2025 IEEE 17th International Conference on Computer Research and Development (ICCRD) | 979-8-3315-3188-1/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICCRD64588.2025.10962837

Comparative Analysis of Advanced RAG Techniques using Mahabharata

Ganesh Vaidyanathan K
Dept. of CSE
PES University
Bengaluru, India
kganeshv12@gmail.com

Varun M S
Dept. of CSE
PES University
Bengaluru, India
varun80042@gmail.com

Shauryadeepsinh Gajendrasinh Raolji,
Dept. of CSE
PES University
Bengaluru, India
raoljishauryadeep@gmail.com

Bhaskarjyoti Das
Dept. of CSE, AI&ML
PES University
Bengaluru, India
bhaskarjyoti01@gmail.com

***Abstract*—Retrieval Augmented Generation (RAG) is a hot topic and one of the most compute-effective context-extending techniques in the LLM industry. Unfortunately, RAG also faces certain challenges, which make it vulnerable to mistakes. To overcome these challenges, recent studies have come up with numerous methods, which are collectively referred to as advanced RAG. In our study, we test out the effectiveness of individual Advanced RAG methods and compare them when used with a variety of LLMs. We perform this comparative study using the Mahabharata, the Indian Epic, as the external knowledge base. Examining how Advanced RAG techniques can be used in practical applications is this paper's primary contribution. We then test the efficacy of the techniques using answer-independent evaluation metrics. We conclude by tabulating the performance metrics recorded and provide insights on the results obtained.**

***Index Terms*—Retrieval Augmented Generation, Advanced RAG, Comparative Analysis, Mahabharata, LLMs**

I. Introduction

Natural language processing was transformed by the development of Large Language Models (LLMs), which allowed machines to generate text that is remarkably coherent and fluent, much like that generated by a human. With its ability to have conversations and demonstrate the promise of generative AI, ChatGPT's [1] release greatly advanced the industry. Despite being competent, LLMs faced the problem of restricted context and hallucinations [30]. Retrieval-Augmented Generation (RAG) [33] was developed to overcome this in LLMs by fusing generative models with external knowledge retrieval. RAG became a key breakthrough to improve contextual comprehension and factual accuracy in LLMs. Because RAG seamlessly integrates other databases into language generation activities, it is now a key component of applications that demand accurate, context-aware responses.

RAG, with little addition of external context and suitable prompting, is now able to overcome the primary hurdles that LLMs have been facing. Now, RAG is able to prove useful in chatbots as recommender systems, assistants, and so on. Recent studies have demonstrated that RAG has the potential to be used for general language tasks, a variety of downstream applications, and knowledge-intensive jobs like Open-domain Question Answering. It has also been shown that RAG is a promising solution to hallucination, the primary issue [37].

Despite tackling many issues faced by LLMs, RAG is still not perfect [19]. It has its own set of challenges. Aspects like information integration, answering complex questions, dealing with false information retrieved, etc. can become breaking points of RAG. [21]

Another study revealed various failure points through out the RAG pipeline. Right from the knowledgebase all the way till the output answer, this paper shows that there is scope for improving the architecture of naive RAG. [19] The recent survey conducted on RAG [29] separates the RAG pipeline into Pre-retrieval, retrieval and Post-retrieval phases. They further explain how one or more 'Advanced RAG' techniques are able to enhance and optimize the respective phases.

Through our literature survey, we observed that there are very few papers that compare the various Advanced RAG techniques against each other for task-specific use cases. The term 'Advanced RAG', here refers to the techniques and methods employed to improve 'Naive RAG'. In our paper, we study and implement some of these techniques and evaluate their relative performance.

The primary objectives of this research are:

1) To conduct a comprehensive comparative analysis of advanced RAG techniques across multiple LLM architectures
2) To test the RAG techniques across different categories of questions and difficulty levels pertaining to the Mahabharatha.
3) To evaluate the performance of the technique-LLM-question combination across different RAG evaluation metrics.

Aragog [22] is an interesting paper, which aims to compare few of the ARAG methods and test their performance and efficiencies. Though our aim is similar, we intend to perform a broader and deeper study while trying to answer the following questions :

RQ1: How do individual advanced RAG techniques perform?

RQ2: How does the category of the question affect the performance of the Advanced Technique?

RQ3: How do different LLMs affect the RAG performance?

RQ4: How do the individual RAG techniques perform across different evaluation metrics?

All the codes used in this research are available on our Github Repository [27].

II. Theoretical Framework

Our main work revolves around comparing the various Advanced RAG techniques. Naive RAG forms the foundation and the core of our study, which is why we will be exploring the aspects of Naive RAG at first.

A. NAIVE RAG

Naive RAG is a core technique that marries a retrieval component with a generative model, often some large LLM, to answer user queries. In this framework, the retriever scans an extensive corpus of documents to pick out pertinent information in line with the input query. These identified documents are then handed over to the LLM, which analyzes the content of the documents to produce a coherent and contextually suitable reply. The Naive RAG framework is somewhat simple, and this has the advantage of making its implementation possible; thus, it is a great option in many applications of NLP. However, effectiveness for this framework is still very dependent on the quality of documents retrieved by the retrieval mechanism for their queries. If the retrieved documents fail to adequately satisfy the query or are irrelevant, the quality of the responses produced by the generative model may also be an issue. This limitation makes Naive RAG relatively less robust than more advanced RAG techniques which include developments like enhanced retrievals techniques, query expansion, or better document ranking systems. Such advanced approaches cope with some of the shortcomings in Naive RAG through improvement in the retrieval phase, thus improving the overall output produced. Subsequently, while Naive RAG provides a naive and efficient method, its sensitivity to the quality of the retrieved documents makes it prone to problems like information sparsity and ambiguity.

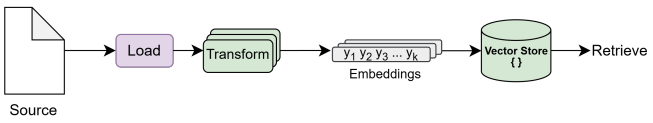

Fig. 1. Workflow of RAG

B. ADVANCED RAG

1) MULTI QUERY RETRIEVER: The Multi-query retriever is a complex methodology targeted for the improvement process of prompt tuning using a large language model to generate several unique queries for a specific search task across an information retrieval assignment. This method makes the generation of queries focused and better with more informative extraction. Multiple queries retriever has the most important advantage because it can answer intricate questions that depend on one or several subquestions that call for different views or dimensions of information. It achieves this by concurrently producing and executing several queries and accesses a wider scope of relevant content while providing a richer and broader range of results. This parallel querying, in addition to enhancing the depth of information retrieved, also reduces the likely risk of missing critical details that may only be known through specific search angles. The use of an LLM in generating these queries ensures that the varied perspectives are only contextually relevant as well as aligned with the user's informational needs. This aspect notably elevates the overall effectiveness of information retrieval systems especially in complex or multidimensional queries. In terms of operational efficiency and effectiveness, the Multi-query retriever results make it an asset for any field, be it in research and data analysis, or applied directly in the practical executions found on various AI systems [8].

