector representations called vector embeddings, which are stored in a vector database [31]. Two embedding models, BGE-M3 [32] and nomic-embed-text-v1.5 [33], were selected to generate the vector embeddings for indexing in the Qdrant[5] vector database. The documentation chunking and indexing phases are executed only once to store the documentation in the database. Consequently, this information is consulted during information retrieval.

Retrieval After the documentation is divided into chunks and indexed by Qdrant, the retrieval process is ready to start receiving queries about the SDK documentation. Qdrant will convert the question into a vector embedding chunk using the same embedding model, then it will search for similar vectors in the database using semantic similarity. By default, the search will retrieve 4 documents that will be passed to the generative LLM, which will use these documents as context to formulate an answer for the user query, as shown in Figure 3.

Generation During the generation phase, the LLM is prompted to use the retrieved documents from Qdrant and to answer the user's query based on this domain-specific retrieved context as demonstrated in Figure 4. The prompt guides the LLM to act as an assistant providing explanations on how to use a Python SDK according to the retrieved context.

If the retrieved context is insufficient to answer the query, the LLM is instructed to state that it lacks information, with the aim of preventing hallucinations. At the end of the

[5] https://qdrant.tech/

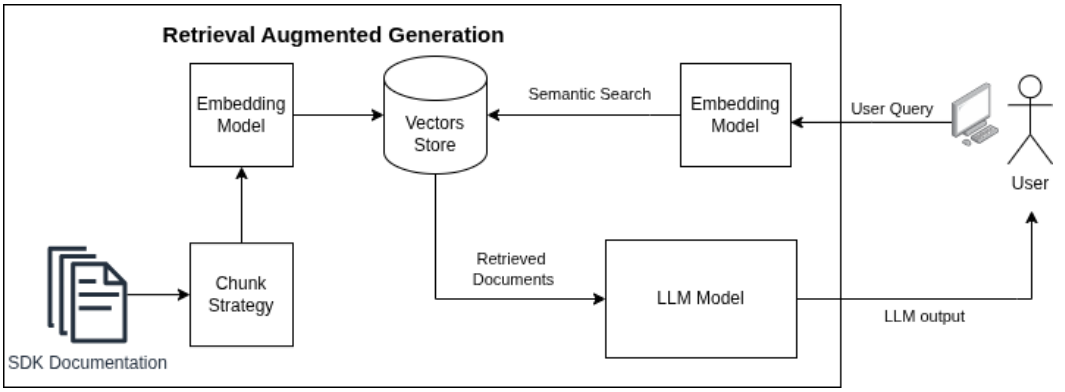

Fig. 3. Retrieval Augmented Generation - RAG

prompt, a retrieved context field is included, populated with documents from the Qdrant database, and a user query field. When the LLM is called, these fields are filled with actual data. Once the prompt template is populated with the retrieved context, the LLM infers an answer.

