Evaluating the Efficacy of Open-Source LLMs in Enterprise-Specific RAG Systems: A Comparative Study of Performance and Scalability

1st Gautam Balakrishnan
Indian Institute of Technology Madras
Chennai, India
gautamb1a2b@gmail.com

2nd Anupam Purwar
Independent
Delhi, India
anupam.aiml@gmail.com

2024 IEEE 21st India Council International Conference (INDICON) | 979-8-3503-9128-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/INDICON63790.2024.10958508

***Abstract*—This paper presents an analysis of open-source large language models (LLMs) and their application in Retrieval-Augmented Generation (RAG) tasks, specific for enterprise-specific data sets scraped from their websites. With the increasing reliance on LLMs in natural language processing, it is crucial to evaluate their performance, accessibility, and integration within specific organizational contexts. This study examines various open-source LLMs, explores their integration into RAG frameworks using enterprise-specific data, and assesses the performance of different open-source embeddings in enhancing the retrieval and generation process. Our findings indicate that open-source LLMs, combined with effective embedding techniques, can significantly improve the accuracy and efficiency of RAG systems, offering a viable alternative to proprietary solutions for enterprises.**

***Index Terms*—Large language models(LLMs), Retrieval Augmented Generation (RAG), Natural language processing(NLP),information retriever, ROUGE score, cosine similarity, top-k, Llama3, Mistral,Generative Pre-Trained Transformers(GPT),Generative AI(Gen AI),Cosine Similarity with Groundtruth Answer(CSGA)**

I. Introduction

The rapid advancements in natural language processing (NLP) have led to the development of sophisticated large language models (LLMs) that excel in tasks such as text generation, summarization, and question answering. Among these advancements, Retrieval-Augmented Generation (RAG) has emerged as a promising approach for the retrieval-based systems with generative models to produce highly accurate and contextually relevant outputs.The concept of Retrieval-Augmented Generation (RAG) was introduced by Lewis et al. In their seminar 2020 paper titled "Retrieval-Augmented Generation for Knowledge- Intensive NLP Tasks." [20].In their research, Lewis et al. present a method that combines retrieval-based and generative models to enhance the performance of knowledge-intensive tasks. By integrating non-parametric memory (retrieved documents) with parametric memory (the generative model's internal parameters), RAG models achieve superior accuracy and flexibility in tasks such as open-domain question answering and abstract question answering.

Karpukhin et al. (2020) developed dense passage retrieval for open-domain question answering, which significantly boosts retrieval accuracy by using dense vector representations and a neural retriever [18].More recent work further advances the field by introducing novel methodologies for fine-tuning LLMs specifically for RAG tasks in knowledge-intensive environments [24]. There has been efficient ways to improve the retrieval process such as the Keyword Augmented Retrieval (KAR), which integrates keyword generation using transformer models with document metadata to identify the right context quickly and cost-effectively [23]. Also, approach to handle sparse information where classical RAG using hybrid retriever fails to generate correct answers have been reported [17]. More recent work by Tay et al. (2023) on the UL2 model and studies on ColBERT by Khattab and Zaharia (2020) have further pushed the boundaries of retrieval and generation synergies in RAG frameworks [19] [25].

Despite the potential of RAG systems, their application within enterprise environments remains under explored, particularly concerning the use of open-source solutions.Enterprises possess vast and diverse repositories of data, typically scattered across various internal systems and public websites.Proprietary LLMs, while powerful, come with high licensing costs and restrictive usage policies, limiting their accessibility and adaptability for many organizations [22]. This creates a substantial barrier for enterprises, especially small to medium-sized ones, to leverage advanced Generative AI capabilities.

Evaluating the RAG framework involves several key metrics and methodologies to ensure its effectiveness and efficiency in real-world applications [27].There have been several metrics such as retrieval accuracy, response relevance, latency, and other metrics by others frameworks are crucial for assessing the performance of RAG systems [16] [2]. Studies have shown that the integration of advanced retrieval mechanisms with generative models can significantly enhance the quality of responses in various Generative AI tasks. Additionally, the adaptability of RAG systems to different data sets and contexts plays a vital role in their evaluation, highlighting the importance of domain-specific fine-tuning and optimization. This deepeval based evaluation framework helps in identifying the

strengths and limitations of different RAG implementations, guiding the development of more robust and scalable solutions [2].

Furthermore, our analysis of how TopK affects answer quality for different LLMs reveals significant variations in performance. By adjusting the TopK parameter, which determines the number of top retrievals considered, we can observe changes in the accuracy and relevance of the generated answers. This analysis is crucial for understanding how to optimize retrieval settings for different models and use cases, providing insights into the fine-tuning required for optimal performance in specific enterprise contexts.

This research seeks to address the following things:

1) How do open-source LLMs and embeddings compare with each other and with proprietary alternatives in terms of accuracy and efficiency in RAG tasks using enterprise-specific data?
2) What are the most important metrics to evaluate quality of RAG answers?
3) What is the best combination of hyper parameters for RAG ?

Addressing these questions is critical for providing enterprises with viable Generative AI solutions that are both effective and economically sustainable. This study will contribute to the broader understanding of the potential and limitations of open-source Generative AI tools in real-world business settings, offering a pathway for organizations to enhance their information retrieval and content generation capabilities.