2) RAG FUSION: RAG Fusion is a high-ranking aggregation method combining the capabilities of a retrieval-augmented generation and reciprocal rank fusion for information retrieval and document ranking accuracy optimization. The method improves the effectiveness of RAG since it presents multiple queries, depending on user input, and the algorithm will reorder the generated results using their respective relevance scores. These queries are designed to capture documents from various viewpoints, thus raising the scope of relevant material under consideration. After the retrieval of documents, the RAG Fusion puts together the results and their scores to combine them in ranked aggregation. It incorporates the ideas of Reciprocal Rank Fusion, which aggregates the rankings acquired from the various sources of retrieval based on the strength of their ranks in reciprocal values, so that the documents ranked favorably across multiple sources of retrieval gain greater significance. Thus, the aggregation of several sources of retrieval yields more robust and accurate rankings because it minimizes any bias or constraint that may exist in any single source of retrieval. By systematically combining the documents along with their score values, RAG Fusion ensures that the ranking generated at the final stages comprises more informative relevance assessment. This effectively enhances the effectiveness of information retrieval systems where several retrieval approaches are conducted simultaneously with each other [36].

3) DECOMPOSITION: One of the significant problem-solving techniques used in RAG is decomposition. Decomposition refers to the process by which a large, complex question or task is broken down into several smaller, more manageable sub-questions. This makes the problem easier to handle because it focuses on individual components that can be answered precisely. In RAG, the relevant chunks of information about each sub-query are returned by this model, which enables it to work on smaller, context-specific pieces of the overall problem. This process of segmentation allows for the improvement of the effectiveness of the model in dealing with parts of the problem in isolation and not getting confused

by the processing of a large complex query all at once. Decomposition also aids more precise response generation, wherein the model can aggregate the insights derived from each of these sub-queries in a step-by-step manner, resulting in a much more coherent and contextually relevant answer. Further, by addressing each of the sub-parts individually, the system is far better enabled to navigate the relevant data and zoom into the most relevant information that pertains to that specific aspect of the task. This in addition to improving the retrieval efficiency ensures that the final response is not only more accurate and robust but also aligned with the requirement of multi-step reasoning tasks [20].

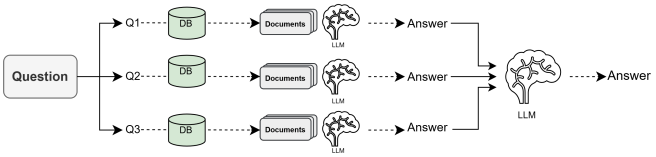

Fig. 2. Working of Decomposition method

4) STEP BACK PROMPT: Step Back Prompting is the prompting methodology to promote the abstracting capabilities of LLMs as it tries to make them step away from minute specifics to more generalized high-level ideas and first principles. In Step Back Prompting, the model is guided to 'step back' away from the particularities of a problem and instead reflect at the core principles underlying the provided instances. The Step Back Prompting thereby frames the problem in terms of more general concepts, making it possible for the model to comprehend the situation more deeply and to extract relevant, fundamental insights from the problem. This enables LLMs to approach reasoning in a more structured and logical manner, thus improving their capability to continue along a correct path toward a solution. The technique fosters more nuanced understanding as it focuses on the role of context and conceptual connections rather than memorizing details. With this, Step Back Prompting has proven to be an effective guide in helping LLMs steer toward more efficient and coherent problem-solving, especially for complex queries that need deeper cognitive abstraction. This technique significantly enhances the reasoning power of LLMs by exploiting high-level abstractions that result in more accurate and contextually appropriate responses in various applications [39].

5) HyDE: The HyDE technique, introduced by Gao et al. (2022) [28] is an innovative approach for improving document retrieval systems by exploiting the abilities of Large Language Models (LLMs). The heart of the idea behind HyDE lies in utilizing LLMs to generate hypothetical answers to user queries simulating potential responses the system might return based on the content of the documents in question. These hypothetical answers are created with rich context, giving the system the capacity to capture nuances and deeper meanings in the query. Once generated, the answers are embedded, which means that they turn into numerical representations that can be analyzed further by the system. The embedding procedure enables the system to streamline and narrow down the document retrieval, which allows only the most relevant documents to be retrieved based on the hypothetical answers produced by the LLM. HyDE enhances retrieval precision and reduces reliance on traditional keyword-based matching by using hypothetical answers as a bridge between the query and the document content. Therefore, this method is a step further ahead in utilizing semantic understanding in document retrieval- it would allow for deeper and more contextually relevant results. HyDE, above all, does not just help in improving retrieval systems' efficiency; it also adds to the growing field of information processing based on semantic understanding.

6) SEMANTIC ROUTING: Semantic routing is an advanced RAG technique used in retrieval that depends on semantic understanding to guide incoming queries toward the most suitable response or action. Unlike the traditional methodologies, which mostly rely on keyword matching, semantic routing prioritizes the meaning behind the queries so systems can better grasp the intent and context of the user's request. This approach basically represents a layer of decision in the system that guides the queries in relevance to their importance. Semantic routing can analyze, based on the semantic structure of the input, the details and underlying context behind queries and get a relevant response. Moreover, semantic routing reduces misclassifications and redundant queries, thus improving the scalability and efficiency of systems. It, therefore, forms an indispensable component in the design of intelligent, context-aware applications that more easily transcend from keyword-based interaction [38].

7) CHUNKING: Chunking is an important NLP and information retrieval technique that breaks large text documents into small, manageable segments or 'chunks.' This process improves the relevance of content retrieved from a vector database in a way that supports better searching and information extraction. The two main governing parameters within the chunking process are chunk size and chunk overlap. Chunk size refers to the number of words that should comprise each chunk, and alterations to the value will directly impact the chunks generated; smaller chunk sizes will result in a higher number of chunks, while larger chunk sizes only produce fewer, larger chunks. Overlap between chunks, on the other hand, determines how much contiguous chunks share common text. A larger overlap in chunks would generate more significant overlap between those chunks, providing better contextual continuity, and a smaller overlap provides the minimum redundancy but may sometime not ensure continuous context between two adjacent chunks. Overall, it is very crucial to find a balance between chunk size and overlap for the trade-off between granularity and computational efficiency. Properly optimized chunking improves the quality of content retrieval since the aligned textual segments should better express the query submitted by the user, reducing ambiguity and enhancing the relevance in search results. To make effective use of chunking, therefore, understanding and fine-tuning these parameters are a prerequisite [32].