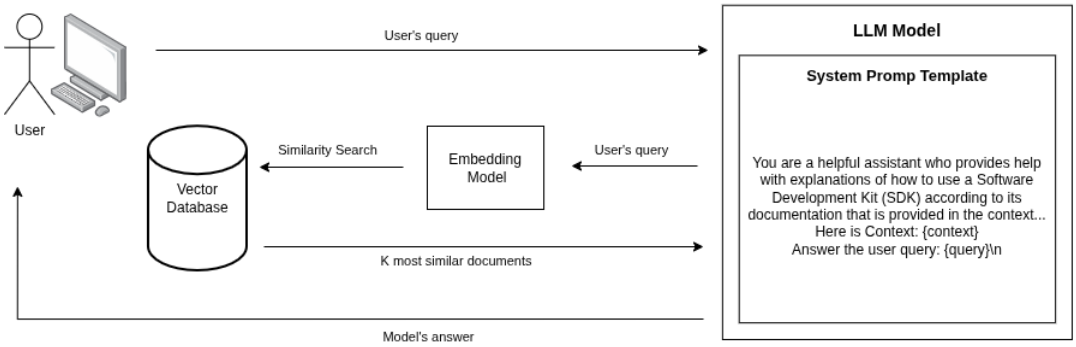

Fig. 4. System Template

To select an LLM, four generative models were compared: qwen2.5:7b [26], qwen3:8b [27], llama3.1:8b [28], and mistral:7b [29]. These models were chosen due to hardware restrictions and organizational policies for cloud applications. Open-source and smaller LLMs enabled on-premise deployment, ensuring private data security while balancing performance and computational efficiency.

3.3 Evaluation metrics

A combination of each chunk strategy with an embedding model and a generative LLM model has been implemented, resulting in 16 distinct experiments.

The Nomic embedding model was selected to generate vector embeddings for the LLM answer and ground truths for comparison within the RAGAS framework for semantic similarity. To compute BERTScore, the bertbase-uncased model was utilized.

Evaluation Tools For human evaluation, detailed in the following Section, we developed a front-end tool in React, as proposed in the work [34], to provide an easy interface for

volunteers to evaluate the generated answers. For each experiment, we stored the results including questions, retrieved chunks, and generated answers in a JSON file and uploaded it to our front-end tool.

The front-end tool, illustrated in Figure 5, allowed volunteers to upload a JSON file containing questions, generated answers, retrieved chunks and ground truth. They could also view the question, the generated answer besides the expected answer, and the chunks retrieved by the RAG. They used the five options explained in the previous Section to evaluate the Adequacy, Usefulness, and Relevance of the generated answer.

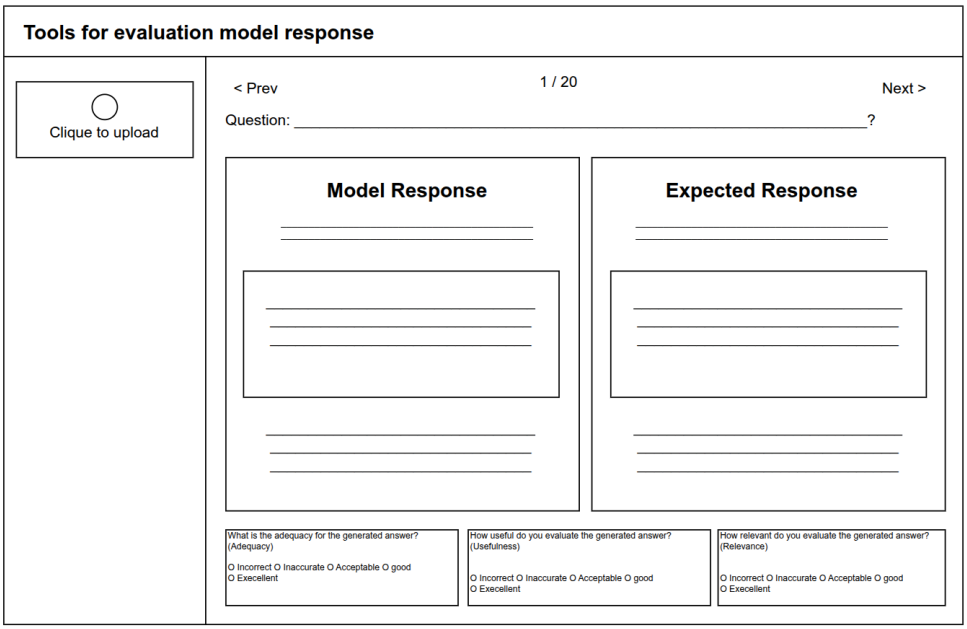

Fig. 5. Human Evaluation Tool

To evaluate the metrics BLEU, ROUGE, and semantic similarity, we selected the open-source RAGAS framework. It offered the necessary metrics and was straightforward to use, requiring minimal configuration. Integration with external systems or tools was unnecessary since we avoided metrics that required an LLM as a judge. Additionally, we used the BERTScore library in a Python script to calculate this metric, which was also simple to use.