II. Methodology

This section outlines the methodology employed to evaluate the effectiveness of open-source large language models (LLMs) and embedding techniques in enhancing Retrieval-Augmented Generation (RAG) systems for enterprise-specific data. The methodology is structured into several key stages: data collection, model selection, system architecture, evaluation metrics, and experimental procedure.

A. Data Collection

Data collection is a crucial step in developing and evaluating Retrieval-Augmented Generation (RAG) systems, particularly when dealing with enterprise-specific datasets. In this study, data was scraped from the website https://i-venture.org/, utilizing a structured approach as mentioned below to ensure the accuracy and comprehensiveness of the collected data.

1) Sitemap Extraction: The initial step in the data collection process is extracting Uniform Resource Locator(URL) from the website's sitemap. The process included:

1) **URL Retrieval**: Accessing the sitemap located at https://i-venture.org/sitemap.xml to retrieve the Extensible Markup Language (XML) content.
2) **Parsing the Sitemap**: Using an XML parser to read and interpret the sitemap.
3) **URL Extraction**: Extracting all `<loc>` tags, which contain the URLs of the web pages, and compiling these into a list. This list served as the foundation for the subsequent crawling process.

2) Sitemap Extraction: The initial step in the data collection process is extracting Uniform Resource Locator(URL) from the website's sitemap. The process included:

1) **URL Retrieval**: Accessing the sitemap located at https://i-venture.org/sitemap.xml to retrieve the Extensible Markup Language (XML) content.
2) **Parsing the Sitemap**: Using an XML parser to read and interpret the sitemap.
3) **URL Extraction**: Extracting all `<loc>` tags, which contain the URLs of the web pages, and compiling these into a list. This list served as the foundation for the subsequent crawling process.

3) Web Crawling: With the URLs extracted from the sitemap, each URL has to be crawled to extract textual content. This was performed using a breadth-first search approach with the following steps:

1) **Queue Initialization**: A queue was initialized with the URLs obtained from the sitemap.
2) **Directory Setup**: Directories were created to store raw text files and processed files, ensuring a well-organized data repository.
3) **Content Extraction**: For each URL in the queue, the web page content was fetched using HTTP requests. The HTML content was parsed using an HTML parser to extract the main textual content while ignoring the HTML tags.
4) **File Storage**: The extracted text was cleaned (e.g., by removing extra white spaces ,new line characters and special characters) and saved into text files named based on the URL structure. This involved replacing slashes and other non-filename characters with underscores to ensure valid file names.

This ensures that the dataset from https://i-venture.org was comprehensive, clean, and ready to be used for RAG task.

B. Text Splitting

The next step is splitting the data into chunks. This is essential for ensuring that the text is appropriately segmented so that only the most relevant chunks gets passed to the LLM for RAG. The process was carried out using the `langchain` library, specifically leveraging the `DirectoryLoader` and `NLTKTextSplitter` tools [3].

1) **DirectoryLoader**: This tool was directed to the directory containing all the text files to load all the text.
2) **NLTKTextSplitter**: This tool was used to divide the text into smaller chunks based on NLTK tokens without trying to break paragraphs sentences and words. Smaller text chunks improve the accuracy of the retrieval component by allowing it to match queries more precisely with relevant sections of text [4]. For NLTKSplitter chunk size need not be specified. However TopK value of 5 has been used for this evaluation while using NLTKSplitter.

3) **RecursiveCharacterTextSplitter**: This can also be used which splits the text into chunks as small as possible while preserving the structure of all paragraphs (and then sentences, and then words) together , as those would generically seem to be the strongest semantically related pieces of text. It also uses chunk overlap ensuring information is available between neighbouring chunks [5]. In this study for the evaluation the RecursiveCharacterTextSplitter with token limit of 1024 and chunk overlap of 102 tokens has been used.

C. Embedding Generation

The next step involves generating embeddings for the text chunks created by text splitting. Embeddings are crucial for converting text into numerical representations that can be efficiently processed by llms [21]. This study utilized embeddings from Hugging Face, a popular platform providing pre-trained models for various Generative AI tasks [6].

1) **Embedding Storage**: A local file store was set up to cache the embeddings, ensuring efficient access and retrieval during RAG.
2) **Embedding Model**: In this work the required embedding model has been loaded from the Hugging Face [6]. **BAAI/bge-large-en-v1.5** has been selected as the embedding model owing to it's good performance in semantic search [26] . Besides it also supports ReRanking of retrieved texts [1].

D. Vector Database Creation

FAISS (Facebook AI Similarity Search) has been used to create a vector database to store embeddings [7]. The text chunks created before were converted into embeddings and stored in this FAISS vector database. This creates a structured repository that supports fast retrieval based on semantic similarity.

E. LLM Integration

This step involves incorporating LLMs to enhance the generative component of the Retrieval-Augmented Generation (RAG) system. For this study, we utilized open-source LLMs provided by Perplexity, integrating them into the `langchain` framework through a custom wrapper function (git link to wrapper code) . This integration was done by adapting the approach detailed in this github repository [8] [9].