8) RERANKING: Reranking is generally a vital step in information retrieval. This is because the accuracy in search results becomes more enhanced through refining the pool of retrieved documents. It denotes reordering the document chunks initially retrieved according to their relevance for the user query, thus reducing the total set of documents for further processing. This mechanism is both a filtering mechanism that limits the pool of documents as well as an enhancer to feed richer inputs to subsequent language models for processing. Reranking can be done in many ways, either in specialized models like the Cohere Reranker, or directly using large language models. The Cohere Reranker utilizes the architecture of the cross-encoder, which evaluates the query and document chunks together. This way, it is possible to derive a more mature understanding of the relevant query and document connections. Since LLM-based reranking exploits the stronger capacity of LLMs in terms of contextual relationship analysis and relevance judgment, it uses the advanced LLM capacities. Both increase the retrieval process in the sense that the returned documents are most relevant; hence, they guarantee the accurate processing of language models by Cohere Reranker [16].

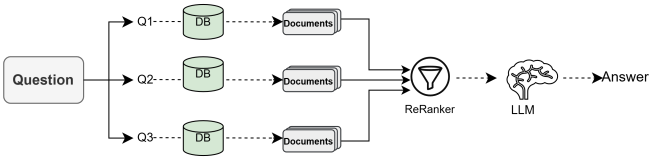

Fig. 3. Working of Reranker method

9) SENTENCE WINDOW RETRIEVER: Sentence window retrieval is an advanced technique of information retrieval in which a set of contextually rich sentences surrounding a matched sentence in the answer space is returned in response to a user query to enhance relevance and accuracy. Unlike classical retrieval methods that typically return a single sentence based on keyword matching, this approach extracts a window of surrounding sentences, giving the needed contextual structure in order to gain better understanding of the information contained within the document. This technique is quite valuable within scenarios where the user query is complex, such as questions or phrases that actually demand deeper context to ultimately comprehend. For example, if the query requires an explanation or a detailed response, more information than what is offered by a single sentence may be necessary. The system returns a set of more relevant sentences and lets the language model (LLM) generate more detailed output that is nuanced and gives more insight. More context has helped in the interpretation of minor relationships and clarifying ambiguities to further ensure that the answer aligns with the broader context of the document. Retrieval accuracy and generation quality are both improved, with the use of sentence window retrieval enabling it to be both informative and contextual. Sentence window size is thus dependent on query nature, and due to this, balance between depth of context and retrieval specificity can be found, making optimal performance in any application, such as question answering, possible [17].

10) PARENT DOCUMENT RETRIEVER: When splitting documents for retrieval, the choice of what the ideal chunk size would be optimal for embeddings for document splitting and retrieval. The smaller chunks make for embeddings that are semantically relevant to what they're about because they're talking about something specific, directed, with smaller information; yet they simultaneously are de-contextualized as they exist apart from larger ideas and do not necessarily connect to their other, like ideas in the larger text. The larger chunks make for embeddings that are contextualized as larger ideas, which is good because it maintains continuity across like ideas and the larger narrative document trajectory; yet, the embeddings are less semantically relevant to the smaller bits of information as the larger concepts drown out any smaller, significant relevance. The Parent Document Retriever seeks to remedy this. It chunks documents into a lot of smaller pieces for semantic relevance at the same time each chunk is always associated with its parent document—from where it came. So in retrieval, the Parent Document Retriever retrieves those smaller, semantics-based chunks but also retrieves the parent document ids associated with those chunks, bringing up accompanying parent document text in the retrieval output. So it can give the smaller, contextualized meaning but also surround it with a richer context of surrounding information from its parent document. Therefore, this is the effective solution because it can have small contextual relevance with large context to bolster the retrieval output [9].

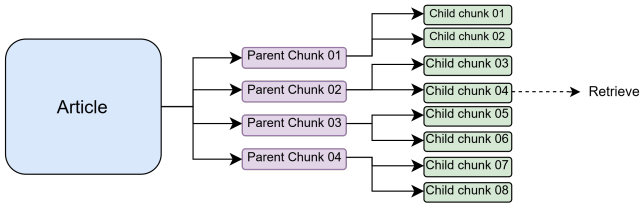

Fig. 4. Working of Parent Document Retriever Method

III. METHODOLOGY

In this paper, we carry out a four-fold comparative study, as can be seen in Fig. 5. Let us consider the top-down sequence. We compare various 'Advanced RAG' techniques on different Large Language models. Each of the above combination needs to answer a certain set of questions, which are further subdivided into categories based on type and difficulty. And all of these combination are evaluated on 3 different metrics. Having discussed the Advanced RAG techniques that we will be experimenting with, we now move onto the other requirements:

A. Advanced RAG methods Used

The Advanced RAG methods that we used for the comparative analysis can be found in Table I :

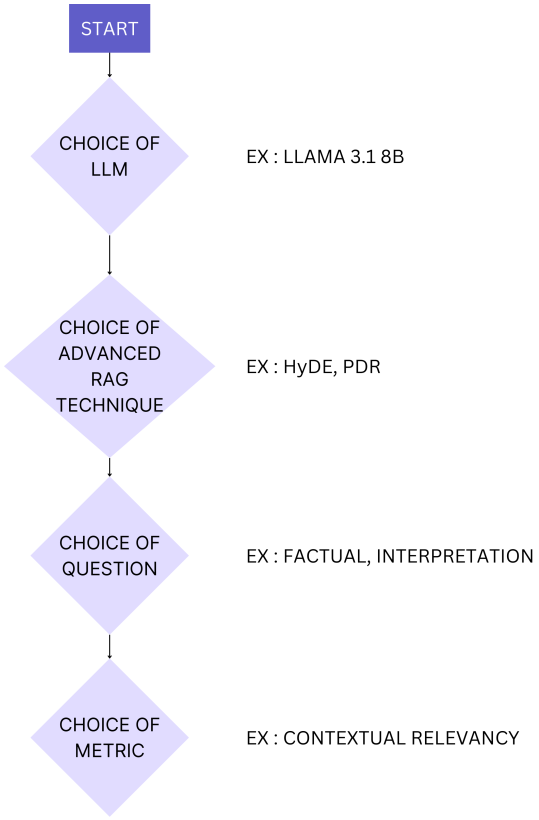

Fig. 5. Four-fold Comparison study

TABLE I
ADV RAG METHODS USED FOR COMPARISON

LLMs Used
HyDE
Naive RAG
RAG Fusion
Decomposition
Semantic Routing
Step Back Prompting
Sentence Window Retriever
Parent Document Retriever
RAG with Low Chunk size & overlap