3.4 Human Evaluation Methodology

Before conducting the human evaluation, we instructed users on how to evaluate responses and set thresholds for each feedback category. They were also briefed about the evaluation system.

The human evaluation consisted of 4 rounds, with only one round conducted per day for 12 evaluators, who were end users from the organization. The criteria for evaluating each generated answer were:

- Adequacy examines the quality of responses based on their completeness and richness;
- Usefulness judges the actual value of the answer helping to understand and perform the task effectively;
- Relevance measures the LLM response accuracy to the ground truth.

Each criterion was evaluated by developers using five-point Likert scale: Incorrect if the answer was totally incorrect; Inaccurate if the answer contained more wrong statements than correct ones; Acceptable if most statements were correct; Good if all statements were correct; Excellent if the answer was detailed and explained with examples beyond the ground truth.

4 Results and Discussion

This Section evaluates and analyzes results for metrics such as BLEU, Rouge, Semantic Similarity, and Bert Score. Scores range from 0 to 1, with higher values indicating superior performance and 1 representing the optimal result.

The subsequent tables break down these metrics for each LLM, embedding model, and chunk strategy. Tables 1, 2, and 3 present the scores for easy, medium, and hard question levels, respectively, allowing a comparison of their relative strengths and weaknesses. The top metric value per model appears in bold, and the highest values are bolded and grayed out for emphasis.

Table 1 shows that for easy-level questions, the split chunk method achieved higher or equal Semantic Similarity scores compared to the Section chunk method, regardless of the LLM and Embedding model used. No other patterns emerged between the chunk method and other metrics. In general, Llama3.1:8B achieved higher results with BGE-M3 and the Section chunk method; Mistral:7B also used BGE-M3 but performed better with the Split chunk method; Qwen2.5:7B using Nomic-V.15 and the Split chunk method yielded the highest values for most metrics; and Qwen3:8B did not show any pipeline with more than one highest-value metric. Qwen3:8B BERTScore and BLEU results were lower than those of Qwen2.5:7B, while Semantic Similarity remained similar. The Rouge score improved over the previous model.

Table 1. Model Scores for Easy Level Question

LLM	Embedding	Chunk	Sem. Sim.	BERTScore	BLEU	ROUGE
Llama3.1:8B	BGE-M3	Section	0.892	**0.785**	**0.437**	0.378
	BGE-M3	Split	0.895	0.780	0.422	0.373
	Nomic-V1.5	Section	**0.889**	0.776	0.412	0.369
	Nomic-V1.5	Split	0.892	0.784	0.399	**0.384**
Mistral:7B	BGE-M3	Section	0.890	0.765	0.456	0.351
	BGE-M3	Split	**0.897**	**0.771**	0.453	**0.367**
	Nomic-V1.5	Section	0.895	0.769	**0.470**	0.358
	Nomic-V1.5	Split	0.895	0.770	0.443	0.364
Qwen2.5:7B	BGE-M3	Section	0.905	0.790	0.473	0.408
	BGE-M3	Split	0.905	0.790	0.485	0.397
	Nomic-V1.5	Section	0.904	0.792	**0.497**	0.409
	Nomic-V1.5	Split	**0.911**	**0.794**	0.490	**0.410**
Qwen3:8B	BGE-M3	Section	0.905	**0.789**	0.425	0.411
	BGE-M3	Split	0.908	0.785	0.425	**0.423**
	Nomic-V1.5	Section	0.906	0.784	**0.431**	0.411
	Nomic-V1.5	Split	**0.910**	0.784	0.425	0.415

Table 2 shows that for medium-level questions. No pattern was found for any metric, the best scores varies for each metric. Qwen2.5:7B showed the greater score with BGE-M3 embedding and split chunks, reaching highest Semantic Similarity score of 0.893. For BLEU scores the highest score achieved by Qwen2.5:7B paired with section chunks and