1) Perplexity API: Perplexity offers API to a wide range of powerful open-source LLMs that can generate human-like text, making them ideal for tasks such as text generation, summarization, and question answering [10]. These models were chosen for their accessibility and flexibility, which are essential for enterprise applications where commercial/proprietary models might be prohibitively expensive [11].

2) Benefits of Using Perplexity API:

- **Cost-Effectiveness**: As open-source models, Perplexity provide a cost-effective alternative to proprietary solutions. For example, GPT-3.5 costs around 2 USD per million tokens on average, Perplexity LLMs takes only 0.6 USD per million tokens which is less than one-third of the price [11] [12].
- **Accessibility**: Perplexity provides API access to these open source LLMs without having to invest in local GPU capacity [13] [14].

F. Retrieval-Augmented Generation (RAG)

The next stage involves implementing the Retrieval-Augmented Generation (RAG) framework. This framework combines powerful retrieval techniques with generative models to produce accurate and contextually relevant outputs. The RAG system implemented in this work utilizes a hybrid retriever approach, combining a custom Best Match 25(BM25) retriever with a FAISS-based vector retriever, followed by a question-answering (QA) module that generates responses based on the retrieved information [15].

1) Hybrid Retriever Setup: To enhance retrieval accuracy, a hybrid retriever approach was adopted. This involved using both BM25 and FAISS retrievers with same weightage to both retrievers.

- **BM25 Retriever**: The BM25 algorithm is a well-known probabilistic information retrieval model that ranks documents based on their relevance to a query. It was configured to retrieve the top 5 documents most relevant to each query.
- **FAISS Retriever**: The FAISS retriever, leveraging the vector embeddings generated in the previous steps, was also set to retrieve the top k documents. FAISS excels in handling large-scale similarity searches efficiently.
- **Ensemble Retriever**: An ensemble retriever was created to combine the results from both BM25 and FAISS retrievers. Each retriever was assigned equal weight, ensuring a balanced contribution from both methods. This type of hybrid retriever has demonstrated better performance compared to a vector retriever alone [17].

2) RetrievalQA: After setting up the hybrid retriever, the next step was to implement the QA module using the `RetrievalQA` chain from the `langchain` library. This module is designed to generate answers to queries by leveraging the retrieved documents and providing source references for the generated answers.

- **Retriever Integration**: The hybrid retriever was used in conjunction with the QA module to ensure that the generated answers were based on the most relevant and contextually appropriate documents.
- **Callback Handling**: A callback handler was integrated to facilitate real-time monitoring and debugging of the QA process.

G. Evaluation

The evaluation of the Retrieval-Augmented Generation (RAG) system involves multiple metrics to assess its performance comprehensively. This includes measuring the quality of the generated responses, the efficiency of the retrieval process, and the contextual relevance of the answers. The following methods were employed:

1) ROUGE Scores: ROUGE (Recall-Orieated Understudy for Gisting Evaluation) is a set of metrics commonly used for evaluating the quality of text in tasks such as summarization and machine translation. It compares the overlap of n-grams between the generated text and the reference text.

2) DeepEval Metrics: To further assess the contextual quality of the responses, the DeepEval framework was used. This framework provides metrics for evaluating the precision, recall, and relevancy of the generated text in relation to the expected outputs and the context provided by the retrieved documents [2].

- **Contextual Precision**: Measures how many of the retrieved documents contain information that is relevant to the generated response.
- **Contextual Recall**: Measures the proportion of relevant information in the retrieved documents that is used in the generated response.
- **Contextual Relevancy**: Assesses how relevant the generated response is to the query and the context provided by the retrieved documents.

The evaluation process involved creating test cases where the input query, the actual output from the system, the expected output (generated using GPT-4), and the retrieval context were used to measure these metrics. Research has shown that GPT-4 agrees with human labelers around 80% of the time, making it a reliable and scalable option for evaluating natural language output [28].

The evaluation combines both qualitative and quantitative measures to provide a comprehensive assessment of the RAG system's performance. By using ROUGE scores, inference time, and DeepEval metrics, this study ensures that the system is evaluated for its accuracy, efficiency, and contextual relevance.

III. Results

When evaluating a dataset, categorizing the evaluation set into different segments helps in understanding the performance of models under various conditions. Here, the categories are based on the density of reasoning and factual information. Each category is defined as follows:

- **Reason Dense:** Reason dense segments consist of reasoning-based questions where the reasoning-related information appears repeatedly throughout the dataset. This involves complex reasoning tasks with repeated reasoning patterns or information.
- **Reason Sparse:** Reason sparse segments include reasoning-based questions where the reasoning-related information appears infrequently. This involves simpler reasoning tasks with limited instances of reasoning information.
- **Factual Dense:** Factual dense segments consist of factual questions where detailed factual information is repeated many times in the dataset. This involves numerous repeated facts requiring detailed knowledge.
- **Factual Sparse:** Factual sparse segments include factual questions where the factual information appears infrequently. This involves general knowledge with minimal repeated factual details.