B. LLMs Used

Cloud-hosted LLMs will be used for investigation. All the LLMs used are Open-Source and are sourced from Groq AI [7]. The following table II has the list of LLMs used along with their parameter count:

For our comparative study, we make use of open-source LLMs right from 1B parameter model all the way up to 70B parameter model. All the above LLMs have been configured with *max_tokens* as 8192 and a temperature of *0.1*. Our intention is to keep the temperature minimal so as to give precedence to precise answers over creative ones, as the LLMs need to adhere to the content present in the knowledge base [35].

TABLE II
LLMS USED FOR COMPARISON

LLMs Used	No. of Parameters
Llama 3.2 [13]	1B
Llama 3.2 [13]	3B
Mixtral 8x7B [31]	7B
Gemma [26]	7B
Llama 3.1 [24]	8B
Gemma 2 [25]	9B
Llama 3.1 [24]	70B

C. Choice of Dataset

This study, as previously mentioned, aims to test Advanced RAG techniques on real-life applicability and effectiveness. To accomplish the above, we had two effective options: 1. Find a task-dependent dataset that can put the method to test on specific categories. or 2. Use a dataset that encompasses most, if not all, categories to test the Advanced RAG method. We believe the Indian epic 'Mahabharata' will be an ideal fit for the purpose. Mahabharata is one such piece of literature that encompasses an answer to almost every question with respect to life [34]. The Mahabharata is not just a war between the Good and Evil, it is in a way a multifaceted case study, delving into topics from politics and warfare to social norms and righteousness. It has an interplay of complex emotions, coupled with a long list of characters, which makes it even more challenging to comprehend. The term 'Dataset' is not really applicable in our study.

We from now on refer to the data source as a knowledge base, as it acts as the external knowledge base from which the RAG system retrieves relevant context. We make use of the cleaned and preprocessed version of the mahbharata released by Project Gutenberg [15]. It is important to note that the original version of the Mahabharata is written in Sanskrit. This study makes use of its English adaptation as provided in Project Gutenberg's resources.

We further add a few extra resources to our knowledge-base to enhance the ability to deal with Interpretation and Reasoning based questions. These resources contain 'lessons', which can be used as retrieved knowledge. The resources are : 'Important Lessons from Mahabharata' [18] and '31 Lessons of Management from Mahabharata' [12].

As for the Data preprocessing, we scrape the source [15] into text files. As the entire Mahabharata is divided into 18 *Parvas*, we scrape them into 18 text files, one for each *Parva*. We then perform Data cleaning techniques like stop word removal and data cleaning to ensure that the Data is in a consumable format.

D. Evaluation Metrics

In this study we are making use of 'Answer-Independent' RAG evaluation metrics. Initially proposed in Ragas [23], these metrics are able to judge the model without the need to rely on ground truth human annotations. This suite of

evaluation metrics are better suited, as it is often difficult to obtain answers to given questions, especially if the dataset is massive. Also, there are multiple cases when there are many answers to one single question, which makes answer-based evaluation tough and meaningless. On the same lines, any modern-day Advanced RAG application will be dealing with a decently sized knowledge base. Accounting for the above, we decide to use the following metrics for our evaluation :

1) Answer Relevancy Metric: The answer relevancy metric is a crucial component in evaluating the performance of a Retrieval-Augmented Generation (RAG) pipeline, particularly in the context of applications that utilize large language models (LLMs). This metric assesses the quality of the generated output by comparing it to the input provided to the system. Essentially, it measures how well the response generated by the LLM aligns with the expectations set by the input query or prompt.

In practical terms, the answer relevancy metric operates by analyzing various aspects of the output, such as its coherence, accuracy, and relevance to the original question or request. A high relevancy score indicates that the generated response effectively addresses the input, providing useful and pertinent information. Conversely, a low score suggests that the output may be off-topic, inaccurate, or lacking in detail, which can hinder the overall effectiveness of the RAG pipeline. We use it directly from DeepEval's Python library [3].

The metric is calculated as follows :

$$AnswerRelevancy = \frac{NumberofRelevantStatements}{TotalNumberofStatements}$$

2) Faithfulness Metric: The faithfulness metric is an essential component in evaluating the performance of a Retrieval-Augmented Generation (RAG) pipeline. This metric specifically assesses the quality of the generator by determining whether the output it produces aligns accurately with the information contained in the retrieval context. In other words, it examines whether the generated responses are not only coherent and relevant but also factually correct and grounded in the data retrieved from the source materials.

The faithfulness metric serves as a vital tool for assessing the reliability of a RAG pipeline's outputs, ensuring that the generated content is not only relevant but also accurately reflects the information retrieved. We use it directly from DeepEval's Python library [6].

The metric is calculated as follows :

$$Faithfulness = \frac{NumberofTruthfulClaims}{TotalNumberofClaims}$$

3) Contextual Relevancy Metric: The contextual relevancy metric is a crucial tool for evaluating the effectiveness of your Retrieval-Augmented Generation (RAG) pipeline, particularly focusing on the performance of the retriever component. This metric provides insights into how well the information retrieved aligns with the specific needs of a given input query.

To understand its functionality, it is helpful to compare it to the Answer Relevancy Metric. While the Answer Relevancy Metric directly assesses the relevance of a provided answer to a question, the Contextual Relevancy Metric takes a more nuanced approach. It begins by leveraging a large language model (LLM) to extract all relevant statements or pieces of information from the retrieval context. This context typically consists of a collection of documents or data points that the retriever has gathered in response to the input query.

Once the LLM has identified and extracted these statements, the next step involves evaluating their relevance to the original input. The same LLM is employed to analyze each extracted statement, determining whether it is pertinent to the input query. This dual-step process allows for a more comprehensive assessment of the contextual relevance, as it not only considers the statements in isolation but also evaluates them in relation to the specific context of the input. We use it directly from DeepEval's Python library [5].