Nomic-V1.5 embedding model. For BERTScore, the Split chunk strategy with Nomic-V1.5 embedding proved the best score with Qwen2.5:7B among the LLMs. Qwen2.5:7B, demonstrated consistent higher performance across multiple metrics, achieving the highest values in Semantic Similarity (0.893), BLEU (0.459), and BERTScore (0.783), while Qwen3:8B showed improved ROUGE scores compared to its predecessor but maintained similar performance in other metrics.

Table 2. Model Scores for Medium Level Question

LLM	Embedding	Chunk	Sem. Sim.	BERTScore	BLEU	ROUGE
Llama3.1:8B	BGE-M3	Section	0.880	0.770	0.393	0.342
	BGE-M3	Split	0.881	0.774	0.397	0.350
	Nomic-V1.5	Section	0.872	0.761	**0.418**	0.345
	Nomic-V1.5	Split	**0.886**	**0.779**	0.406	**0.361**
Mistral:7B	BGE-M3	Section	0.879	**0.770**	0.424	**0.351**
	BGE-M3	Split	**0.886**	0.764	0.418	0.340
	Nomic-V1.5	Section	0.884	0.767	**0.451**	0.342
	Nomic-V1.5	Split	**0.886**	0.766	0.427	0.340
Qwen2.5:7B	BGE-M3	Section	0.869	0.778	0.441	0.358
	BGE-M3	Split	**0.893**	0.781	0.454	**0.371**
	Nomic-V1.5	Section	0.884	0.779	**0.459**	0.363
	Nomic-V1.5	Split	0.891	**0.783**	0.445	0.368
Qwen3:8B	BGE-M3	Section	0.886	0.774	0.394	0.366
	BGE-M3	Split	0.890	0.772	**0.410**	0.368
	Nomic-V1.5	Section	0.885	0.773	0.401	0.369
	Nomic-V1.5	Split	**0.892**	**0.779**	0.403	**0.388**

Table 3 shows that for hard-level questions, the scores became more diverse. For Semantic Similarity, Mistral:7B and Qwen2.5:7B achieved highest score of 0.873 with different configurations, Qwen2.5 using Nomic-V1.5 with Split chunks, while Mistral paired with BGE-M3. Llama3.1:8B showed the same performance for Semantic Similarity when using Nomic-V1.5 embedding, for both chunk strategies achieving the score 0.861. For BERTScore, Qwen2.5:7B achieved the highest score of 0.759 with BGE-M3 and Split chunks. For BLEU metric Qwen2.5:7B achieved the best score of 0.484 using Nomic-V1.5 and Section chunks. For ROUGE scores, Qwen3:8B achieved the highest value of 0.335 with BGE-M3 and Split chunks. For hard level questions, BLEU and Rouge metrics achieved the the lowest scores in comparison to easy and medium question levels.

The overall results for evaluated metrics across all question levels is shown in Table 4. All four LLMs achieved higher BLEU scores with the section chunk method. The Nomic embedding model paired best with this strategy for Mistral, Qwen2.5, and Qwen3, while BGE-M3 performed optimally for Llama3.1.

For the ROUGE score, the best chunk strategy was the split method, which, when combined with the nomic-v1.5 embedding model, achieved better results for Mistral, Llama3.1, and Qwen2.5. For Qwen3, the best result was achieved by combining it with the bge-m3 embedding model.

Regarding semantic similarity, the split chunk strategy proved most effective for overall questions, achieving the highest scores for Llama3.1, Qwen2.5, and Qwen3 when paired with nomic, while Mistral performed better with bge-m3.

For BertScore, the best chunk method for Mistral resulted in a tie, as Mistral achieved the same score with both nomic combined with section and split chunk. For Llama3.1, Qwen2.5, and Qwen3, the nomic and split chunk method yielded better performance.