A. Evaluating Answer Quality

For this evaluation, we selected three questions from each category of the above mentioned 4 categories and RecursiveCharacterTextSplitter has been used. The performance of the open sources LLMs used in RAG framework has been evaluated by calculating various metrics viz:

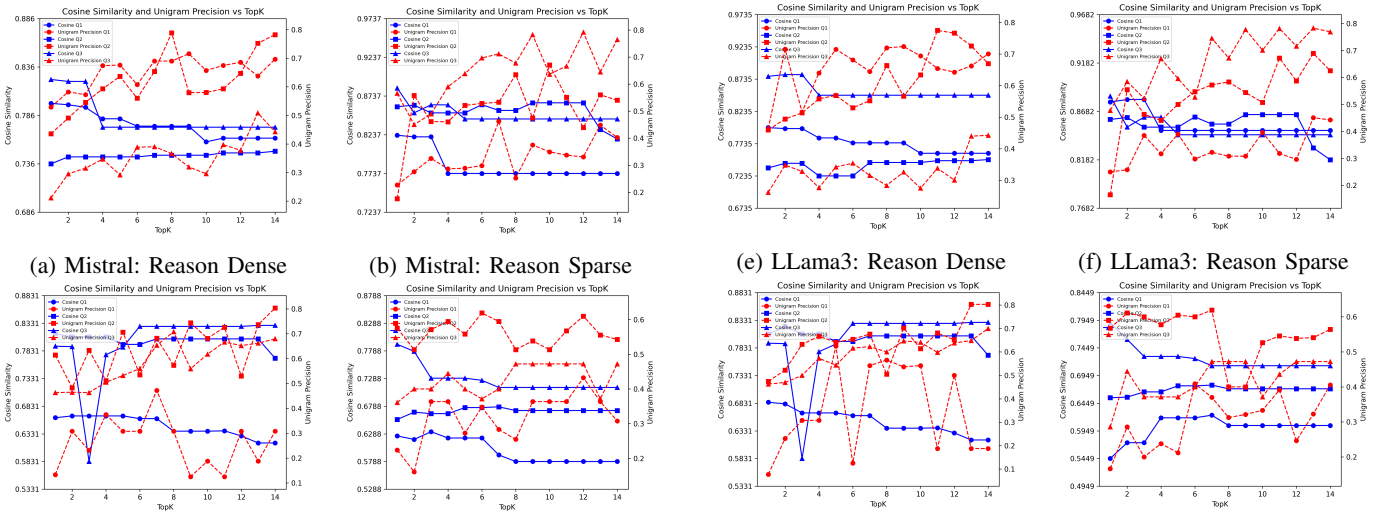
1) Cosine similarity: The cosine similarity metric measures the cosine of the angle between two non-zero vectors, providing a similarity score that ranges from -1 to 1. A higher score indicates greater similarity between the query and the content. In this work the cosine similarity between score between the query and the retrieved context for various top-k values has been calculated,where top-k denote the number of top ranking documents extracted during retrieval.

Based on the analysis presented in Figure 1, it is observed that as the value of top-k increases, the cosine similarity scores exhibit a trend of initial increase, followed by a plateau as top-k continues to rise. This suggests a diminishing return on similarity with higher top-k values.

2) Unigram Precision: : Unigram precision measures the fraction of the words in the LLM generated answer that are also present in the reference/context. Conversely, unigram precision shows a more erratic pattern but generally demonstrates an upward trend initially, reaching a peak before declining. This indicates that while increasing top-k can enhance unigram precision up to a certain point, further increments may lead to reduced precision. Additionally, it is noted that as cosine similarity starts to decrease, there are instances where unigram precision also dips across all four types of questions, further highlighting the interplay between similarity and precision metrics.

The Precision vs top-k graphs did not exhibit consistent variations across the Llama3 and Mistral models, as shown in Figure 1 . For Reason Dense data, the Llama3 model displayed an initial peak followed by a decrease for one question, whereas Mistral demonstrated a steady increase. In the Reason Sparse category, Llama3 showed a gradual increase, while Mistral peaked and then declined for two questions. For Factual Dense data, both models generally showed a steady increase, except for one question where Mistral initially decreased sharply before rising again. In the Factual Sparse category, the initial peak for Llama3 was not achieved again, whereas Mistral remained relatively constant. These observations highlight the different behaviors of the two models across various data types and question contexts.