The metric is calculated as follows :

$$ContextualRelevancy = \frac{NumberofRelevantStatements}{TotalNumberofStatements}$$

E. Questions for Testing

To evaluate an Advanced RAG application that will be used in varied contexts, spanning multiple domains, we needed different categories of questions to test the model with. This carefully designed test set for a Mahabharata RAG (Retrieval-Augmented Generation) model offers a unique way to assess how well an AI system grasps this intricate epic. These questions showcase a multi-layered testing method with several key characteristics:

Organised Sructure The test questions are neatly divided into three main categories:

1) Factual Questions
2) Inference/Interpretation Questions
3) Detailed/Long Answer Questions

Gradual Difficulty Each of the above questions is further divided into three difficulty levels:

1) Easy
2) Intermediate
3) Hard

Based on the above, here are the following questions we have handpicked to test the *Method-LLM* combination :

Factual Questions :

1) *Easy* - Who was the eldest among the Pandava brothers?
2) *Intermediate* - What were the conditions of the dice game between the Pandavas and Kauravas, and what was wagered in each round?
3) *Hard* - Detail the various divine weapons (astras) received by Arjuna during his training, from whom he received them, and under what circumstances?

Interpretation/Analytical Questions :

1) *Easy* - Why did Krishna choose to be Arjuna's charioteer rather than fight in the war himself?
2) *Intermediate* - How does Bhishma's vow of celibacy and his devotion to his father's happiness reflect the theme of duty versus personal desire in the Mahabharata?
3) *Hard* - Analyze how the concept of Time (Kala) is portrayed as both a destroyer and preserver throughout the epic, using specific examples from different parvas.

Long Answer/In-Detail Questions :

1) *Easy* - WDescribe the events leading up to and including the burning of the Khandava Forest. What were its immediate and long-term consequences?
2) *Intermediate* - Compare and contrast the characters of Karna and Arjuna, analyzing their similarities, differences, and how their parallel journeys contribute to the epic's themes.
3) *Hard* - Examine the role of women in the Mahabharata, focusing on Draupadi, Kunti, and Gandhari. How do their actions and choices drive the narrative and reflect the epic's views on gender, power, and dharma?

Here, we further want to explain the intent and some background for choosing *Mahabharata* as our evaluatory knowledge base. Mahabharata, as seen from a literary point of view [10], contains a vast range of themes, complex characters, psychological depth, and cultural and social context that is even applicable in today's world! Not to forget the moral and ethical complexity, wherein the characters are not simply categorized as good or bad; instead, their actions and decisions are shown to have complex motivations.

1) **Factual Knowledge**: The Mahabharata is a legendary Indian epic credited to the sage Vyasa, containing more than 100,000 verses. It narrates the tale of the power struggle for the throne of Hastinapura, mainly between the Pandavas and the Kauravas.
2) **Understanding of the Narrative**: The main storyline of the Mahabharata focuses on the Kurukshetra War, featuring key moments like Arjuna's ethical crisis and Lord Krishna's teachings in the Bhagavad Gita. Additional plots, such as the dice game and the Pandavas' exile, add depth to the characters and their destinies.
3) **Philosophical Insights**: The Mahabharata delves into significant philosophical ideas like dharma (righteousness), karma (the effects of actions), and moksha (spiritual liberation). The Bhagavad Gita, which is part of the Mahabharata, presents a conversation between Arjuna and Krishna about duty, ethics, and the essence of life and death.
4) **Character Evaluations**: The characters in the Mahabharata are richly layered, with Yudhishthira embodying virtue and Duryodhana representing ambition and pride. Lord Krishna serves a dual purpose as both a divine mentor and a strategist, shaping the events with his clever insights.
5) **Thematic Analysis**: Major themes in the Mahabharata include the battle between good and evil, the significance of righteousness, and the unavoidable outcomes of war. The epic also examines fate, justice, family loyalty, and the intricacies of moral decisions in challenging situations.
6) **Cultural and Historical Background**: The Mahabharata is set in a period often thought to reflect a historical age of ancient Indian kingdoms, even though its events are rooted in mythology. It showcases the societal norms and political systems of ancient India, highlighting kingship, family relationships, and the principles of justice and dharma within the framework of Vedic culture.

IV. Experimental Setup

We conduct our experiments on Intel i5 12th Generation CPU, consumer-grade laptops. For all our Large Language models, and Vector Databases we make use of OpenAI embeddings [2], namely *text-embedding-3-small*. As for our vector database, we use the Persistent Store from *Chroma DB* [4]. In terms of chunking, we use chunk_size as 1000 and chunk_overlap as 200 as standard. For the reduced chunk size experiment, we use chunk_size as 200 and chunk_overlap as 20. For all the LLMs we use *max_tokens* as 8000. For Parent Document Splitter, we use *parent_chunk_size* as 4000, *parent_chunk_overlap* as 300, *child_chunk_size* as 400 and *child_chunk_overlap* as 30. We make use of the Chroma DB *persistent store* for storing the child chunks and an *InMemoryStore* for storing the parent chunks. We use LangChain [11] to implement all of the methods used except Sentence Window Retriever [17], for which we make use of LlamaIndex [14]. We use a *sentence_window_size* of 3 for the Sentence Window Retriever.

V. Results & Analysis

The results have been recorded in table III and table IV. The results have been calculated for each *method-LLM-question-metric* combination. The LLMs labelled L1 to L7 are : Gemma 7B, Gemma 2 9B , Llama 3.1 8B, Llama 3.1 70B, Llama 3.2 1B, Llama 3.2 3B, Mixtral 8x7B. Due to the lack of space to show all obtained values, we split the results into question-based and metric-based results. The highest value obtained for 'long answer questions' and 'answer relevancy' metrics are highlighted using **bold**, highest value obtained for 'interpretation questions' and 'context relevancy' metrics are highlighted using *italics* and 'factual questions' and 'faithfulness' metrics are highlighted with underline.

A. Based on Category of Questions :

The results obtained while categorizing based on Questions can be found in table III. Since there are 3 questions in each category and there are 3 metrics for each question, we first calculate the average score for each metric across the 3 questions in a category and then take the average across the 3 metrics. For example, we calculate the average of the faithfulness metric for the 3 questions pertaining to, say, the category 'Factual Questions'. Next, we take the average of the

TABLE III
TABLE FOR COMPARISON BETWEEN METHODS AND MODELS BASED ON TYPES OF QUESTIONS.