Table 3. Model Scores for Hard Level Question

LLM	Embedding	Chunk	Sem. Sim.	BERTScore	BLEU	ROUGE
Llama3.1:8B	BGE-M3	Section	0.832	0.727	0.331	0.258
	BGE-M3	Split	0.847	0.729	**0.357**	0.277
	Nomic-V1.5	Section	**0.861**	**0.755**	0.339	0.290
	Nomic-V1.5	Split	**0.861**	0.746	0.324	**0.298**
Mistral:7B	BGE-M3	Section	0.862	0.754	0.402	0.310
	BGE-M3	Split	**0.873**	0.744	0.372	0.294
	Nomic-V1.5	Section	0.852	**0.756**	**0.427**	0.306
	Nomic-V1.5	Split	0.850	0.751	0.410	**0.313**
Qwen2.5:7B	BGE-M3	Section	0.844	0.752	0.424	0.303
	BGE-M3	Split	0.856	**0.759**	0.422	0.322
	Nomic-V1.5	Section	0.864	0.754	**0.484**	**0.323**
	Nomic-V1.5	Split	**0.873**	0.756	0.446	0.318
Qwen3:8B	BGE-M3	Section	0.865	0.734	0.351	0.315
	BGE-M3	Split	0.866	**0.746**	0.317	**0.335**
	Nomic-V1.5	Section	**0.872**	0.737	**0.363**	0.323
	Nomic-V1.5	Split	0.857	0.735	0.357	0.312

To address RQ1, we tested various document chunking and embedding techniques. The combination of the split chunk method and the nomic-v1.5 embedding model slightly outperformed others, achieving the highest scores in 2 out of 4 LLMs for overall scores, the exception was the withdraw in Mistral:7B and Qwen3:8B where each metric had its best score with a different embedding model and chunk strategy.

This result indicates that split chunk strategy retrieves more relevant information than section chunk. Increasing the quantity of information per chunk does not always positively impact the LLM's response, as it may introduce irrelevant or redundant information, since section chunks are about 50% larger than split chunks. Conversely, providing smaller pieces of information adds variety and enriches the LLM's context for improved responses, this might be because some questions needed specific information from different parts of the documentation.

Table 4. Overall scores

LLM	Embedding	Chunk	Sem. Sim.	BERTScore	BLEU	ROUGE
Llama3.1:8B	BGE-M3	Section	0.883	0.775	**0.414**	0.357
	BGE-M3	Split	0.886	0.774	0.409	0.357
	Nomic-V1.5	Section	0.882	0.770	0.407	0.354
	Nomic-V1.5	Split	**0.888**	**0.779**	0.394	**0.369**
Mistral:7B	BGE-M3	Section	0.884	0.765	0.442	0.347
	BGE-M3	Split	**0.892**	**0.767**	0.435	0.352
	Nomic-V1.5	Section	0.888	**0.767**	**0.461**	0.349
	Nomic-V1.5	Split	0.888	**0.767**	0.435	**0.353**
Qwen2.5:7B	BGE-M3	Section	0.889	0.783	0.459	0.384
	BGE-M3	Split	0.897	0.784	0.470	0.383
	Nomic-V1.5	Section	0.895	0.785	**0.485**	0.388
	Nomic-V1.5	Split	**0.902**	**0.788**	0.473	**0.389**
Qwen3:8B	BGE-M3	Section	0.896	**0.779**	0.410	0.390
	BGE-M3	Split	0.899	0.778	0.411	**0.399**
	Nomic-V1.5	Section	0.897	0.776	**0.416**	0.391
	Nomic-V1.5	Split	**0.900**	0.778	0.413	0.398

For RQ2, we evaluated four different LLM models. Referring to the overall scores in Table 4, the Qwen2.5:7B model achieved the highest scores in most metrics such Qwen2.5:7B

excelled in BLEU (0.485), semantic similarity (0.902), and BERTScore (0.788), the exception was for the ROUGE score, where Qwen3:8B outperformed it with 0.39.