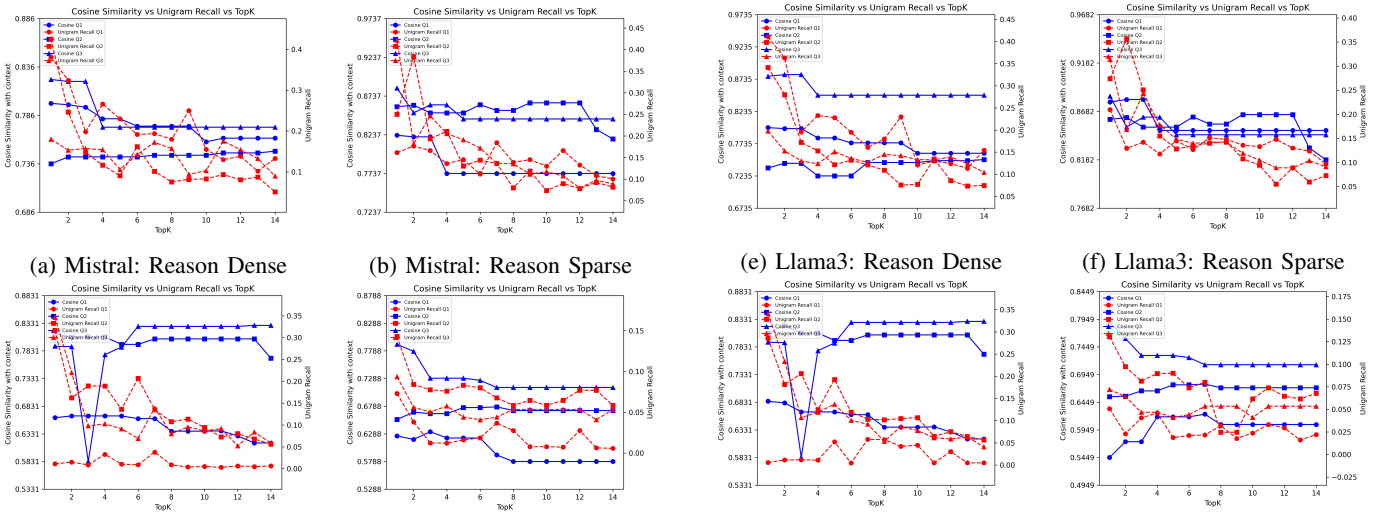
3) Unigram Recall: Unigram recall is the fraction of words in the reference/context that also appear in the LLM generated answer. We can observe from Figure1 that in general recall does not increase even when we increase top-k value. This can be explained by the fact that for this enterprise specific data, adding more chunks as context to LLM in the prompt



(a) Mistral: Reason Dense (b) Mistral: Reason Sparse (e) LLama3: Reason Dense (f) LLama3: Reason Sparse

(c) Mistral: Factual Dense (d) Mistral: Factual Sparse (g) LLama3: Factual Dense (h) LLama3: Factual Sparse

Fig. 1: Analysis of Cosine Similarity and Unigram Precision vs TopK for Mistral8x7B LLM model (left) and LLama3-8B LLM model (right)



(a) Mistral: Reason Dense (b) Mistral: Reason Sparse (e) Llama3: Reason Dense (f) Llama3: Reason Sparse

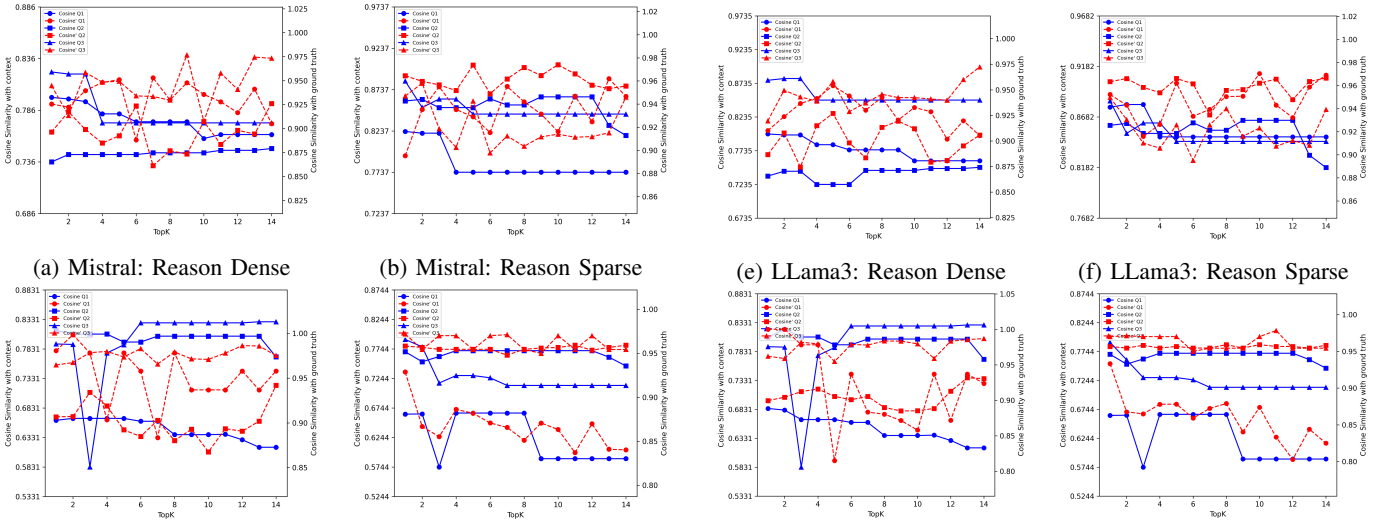
(c) Mistral: Factual Dense (d) Mistral: Factual Sparse (g) Llama3: Factual Dense (h) Llama3: Factual Sparse

Fig. 2: Analysis of Cosine Similarity and Unigram Recall vs TopK for Mistral8x7B LLM model (left) and Llama3-8B LLM model (right)

do not necessarily help improve response's recall because the text in additional chunks is not relevant to the query. This also correlates with the observation of no improvement in cosine similarity of retrieved context with the query as top-k value is increased. Based on this one can infer that there is no defined correlation between cosine similarity and Unigram recall so this metric is not useful in generating a good measure of answer quality.