Methods	Category of Questions	LLMs used						
		L1	L2	L3	L4	L5	L6	L7
Naive RAG	Factual	0.551	0.551	0.183	0.551	0.183	0.183	0.183
	Interpretation	*0.871*	0.871	*0.769*	*0.871*	*0.769*	*0.*769	0.769
	Long Answer	0.863	0.863	0.774	0.863	0.774	0.774	0.774
RAG Fusion	Factual	0.277	0.440	0.548	0.277	0.548	0.548	0.548
	Interpretation	0.604	*0.780*	0.711	0.604	0.711	0.711	0.711
	Long Answer	0.738	0.806	0.777	0.738	0.777	0.777	0.777
Decomposition	Factual	0.584	0.584	0.492	0.492	0.563	0.563	0.609
	Interpretation	0.733	0.733	0.550	0.550	0.677	0.677	*0.831*
	Long Answer	0.57	0.757	0.855	0.855	0.698	0.698	0.893
Step Back Prompt	Factual	0.702	0.702	0.515	0.515	0.515	0.515	0.515
	Interpretation	0.734	0.734	0.711	0.711	0.711	0.711	0.711
	Long Answer	**0.868**	**0.868**	0.806	0.806	0.806	0.806	0.806
HyDE	Factual	0.314	0.440	0.314	0.314	0.314	0.314	0.314
	Interpretation	0.694	*0.780*	0.694	0.694	0.694	0.694	.0694
	Long Answer	0.765	0.806	0.765	0.765	0.765	0.765	0.765
Semantic Routing	Factual	0.515	0.515	0.515	0.515	0.515	0.515	0.515
	Interpretation	0.711	0.711	0.711	0.711	0.711	0.711	0.711
	Long Answer	0.806	0.806	0.806	0.806	0.806	0.806	0.806
Chunking-Low	Factual	0.702	0.702	0.702	0.702	0.348	0.348	0.348
	Interpretation	0.734	0.734	0.734	0.734	0.574	0.574	0.574
	Long Answer	**0.868**	**0.868**	**0.868**	**0.868**	**0.812**	**0.812**	**0.812**
Sentence Window Retriever	Factual	0.702	0.702	0.492	0.492	0.515	0.515	0.515
	Interpretation	0.734	0.734	0.550	0.550	0.711	0.711	0.711
	Long Answer	**0.868**	**0.868**	0.855	0.855	0.806	0.806	0.806
Parent Document Retriever	Factual	0.492	0.492	0.492	0.492	0.492	0.492	0.492
	Interpretation	0.594	0.594	0.594	0.594	0.594	0.594	0.594
	Long Answer	0.741	0.741	0.741	0.741	0.741	0.741	0.741

3 metrics, namely: faithfulness, answer relevancy & Context relevancy, to obtain an individual decimal value presented in the table.

With reference to Table III, we can summarize the results as follows :

Naive RAG

- Naive RAG shows that chunking plays a significant role, with higher chunking resulting in better performance.
- The Gemma family and LLaMA 70B outperform other models in this setup.
- Interpretation-based questions have higher accuracy compared to other categories.

RAG Fusion

- The LLaMA family of models, except for 70B, performs better than others.
- In this method, long-answer questions seem to be more accurate compared to other question types.

Decomposition

- The Mixtral and Gemma models perform well, with Mixtral having a significant lead over others.
- Gemma models excel in answering interpretation-based questions, while Mixtral and LLaMA perform better on long-answer questions.

Step Back Prompt

- Gemma models outperform all other models, ranking high across all categories of questions.
- The highest accuracy is observed for long-answer questions.

TABLE IV
TABLE FOR COMPARISON BETWEEN METHODS AND MODELS BASED ON TYPES OF METRICS.

Methods	Category of Metrics	LLMs used						
		L1	L2	L3	L4	L5	L6	L7
Naive RAG	Faithfulness	0.949	0.949	0.695	0.949	0.695	0.695	0.695
	Contextual Relevancy	*0.457*	*0.457*	0.382	*0.457*	0.382	0.382	0.382
	Answer Relevancy	0.879	0.879	0.648	0.879	0.648	0.648	0.648
RAG Fusion	Faithfulness	0.865	0.830	0.822	0.865	0.822	0.822	0.822
	Contextual Relevancy	0.161	0.339	0.377	0.161	0.377	0.377	0.377
	Answer Relevancy	0.594	0.856	0.837	0.594	0.837	0.837	0.837
Decomposition	Faithfulness	0.962	0.962	0.916	0.916	0.747	0.747	0.913
	Contextual Relevancy	0.354	0.354	0.348	0.348	*0.488*	*0.488*	*0.646*
	Answer Relevancy	0.759	0.759	0.634	0.634	0.702	0.702	0.775
Step Back Prompt	Faithfulness	0.910	0.910	0.726	0.726	0.726	0.726	0.726
	Contextual Relevancy	0.395	0.395	*0.404*	0.404	0.404	0.404	0.404
	Answer Relevancy	**1.0**	**1.0**	0.904	0.904	**0.904**	**0.904**	**0.904**
HyDE	Faithfulness	0.733	0.830	0.733	0.733	0.733	0.733	0.733
	Contextual Relevancy	0.356	0.339	0.356	0.356	0.356	0.356	0.356
	Answer Relevancy	0.685	0.856	0.685	0.685	0.685	0.685	0.685
Semantic Routing	Faithfulness	0.726	0.726	0.726	0.726	0.726	0.726	0.726
	Contextual Relevancy	0.404	0.404	*0.404*	0.404	0.404	0.404	0.404
	Answer Relevancy	0.904	0.904	0.904	0.904	**0.904**	**0.904**	**0.904**
Chunking-Low	Faithfulness	0.910	0.910	0.910	0.910	0.606	0.606	0.606
	Contextual Relevancy	0.395	0.395	0.395	0.395	0.484	0.484	0.484
	Answer Relevancy	**1.0**	**1.0**	**1.0**	**1.0**	0.644	0.644	0.644
Sentence Window Retriever	Faithfulness	0.910	0.910	0.916	0.916	0.726	0.726	0.726
	Contextual Relevancy	0.395	0.395	0.348	0.348	0.404	0.404	0.404
	Answer Relevancy	**1.0**	**1.0**	0.634	0.634	**0.904**	**0.904**	**0.904**
Parent Document Retriever	Faithfulness	0.872	0.872	0.872	0.872	0.872	0.872	0.872
	Contextual Relevancy	0.134	0.134	0.134	0.134	0.134	0.134	0.134
	Answer Relevancy	0.821	0.821	0.821	0.821	0.821	0.821	0.821

HyDE

- All models perform equally, with Gemma 9B significantly better than the rest.
- These models are poor at answering factual questions but perform well on long-answer questions.