Qwen2.5 achieved the highest scores in most metrics. The easy level obtained better scores than the medium level, and the model performed better for medium level questions than for hard level ones. The other three models exhibited similar behavior, with scores decreasing as questions became more challenging. This outcome was expected, as answering harder level questions required more refined and combined chunks, necessitating greater refinement to generate better answers.

The human evaluation average scores range from 1 to 5 and are presented in Table 5. This evaluation aimed to assess the answers generated by the optimal combination of chunk strategy and embedding model for each LLM model.

For human evaluation, we selected the combination that achieved the highest scores across most metrics for each LLM. Therefore, the selected models were: Llama3.1, Mistral, Qwen2.5 paired with split chunks and nomic-v1.5, and Qwen3 paired with section chunks and BGE-M3. The Qwen2.5:7B model achieved the highest average score across all three evaluated aspects, with the following average scores and their respective standard deviations (std.): Adequacy 3.27 (1.24), Relevance 3.36 (1.24), and Usefulness 3.36 (1.23).

Table 6 shows Qwen2.5:7B human evaluation scores by question level. It is possible to notice that as the difficulty level increases, the scores decrease. This behavior is observed across all four models used in this experiment. This clarifies the tendency to reduce the score as questions become more difficult, a trend already mentioned in the metrics results.

Table 5. Human Evaluation - Overall Scores

Model	Adequacy	Relevance	Usefulness
Qwen2.5:7B	**3.27**	**3.36**	**3.36**
Llama3.1:8B	3.20	3.22	3.22
Mistral:7B	3.02	3.05	3.04
Qwen3:8B	2.97	2.99	2.99

Table 6. Human Evaluation - Model Scores by Question Level

LLM	Level	Adequacy	Relevance	Usefulness
Mistral:7B	Easy	3.17	3.19	3.18
	Medium	2.81	2.83	2.81
	Hard	2.71	2.72	2.74
Qwen2.5:7B	Easy	3.56	3.67	3.66
	Medium	2.82	2.85	2.90
	Hard	2.70	2.79	2.78
Qwen3:8B	Easy	3.24	3.26	3.26
	Medium	2.42	2.44	2.45
	Hard	2.81	2.85	2.86
Llama3.1:8B	Easy	3.37	3.37	3.37
	Medium	3.04	3.05	3.06
	Hard	2.64	2.68	2.68

For RQ3, Qwen2.5:7B also was the best model evaluated during the human evaluation phase, with a moderate standard deviation value, indicating that evaluators shared different opinions about generated answers. For overall scores, an adequacy score of 3.27 suggests that the model's answer was sufficient to address the query. A relevance score of

3.36 indicates that the answer aligns with the defined ground truth. A usefulness score of 3.36 demonstrates that the model's answer helps to solve the problem and its solutions are reasonable.

5 Lessons Learned and Threats to Validity

During the human evaluation we did not measure the LLMs' hallucinations in generated answers due to the lack of volunteers to conduct this type of inspection manually. We were limited by computational resources to use an LLM as a judge.

To overcome computational limitations in evaluation, we adopted the BLEU and ROUGE metrics, which are widely used for NLP evaluations. However, they lack semantic understanding, as evidenced by their scores all below 0.5. In contrast, semantic similarity and BERTScore scores were all above 0.7. Therefore, adopting semantic similarity and BERTScore metrics was a reasonable choice to address the limitations of BLEU and ROUGE metrics.

Among the embedding models, nomic-v1.5 achieved higher scores than BGE-M3. This indicates that, in the tested domain, the nomic-v1.5 vector of dimension 768 outperformed BGE-M3, which generated vectors of dimension 1024.

The metrics values and human evaluation scores were very close to each other, indicating that the chunk strategies had no impact on embedding generation and, consequently, on the generated answers in the proposed scenarios.