4) Cosine similarity with ground truth answer(CSGA): This metric measures the cosine similarity between the LLM generated answer and the ground truth answer generated using GPT-4. From Figure3 we can infer that the values more or less remains constant as top-k increases signifying that the retrieved answer does not improve a lot even with increased top-k as the extra chunks will become irrelevant to the query.

In our analysis of Mistral and Llama3, we observe the distinct patterns in performance metrics across various question types. For reason dense questions, both models exhibited consistent performance values, with the exception of a single question where a increase was noted. This suggests that both models generally handle reason dense questions with stable accuracy, barring occasional outliers.In contrast, for reason sparse and factual dense questions, the performance values remained uniformly constant across all six questions for both



(a) Mistral: Reason Dense (b) Mistral: Reason Sparse (e) LLama3: Reason Dense (f) LLama3: Reason Sparse

(c) Mistral: Factual Dense (d) Mistral: Factual Sparse (g) LLama3: Factual Dense (h) LLama3: Factual Sparse

Fig. 3: Analysis of Cosine Similarity with context vs Cosine similarity with ground truth vs TopK for Mistral8x7B LLM model (left) and LLama3-8B LLM model (right)

Mistral and Llama3. This indicates a robust and consistent handling of reason sparse and a general reliability in processing factual dense questions with the exception of few anomalies. Lastly, for factual sparse questions, there is a noticeable decrease in performance values for one question, while the other two questions showed consistent values. This suggests that factual sparse questions may occasionally challenge the models, resulting in variable performance on certain questions. We further analysed the performance of Mistral and Llama3 using Deepeval framework.

B. Evaluation using deepeval scores

For this evaluation NLTKTextSplitter has been used. TableI provides the evaluation done for Llama3-8B and based on BAAI/bge-large-en-v1.5 embedding.

The same has been done for mistral 8x7B LLM based on BAAI/bge-large-en-v1.5 embedding model, refer TableII.

We observe that Llama3-8B significantly outperformed Mistral8x7B in all tasks in terms of unigram precision, indicating that the Llama3-8B model is highly effective at capturing relevant information at a granular level. Additionally, the inference time for Llama3-8B is on par with or better than Mistral8x7B, as shown in Table I and Table II. The variations in contextual metrics suggest that the performance of both models can be task-dependent. Mistral performs significantly worse on sparse reasoning and dense factual information retrieval, while it is comparable on dense reasoning and sparse factual information tasks. Despite having fewer parameters (8 billion for Llama3-8B) compared to Mistral8x7B(56 Billion), Llama3-8B has a slight edge over its competitor. Its ability to maintain high precision, coupled with lower inference times, makes it a robust option for text generation using enterprise specific data. These findings suggest that the Llama3-8B model can serve as a more efficient and effective alternative across all 4 question categories.

Moreover, comparing these open-source models such as Llama3-8B and Mistral8x7B with GPT-3.5 indicate that they outperform GPT-3.5 in several key areas:

- **Unigram Precision**: While GPT-3.5 achieves a unigram precision of 0.77, Llama3-8B slightly outperforms it ranging as values from 0.737 to 0.82 , while mistral 8x7B under performs with values ranging from 0.67 to 0.73 respectively.
- **Contextual Recall**: Both Llama3-8B (0.916-0.98) and Mistral (0.91-0.98) surpass GPT-3.5 (0.86) in contextual recall, demonstrating their superior ability to retrieve relevant context.
- **Contextual Relevancy**: Llama3-8B (0.636-0.947) and Mistral (0.56-0.92) significantly outperform GPT-3.5 (0.60) in contextual relevancy, indicating better alignment with the intended context.
- **Answer Relevancy**: Llama3-8B (0.93-1.00) performs on par with GPT-3.5 (1), with Mistral (0.87-0.95) shows under performed results.
- **Contextual Precision**: Despite these strengths, Llama3-8B (0.85-0.938) and Mistral (0.8-0.94) fall short in contextual precision compared to GPT-3.5 (0.98).

These findings suggest that open-source models like Llama3-8B and Mistral offer notable improvements over GPT-3.5, particularly in contextual recall and relevancy,answer relevancy and unigram precision, though they may still have some limitations in contextual precision [17].