Semantic Routing

- All models perform equally well across all categories of questions.
- This suggests that semantic routing is largely independent of the underlying LLM.

Chunking (Low)

- The Gemma and LLaMA 3.1 family of models outperform others.
- These models perform equally well across all categories, with long-answer questions showing the best results.

Sentence Window Retriever

- The Gemma family of models significantly outperforms the rest.
- These models perform equally well across all categories, with long-answer questions being the most accurate.

Parent Document Retriever

- All models perform equally well across all categories of questions.
- This indicates that PDR is independent of the underlying LLM.

B. Based on Category of Metrics :

The results obtained while categorizing based on Metrics can be found in table IV.

Since there are 9 questions and there are 3 metrics for each question, we calculate the average score for each metric across the 9 questions for every *Method-LLM-Metric* combination.

This value is shown in table IV, we can summarize the results as follows :

Naive RAG

- **Faithfulness:** Performs consistently well with high scores (0.949) across the Gemma family and LLama3.1 8B, but struggles with others.
- **Contextual Relevancy:** Moderate performance, with a noticeable drop for all models. It shows weaknesses in maintaining relevant context.
- **Answer Relevancy:** Reasonably high scores for Gemma family and LLama 70B, but dips significantly for others, indicating variability in effectiveness.

RAG Fusion

- **Faithfulness:** Consistent but slightly lower scores compared to Naive RAG, with values ranging from 0.822 to 0.865.
- **Contextual Relevancy:** Mixed performance; lower for Gemma 7B and LLama 70B, but improves with other models.
- **Answer Relevancy:** Relatively strong and consistent scores for most models, peaking at 0.856 for Gemma 9B.

Decomposition

- **Faithfulness:** One of the strongest methods, achieving top scores for most models.
- **Contextual Relevancy:** Moderate-to-good performance, with scores improving for smaller LLamas and Misxtral models.
- **Answer Relevancy:** Performs well but not outstanding, with scores hovering in the mid-range.

Step Back Prompt

- **Faithfulness:** Scores are decent but not leading, with consistent values across models.
- **Contextual Relevancy:** Overall good and slight improvement over others, indicating steady relevancy.
- **Answer Relevancy:** Exceptional performance, achieving perfect scores across multiple models.

HyDE

- **Faithfulness:** Gemma2 9B outperforms all the others on faithfulness and answer relevancy.
- **Observations:** Changes in LLMs do not reflect changes in the metrics obtained.

Semantic Routing

- **Overall Performance:** Competitive performance across all metrics.
- **Observations:** Changes in LLMs do not reflect changes in the metrics obtained.

Chunking Low

- **Faithfulness:** Good initial performance from Gemma and LLama but drops significantly for the others.
- **Contextual Relevancy:** Smaller LLama models and Misxtral perform better in contextual relevancy.
- **Answer Relevancy:** Exceptional performance for the Gemma and LLama3.1 family, achieving perfect scores, but drops significantly for others.

Sentence Window Retriever

- **Faithfulness:** Strong and consistent performance, comparable to Decomposition for certain models.
- **Contextual Relevancy:** Moderate performance but consistent.
- **Answer Relevancy:** Excellent scores, with perfect results (1.0) for the Gemma family; other models also perform well except the LLama3.1 family.

Parent Document Retriever

- **Overall Performance:** Really strong performance for faithfulness and answer relevancy.
- **Observations:** Changes in LLMs do not affect the results.

VI. Discussion

The results point out significant discrepancies in the performance of high-end RAG techniques across distinct question types and models while providing essential insights into how they could be practically applicable and their limitations. Among these, Naive RAG and RAG Fusion techniques demonstrated the extent to which effective chunking along with aggregation of retrieved content indeed impacts models such as Gemma and LLaMA 70B to a great degree, especially for interpretation-based, long-answer questions as well as metrics such as faithfulness and answer relevancy, which means retrieval granularity directly impacts relevance and completeness for retrieved content.

Decomposition methods proved particularly efficient for the complex, multi-step reasoning task, with Mixtral outperforming other models because of its ability to process segmented queries and synthesize coherent responses. Similarly, Step Back Prompting proved to be effective in guiding LLMs towards a high-level reasoning approach, which improves accuracy for long-answer questions but may result in loss of detail for factual questions and requires careful prompt engineering. This is proven by exceptional performance in faithfulness and answer relevancy.

Semantic Routing and Parent Document Retriever methods excelled in LLM independence, as their performance was stable across different architectures. This may imply that these techniques can be a good set of robust frameworks for general-purpose applications where model-specific tuning is not possible. They, however, showed some weakness in the processing of subtle factual questions, pointing to possible places for further refinement. They perform consistently well across metrics such as faithfulness and answer relevancy but

semantic routing takes an edge over contextual relevancy, where parent document retriever performs poorly.

Notably, HyDE and Chunking (low) underscored a compromise between retrieval efficiency and the comprehensiveness of context. Although HyDE improved retrieval precision via the use of hypothetical embeddings, it faced challenges regarding factual accuracy, thereby emphasizing the importance of context-aware integration. This is further proved by their low scores on answer relevancy and moderate scores in faithfulness and context relevancy. Conversely, low chunking methods demonstrated effectiveness in models such as Gemma and LLaMA 3.1, achieving exceptionally high scores in faithfulness and answer relevancy. But necessitated careful adjustment of chunk size and overlap to achieve optimal outcomes as other models had mediocre performance.

In general, the results indicate that there is an essential task-specific alignment of advanced RAG techniques with architectures of LLMs. Hybrid methods combining the advantages of different techniques, making the most of their complementarities, may further enhance retrieval quality as well as generation quality in future studies. Finally, factually, the errors across these techniques are areas to improve on so that the methods are applicable in knowledge-intensive domains.

VII. Conclusion

The comparative analysis of advanced Retrieval-Augmented Generation (RAG) techniques demonstrates that different methods and models exhibit unique strengths across varying question categories. The Gemma family of models consistently outperformed others, particularly excelling in long-answer questions, regardless of the technique employed. LLaMA models, including the 70B variant, showed competitive performance in methods like Naive RAG, while smaller variants performed better in RAG Fusion and Decomposition. Semantic Routing and PDR proved to be LLM-independent, achieving uniform performance across all models and question types. Conversely, HyDE and Step Back Prompting favored specific model architectures, with the latter enhancing reasoning and abstraction. Chunking parameters significantly impacted retrieval efficacy, with lower chunking yielding improved performance in certain configurations.