6 Conclusion and Future Work

This paper investigated the application of the RAG technique in an organizational environment to assist software engineers to use a private SDK documentation for developing and maintaining automation scripts. Two distinct chunk methods were used to extract the documentation, which was then indexed in a vector database with two different embedding models, paired with four LLMs to generate answers to documentation-related questions. For each combination of chunk method, embedding model, and LLM, metrics were calculated, and human evaluation was conducted to assess aspects of the generated answers.

Given the limitations related to data privacy, organizational documentation, and computational resources, we chose to use on-premise small open-source LLMs to implement the RAG system. As shown in the results, a good perception of the generated answers was achieved, which is considered suitable for practical use and its integration with a larger system to provide this documentation assistance is also in course.

The split chunk method, combined with the nomic-v1.5 embedding model, performed better for three out of four LLMs. The only exception was Qwen3:8B, which performed better using BGE-M3 for ROUGE score and Qwen2.5:7B paired with section chunks achieved the highest score for BLEU metric. Qwen2.5:7B, when paired with split chunks and nomic-v1.5, achieved the highest scores for semantic similarity and BERTScore metrics. It also achieved the best scores in human evaluations for the overall score across question levels, with scores tending to decrease as questions became more difficult. This necessitated improved information refinement in the indexing, retrieval, and generation phases, which will be addressed in future work.

For future work we intend to explore advanced RAG and modular RAG [7] techniques to improve results, the intention is that the system correctly answers most of hard level question. Additionally, tests with additional datasets to assess generalization for other documentations is also being considered.

Acknowledgment

This work is the result of the R&D project *Projeto de Engenharia de Software e Ciência de Dados aplicados ao Desenvolvimento de Sistemas*, performed by Sidia Instituto de Ciência e Tecnologia in partnership with Samsung Eletrônica da Amazônia Ltda., using resources from Federal Law No. 8.387/1991, and its disclosure and publicity are under the provisions of Article 39 of Decree No. 10.521/2020.

References

1. Diego Costa, Gabriel Matos, Gilson Russo, Leon Barroso, and Erick Bezerra. Mycroft - retrieval augmented generation for SDK documentation. In *Natural Language Computing*, pages 141–152. Academy and Industry Research Collaboration Center (AIRCC), November 2025.
2. Kailash A Hambarde and Hugo Proenca. Information retrieval: recent advances and beyond. *IEEE Access*, 11:76581–76604, 2023.
3. Yutao Zhu, Huaying Yuan, Shuting Wang, Jiongnan Liu, Wenhan Liu, Chenlong Deng, Haonan Chen, Zheng Liu, Zhicheng Dou, and Ji-Rong Wen. Large language models for information retrieval: A survey. *arXiv preprint arXiv:2308.07107*, 2023.
4. Masoomali Fatehkia, Ji Kim Lucas, and Sanjay Chawla. T-rag: lessons from the llm trenches. *arXiv preprint arXiv:2402.07483*, 2024.
5. Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. Large language models struggle to learn long-tail knowledge. In *International Conference on Machine Learning*, pages 15696–15707. PMLR, 2023.
6. Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):1–55, 2025.
7. Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2(1), 2023.
8. Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.
9. Mahimai Raja, E Yuvaraajan, et al. A rag-based medical assistant especially for infectious diseases. In *2024 International Conference on Inventive Computation Technologies (ICICT)*, pages 1128–1133. IEEE, 2024.
10. Muhammad Rafsan Kabir, Rafeed Mohammad Sultan, Fuad Rahman, Mohammad Ruhul Amin, Sifat Momen, Nabeel Mohammed, and Shafin Rahman. Legalrag: A hybrid rag system for multilingual legal information retrieval. *arXiv preprint arXiv:2504.16121*, 2025.
11. Md Saleh Ibtasham, Sarmad Bashir, Muhammad Abbas, Zulqarnain Haider, Mehrdad Saadatmand, and Antonio Cicchetti. Reqrag: Enhancing software release management through retrieval-augmented llms: An industrial study. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*, pages 277–292. Springer, 2025.
12. Chengrui Wang, Qingqing Long, Meng Xiao, Xunxin Cai, Chengjun Wu, Zhen Meng, Xuezhi Wang, and Yuanchun Zhou. Biorag: A rag-llm framework for biological question reasoning. *arXiv preprint arXiv:2408.01107*, 2024.
13. Antonio Jimeno Yepes, Yao You, Jan Milczek, Sebastian Laverde, and Renyu Li. Financial report chunking for effective retrieval augmented generation. *arXiv preprint arXiv:2402.05131*, 2024.
14. Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2), 2023.
15. Penghao Zhao, Hailin Zhang, Qinhan Yu, Zhengren Wang, Yunteng Geng, Fangcheng Fu, Ling Yang, Wentao Zhang, Jie Jiang, and Bin Cui. Retrieval-augmented generation for ai-generated content: A survey. *arXiv preprint arXiv:2402.19473*, 2024.
16. Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. In-context retrieval-augmented language models. *Transactions of the Association for Computational Linguistics*, 11:1316–1331, 2023.

17. Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.
18. Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.
19. Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert, 2020.
20. Tolga Şakar and Hakan Emekci. Maximizing rag efficiency: A comparative analysis of rag methods. *Natural Language Processing*, 31(1):1–25, 2025.
21. Irina Radeva, Ivan Popchev, Lyubka Doukovska, and Miroslava Dimitrova. Web application for retrieval-augmented generation: Implementation and testing. *Electronics*, 13(7):1361, 2024.
22. Yifan Yao, Jinhao Duan, Kaidi Xu, Yuanfang Cai, Zhibo Sun, and Yue Zhang. A survey on large language model (llm) security and privacy: The good, the bad, and the ugly. *High-Confidence Computing*, 4(2):100211, 2024.
23. Nir Kshetri. Cybercrime and privacy threats of large language models. *IT Professional*, 25(03):9–13, 2023.
24. Simone Balloccu, Patrícia Schmidtová, Mateusz Lango, and Ondřej Dušek. Leak, cheat, repeat: Data contamination and evaluation malpractices in closed-source llms. *arXiv preprint arXiv:2402.03927*, 2024.
25. Hanbo Huang, Yihan Li, Bowen Jiang, Lin Liu, Bo Jiang, Ruoyu Sun, Zhuotao Liu, and Shiyu Liang. On-premises llm deployment demands a middle path: Preserving privacy without sacrificing model confidentiality. In *ICLR 2025 Workshop on Building Trust in Language Models and Applications*, 2025.
26. An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. Qwen2 technical report, 2024.
27. An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025.
28. Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407, 2024.
29. Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023.
30. Sriram Veturi, Saurabh Vaichal, Reshma Lal Jagadheesh, Nafis Irtiza Tripto, and Nian Yan. Rag based question-answering for contextual response prediction system. *arXiv preprint arXiv:2409.03708*, 2024.
31. Jianguo Wang, Xiaomeng Yi, Rentong Guo, Hai Jin, Peng Xu, Shengjun Li, Xiangyu Wang, Xiangzhou Guo, Chengming Li, Xiaohai Xu, et al. Milvus: A purpose-built vector data management system. In *Proceedings of the 2021 International Conference on Management of Data*, pages 2614–2627, 2021.
32. Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. Dense passage retrieval for open-domain question answering, 2020.
33. Zach Nussbaum, John X. Morris, Brandon Duderstadt, and Andriy Mulyar. Nomic embed: Training a reproducible long context text embedder, 2025.
34. Nelson F Liu, Tianyi Zhang, and Percy Liang. Evaluating verifiability in generative search engines. *arXiv preprint arXiv:2304.09848*, 2023.