IV. Discussion

As highlighted in Section III, the Llama3-8B model demonstrates superior performance compared to the Mistral 8x7B

TABLE I: Performance Metrics for Llama3-8B

Metrics	*Reason Dense*	*Reason Sparse*	*Factual Dense*	*Factual Sparse*
Unigram Precision				
Average	0.737	0.789	0.81	0.82
Median	0.709	0.774	0.8	0.81
Contextual Precision				
Average	0.911	0.938	0.864	0.85
Contextual Recall				
Average	0.92	0.98	0.916	0.92
Contextual Relevancy				
Average	0.68	0.6363	0.66	0.947
Median	1	1	1	1
Answer Relevancy				
Average	0.98	0.93	0.97	1
Time (s)				
Average	2.88	2.732	1.6004	1.74
Median	2.61	2.25	1.472	1.401
CSGA Range				
Range	[0.875, 0.975]	[0.87, 0.97]	[0.81, 1]	[0.8, 0.98]

TABLE II: Performance Metrics for Mistral 8x7B

Metrics	*Reason Dense*	*Reason Sparse*	*Factual Dense*	*Factual Sparse*
Unigram Precision				
Average	0.69	0.67	0.73	0.73
Median	0.74	0.72	0.754	0.756
Contextual Precision				
Average	0.9	0.94	0.8	0.85
Contextual Recall				
Average	0.94	0.98	0.93	0.91
Contextual Relevancy				
Average	0.6	0.56	0.73	0.92
Median	1	0	1	1
Answer Relevancy				
Average	0.94	0.87	0.95	0.91
Time (s)				
Average	2.73	2.91	1.8	1.6
Median	2.96	3.06	1.4	1.36
CSGA Range				
Range	[0.87, 0.975]	[0.89, 0.98]	[0.84, 1]	[0.84, 0.98]

model across various tasks. This advantage is likely attributed to Llama3-8B's training on an extensive and diverse dataset encompassing over 15 trillion tokens. Additionally, the Llama3-8B model benefits from instruction-tuning, a process that optimizes it for tasks requiring adherence to user instructions, thereby enhancing its effectiveness for RAG-based applications. It is crucial to note that the results presented in this study are specific to the datasets used and should not be generalized to other datasets. Furthermore, from a cost-efficiency standpoint, these open-source alternatives offer significant cost reductions.

From the top-k vs. cosine similarities graph, it can be inferred that beyond a certain top-k value, the retrieved information becomes increasingly irrelevant to the query. This irrelevance leads to a plateau in the graph across all question sets for both models, indicating that additional documents do not contribute to the query's answer.

For reasoning-dense questions, the Llama3 model effectively utilized the retrieved context better than the Mistral model, suggesting that models with a smaller context window can sometimes use information more efficiently than those with a larger context window. However, for factually dense questions, the Mistral model benefited from the additional retrieved chunks, showing a steady increase in cosine similarity for two of the questions which can be attributed the mixture of experts being a effective way here.

In reasoning-sparse questions, the Llama3 model consistently utilized the retrieved information more effectively than the Mistral model, which showed significant variations across different top-k values. This variation indicates that Mistral did not utilize the retrieved information as efficiently, although both models eventually achieved similar precision scores.

For factually sparse questions, both models performed similarly as top-k increased for two questions, with a notable increase for one question. This result suggests that for factually sparse questions, both models were able to utilize the retrieved information to a comparable extent.

To compare the quality of answers generated by an open-source LLM and GPT-4, as we can infer from Figure1 cosine similarity proves to be a reliable metric. This metric remains

relatively stable even with variations in the top-k parameter, exhibiting minimal changes (typically around 0.5 and a maximum of 1) across different questions. In contrast, unigram recall and precision, which measure the proportion of single words correctly matched between the generated and reference answers, show greater variability with average changes near 1 and maximum changes up to 2. This variability suggests that unigram recall and precision are less reliable for assessing answer quality compared to the consistent performance of cosine similarity with ground truth answer.

As we can infer from Table I and Table II, the values of Cosine similarity with the groundtruth answer (CSGA) do not vary much across all categories of question. Also, these values compare well with Contextual Recall, Contextual Precision and Answer relevancy metrics of Deepeval framework. This consistency makes CSGA a reliable metric. Besides, cosine similarly is easier to calculate compared to Deepeval , which utilizes four evaluation metrics and requires significant inference time for every question. Additionally, we can also infer from appendix that the average time taken for Llama3 and Mistral using perplexity API is nearly 50% lower compared that of GPT-3.5 .

V. Conclusion

This work investigates the efficacy of open source LLMs in providing response to questions related to enterprise specific data. For the same, vector database using open source embedding has been created followed by categorising the questions into 4 categories viz. sparse factual, sparse reason, dense factual and dense reason. Here are some salient findings from this investigation:

- **Effectiveness of Open-Source LLMs in RAG Systems**: The study demonstrates that open-source LLMs integrated within Retrieval-Augmented Generation (RAG) framework, generate response of similar accuracy and relevance of as commercial LLMs.
- **Influence of context length**: RAG based QA evaluation by increasing provided context (by varying top-k) demonstrates no significant improvement in answer quality as CSGA does not change much . Thus, this evaluation on enterprise specific data shows that one need not have very large LLM context window for Question Answering (QA) task.
- **Llama3-8B vs mistral8x7B**: LLM parameter count need not necessarily improve RAG based Question Answering (QA) , as evident from Llama3 outperforming Mistral. Especially for proprietary enterprise datasets as it's difficult for RAG based systems to perform on them over regular open source data sets available online.
- **Performance**: The use of open-source LLMs to build RAG based QA system provides performance similar to commercial LLMs. Besides, this work also demonstrates that open source LLMs can be scaled and adapted to enterprise-specific data sets without need of investing in expensive Graphical Processing Unit(GPU)s for real time inferencing.
- **Cosine Similarity with Groundtruth answer (CSGA)** : is an effective metric for measuring answer quality of RAG answers.