Overall, the findings underscore the importance of aligning RAG techniques and LLM architectures with the nature of the task, suggesting that tailored approaches can maximize accuracy and contextual relevance in real-world applications. We believe that all our collected data and research insights are useful resources not only for better understanding of existing Advanced RAG methods and its characteristics but also to propel further analysis and research in the field of Retrieval Augmented Generation.

References

[1] https://openai.com/index/chatgpt/. [Accessed 29-11-2024].

[2] https://platform.openai.com/docs/models#embeddings. [Accessed 29-11-2024].

[3] Answer Relevancy — DeepEval - The Open-Source LLM Evaluation Framework — docs.confident-ai.com. https://docs.confident-ai.com/docs/metrics-answer-relevancy. [Accessed 26-11-2024].

[4] Chroma Docs — docs.trychroma.com. https://docs.trychroma.com/. [Accessed 29-11-2024].

[5] Contextual Relevancy — DeepEval - The Open-Source LLM Evaluation Framework — docs.confident-ai.com. https://docs.confident-ai.com/docs/metrics-contextual-relevancy. [Accessed 26-11-2024].

[6] Faithfulness — DeepEval - The Open-Source LLM Evaluation Framework — docs.confident-ai.com. https://docs.confident-ai.com/docs/metrics-faithfulness. [Accessed 26-11-2024].

[7] GroqCloud — console.groq.com. https://console.groq.com/docs/models. [Accessed 25-11-2024].

[8] How to use the MultiQueryRetriever. https://python.langchain.com/docs/how_to/MultiQueryRetriever/. Accessed: 2024-11-24.

[9] How to use the Parent Document Retriever — LangChain — python.langchain.com. https://python.langchain.com/docs/how_to/parent_document_retriever/. [Accessed 29-11-2024].

[10] ijhssi.org. https://www.ijhssi.org/papers/v6(6)/Version-3/C0606031213.pdf. [Accessed 26-11-2024].

[11] LangChain — langchain.com. https://www.langchain.com/. [Accessed 29-11-2024].

[12] Lessons of Management from Mahabharata &x2013; Human values and indian Ethos — ebooks.inflibnet.ac.in. https://ebooks.inflibnet.ac.in/hrmp01/chapter/235/. [Accessed 25-11-2024].

[13] Llama 3.2 — Model Cards and Prompt formats — llama.com. https://www.llama.com/docs/model-cards-and-prompt-formats/llama3_2/. [Accessed 29-11-2024].

[14] LlamaIndex - LlamaIndex — docs.llamaindex.ai. https://docs.llamaindex.ai/en/stable/. [Accessed 29-11-2024].

[15] Mahabharata — gutenberg.org. https://www.gutenberg.org/files/19630/19630-h/19630-h.htm. [Accessed 25-11-2024].

[16] Rerank. https://cohere.com/rerank. Accessed: 2024-11-24.

[17] Sentence window retriever - LlamaIndex — docs.llamaindex.ai. https://docs.llamaindex.ai/en/stable/api_reference/packs/sentence_window_retriever/. [Accessed 29-11-2024].

[18] admin. Lessons From Mahabharata - Lonely Philosopher — lonelyphilosopher.com. https://www.lonelyphilosopher.com/lessons-from-mahabharata/. [Accessed 25-11-2024].

[19] S. Barnett, S. Kurniawan, S. Thudumu, Z. Brannelly, and M. Abdelrazek. Seven failure points when engineering a retrieval augmented generation system, 2024.

[20] C.-M. Chan, C. Xu, R. Yuan, H. Luo, W. Xue, Y. Guo, and J. Fu. Rq-rag: Learning to refine queries for retrieval augmented generation. *arXiv preprint arXiv:2404.00610*, 2024.

[21] J. Chen, H. Lin, X. Han, and L. Sun. Benchmarking large language models in retrieval-augmented generation, 2023.

[22] M. Eibich, S. Nagpal, and A. Fred-Ojala. Aragog: Advanced rag output grading, 2024.

[23] S. Es, J. James, L. Espinosa-Anke, and S. Schockaert. Ragas: Automated evaluation of retrieval augmented generation, 2023.

[24] A. G. et al. The llama 3 herd of models, 2024.

[25] G. T. et al. Gemma 2: Improving open language models at a practical size, 2024.

[26] G. T. et al. Gemma: Open models based on gemini research and technology, 2024.

[27] GaneshVaidyanathanK. Comparative-analysis-of-advanced-rag-techniques-using-mahabharata. https://github.com/kganeshv12/Comparative-Analysis-of-Advanced-RAG-Techniques-using-Mahabharata, 2024. Accessed: 2024-12-23.

[28] L. Gao, X. Ma, J. Lin, and J. Callan. Precise zero-shot dense retrieval without relevance labels. *arXiv preprint arXiv:2212.10496*, 2022.

[29] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang. Retrieval-augmented generation for large language models: A survey, 2024.

[30] M. U. Hadi, R. Qureshi, A. Shah, M. Irfan, A. Zafar, M. B. Shaikh, N. Akhtar, J. Wu, S. Mirjalili, et al. A survey on large language models: Applications, challenges, limitations, and practical usage. *Authorea Preprints*, 2023.

[31] A. Q. Jiang. Mixtral of experts, 2024.

[32] A. Kshirsagar. Enhancing rag performance through chunking and text splitting techniques. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 10(5):151–158, Nov. 2024.

[33] 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*, 33:9459–9474, 2020.
[34] H. Pallathadka and L. Pallathadka. India through mahabharata: A critical view. *European Journal of Molecular & Clinical Medicine*, 7(11):8257–8268, 2020.
[35] M. Peeperkorn, T. Kouwenhoven, D. Brown, and A. Jordanous. Is temperature the creativity parameter of large language models? *arXiv preprint arXiv:2405.00492*, 2024.
[36] Z. Rackauckas. Rag-fusion: a new take on retrieval-augmented generation. *arXiv preprint arXiv:2402.03367*, 2024.
[37] K. Shuster, S. Poff, M. Chen, D. Kiela, and J. Weston. Retrieval augmentation reduces hallucination in conversation, 2021.
[38] T. Waitzenberg. Mastering RAG Chatbots: Semantic Router—RAG gateway— Part 1 — talon8080. https://medium.com/@talon8080/mastering-rag-chatbots-semantic-router-rag-gateway-part-1-0773cf4e70ad. [Accessed 29-11-2024].
[39] H. S. Zheng, S. Mishra, X. Chen, H.-T. Cheng, E. H. Chi, Q. V. Le, and D. Zhou. Take a step back: Evoking reasoning via abstraction in large language models. *arXiv preprint arXiv:2310.06117*, 2023.