Acknowledgment

The authors thank I-Venture at Indian School of Business for infrastructural support toward this work. Authors are extremely grateful to Prof. Bhagwan Chowdhry, Faculty Director (I-Venture at ISB) and Rahul Sundar(Scientist at Indian Institute of Technology, Madras) for their continued encouragement and support to carry out this research.

References

[1] 2023. BAAI embedding documentation. (2023). Retrieved from https://huggingface.co/BAAI/bge-large-en-v1.5.
[2] 2024. deepeval documentation. (2024). Retrieved from https://docs.confident-ai.com/.
[3] 2024. langchain documentation. (2024). Retrieved from https://python.langchain.com/v0.1/docs/get_started/introduction.
[4] 2024. NLTKsplitter documentation. (2024). Retrieved from https://api.python.langchain.com/en/latest/nltk/langchain_text_splitters.nltk.NLTKTextS
[5] 2024. RecursiveCharacterTextSplitter langchain documentation. (2024). Retrieved from https://python.langchain.com/v0.1/docs/modules/data_connection/document_transforme
[6] 2024. huggingface documentation. (2024). Retrieved from https://huggingface.co/models.
[7] 2024. FAISS documentation. (2024). Retrieved from https://python.langchain.com/v0.2/docs/integrations/vectorstores/faiss/.
[8] 2024. Perplexity wrapper code. (2024). Retrieved from https://github.com/amaze18/dlabs_hybrid_search/blob/main/pplx.py.
[9] 2024. perplexity api documentation. (2024). Retrieved from https://docs.perplexity.ai/docs/getting-started.
[10] 2024. Perplexity models documentation. (2024). Retrieved from https://docs.perplexity.ai/docs/model-cards.
[11] 2024. OpenAI API pricing documentation. (2024). Retrieved from https://openai.com/api/pricing/.
[12] 2024. Perplexity pricing documentation. (2024). Retrieved from https://docs.perplexity.ai/docs/pricing.
[13] 2024. neuralmagic documentation. (2024). Retrieved from https://docs.neuralmagic.com/get-started/finetune/.
[14] 2024. lancedb documentation. (2024). Retrieved from https://blog.lancedb.com/optimizing-llms-a-step-by-step-guide-to-fine-tuning-with-peft-and-qlora-22eddd13d25b/.
[15] 2024. BM25 documentation. (2024). Retrieved from https://python.langchain.com/v0.2/docs/integrations/retrievers/bm25/.
[16] Shahul Es, Jithin James, Luis Espinosa-Anke, and Steven Schockaert. 2023. Ragas: automated evaluation of retrieval augmented generation. *arXiv preprint* arXiv::2309.15217. Retrieved from https://arxiv.org/pdf/2309.15217.
[17] Kush Juvekar and Anupam Purwar. 2024. Cos-mix: cosine similarity and distance fusion for improved information retrieval. Retrieved from https://arxiv.org/pdf/2406.00638.
[18] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint* arXiv::2004.04906. Retrieved from https://arxiv.org/pdf/2004.04906.
[19] Omar Khattab and Matei Zaharia. 2020. Colbert: efficient and effective passage search via contextualized late interaction over BERT. *arXiv preprint* arXiv::2004.12832. Retrieved from https://arxiv.org/pdf/2004.12832.
[20] Patrick Lewis et al. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. *arXiv preprint* arXiv::2005.11401.
[21] Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2023. Mteb: massive text embedding benchmark. *arXiv preprint* arXiv::2210.07316. Retrieved from https://arxiv.org/pdf/2210.07316.
[22] Micah Musser. 2023. A cost analysis of generative language models and influence operations. *arXiv preprint* arXiv::2308.03740. Retrieved from https://arxiv.org/pdf/2308.03740.

[23] Anupam Purwar and Rahul Sundar. 2023. Keyword augmented retrieval: novel framework for information retrieval integrated with speech interface. *arXiv preprint* arXiv::2310.04205. Retrieved from https://dl.acm.org/doi/10.1145/3639856.3639916.
[24] Keshav Rangan and Yiqiao Yin. 2024. A fine-tuning enhanced RAG system with quantized influence measure as AI judge. *arXiv preprint* arXiv::2402.17081v1. Retrieved from https://arxiv.org/pdf/2402.17081v1.
[25] Yi Tay et al. 2023. UL2: unifying language learning paradigms. *arXiv preprint* arXiv::2205.05131.
[26] Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-pack: packaged resources to advance general Chinese embedding. *arXiv preprint* arXiv::2309.07597. Retrieved from https://arxiv.org/pdf/2309.07597.
[27] Hao Yu, Aoran Gan, Kai Zhang, Shiwei Tong, Qi Liu, and Zhaofeng Liu. 2024. Evaluation of retrieval-augmented generation: a survey. *arXiv preprint* arXiv::2405.07437v1. Retrieved from https://arxiv.org/pdf/2405.07437v1.
[28] Lianmin Zheng et al. 2024. Judging LLM-as-a-judge with MT-bench and Chatbot Arena. Retrieved from https://arxiv.org/pdf/2306.05685.