An End-to-End Framework Towards Improving RAG (Retrieval-Augmented Generation) Based Application Performance

Aritra Sen, Anindita Desarkar and Vishwanathan Raman

December 20, 2024

An End-to-End Framework towards Improving RAG (Retrieval-Augmented Generation) based Application Performance

Aritra Sen[1], Anindita Desarkar[2] and Vishwanathan Raman[3]

[1] Enterprise AI, LTIMindtree Ltd, Kolkata, India, aritra_sen@outlook.com
[2] Enterprise AI, LTIMindtree Ltd, Kolkata, India, aninditadesarkar@gmail.com
[3] Enterprise AI, LTIMindtree Ltd, Chennai, India, vishwanathanraman@gmail.com

Abstract. *Retrieval Augmented Generation (RAG) is a framework designed to address the limitation of Large Language Models (LLM) in terms of business domain awareness and knowledge cutoff. Hence, the adoption of RAG has been immense in recent times as it can overcome the above challenges. However, RAG also consists of several techniques and challenges. A few common challenges include being unable to produce optimal response or output format mismatch, missing to refer most important sources and incapable of retrieving the appropriate paragraphs or contexts. As a result, the response accuracy of RAG based applications deteriorates. Hence a recommendation system is the need of the hour which can assist the users to choose the most appropriate method based on the specific scenario. In this paper, an end-to-end RAG-based application performance improvement framework is proposed which will assist the users to select the optimal approach based on the present evaluation score and other constraints. Evaluation score is calculated based on the well-known metrices which include groundedness, answer relevance and context relevance. The framework is a collection of different techniques applied iteratively at different stages of RAG with the goal of improving the overall score. The embedding being the backbone of the RAG, a right fit embedding model recommendation is part of the overall framework.*

Keywords: Embedding model selection, large language model (LLM), Retrieval Augmented Generation (RAG) improvement framework, Vector database.

1 Introduction

Recent advancement in the field of Generative AI has resulted in a significant surge in the adoption of Large Language Models (LLMs) across various domains and industry.
A powerful single LLM excels at various tasks like summarization, text classification and conversational tasks but suffers from the problem of hallucinations which occurs when the LLM does not have the correct context to answer the given user query. The reason may be lack of domain knowledge or the correct responses are beyond the knowledge cut-off date.

Retrieval Augmented Generation (RAG) [1] is the most well-known technique which enables the LLM to go beyond the data on which they have been trained on. RAG systems have several underlying benefits like enhanced privacy and transparency due to its architectural design, in turn these benefits have increased the adoption of RAG across the industry.

The outcome of RAG based application depends on selection of values of its features made during the application design process. Features include embedding model selection, optimal chunking approach and choosing the advanced RAG strategy. The automatic enhancement of the quality of LLM response happens while these parameters are chosen wisely. Also, it's an iterative approach as respective methods need to be adopted more than one time in various phases of the cycle based on RAG based application evaluation score and other system constraints. However, the number of iterations can be reduced if some standard recommendation engine exists in the eco-system.

This has *motivated the current researchers* to propose the end-to-end RAG method adoption framework for using the best technique considering the current scenario and other user constraints towards improving the application performance. The framework consists of several techniques including in-context learning, embedding model selection, choosing optimal chunk strategy, hybrid search, using metadata filters, multi-query retriever, hypothetical document embeddings (Hyde), reranking, response synthesis, sentence window retrieval and auto merging retrieval. However, choosing the right embedding model always throws a challenge to the users compared to other ones. The availability of several embedding models makes the task more difficult as it confuses the users whether the selection should be random, or parameter driven. There are few embedding model's leader boards which kind of give an idea of leading models, however those leaderboards generally do not cater to complex use case need or user requirements. Hence, a novel method for the embedding model selection process is also presented in the paper which is a part of the proposed holistic framework. This in turn will reduce the confusion of the developers or end users on the selection of embedding model, so that developers can solely concentrate on building advanced text-based AI solutions.

The paper is organized as follows. Section 2 highlights the existing techniques for advanced RAG architecture and available frameworks for adopting the same in various business scenarios. The problem statement is summarized in section 3. Section 4 presents the failure point analysis of generic RAG architecture. Outline of the proposed approaches are presented in Section 5. Section 6 describes the solution approach in detail whereas the experimental results are summarised in Section 7. Section 8 contains the conclusions drawn based on the aforesaid research and investigations along with future directions.

2 Literature Review

Recent advances in the LLM space like ChatGPT [2] and LLaMA [3] have shown that increasing the model parameters is directly correlated with LLM performance improvement. However, RAG based systems are essential to enable the LLM based systems to go beyond the training data. RAG evaluation [4] is the first step towards improving the system performance. RAGAS [5] is one such system where researchers came up with three main evaluation parameters namely faithfulness, answer relevancy and context relevancy. These evaluation criteria give adequate directions to the user towards improving the RAG based application system.

Scott Barnett et. al. [14] have found that RAG systems can have several failure points in different stages of the pipeline. In the literature, they have mainly discussed seven failure points which are related to two stages: Index process and Query process. These failure points motivate the current researchers to design an optimal RAG system.

Xiaohua Wang et. al. [15] have explored various processing steps present in RAG architecture, each of which can be executed in various ways. They have investigated existing RAG approaches and their potential combinations to identify optimal practices. As a result, they have suggested several strategies for deploying RAG that balance both performance and efficiency.

Cheonsu Jeong [16] have leveraged agentic framework to evaluate the reliability of RAG systems and synthesize diverse data to generate more accurate and enhanced responses. The author also has demonstrated the graph-based agentic RAG system, along with specific algorithm and validation results. This has demonstrated the feasibility of an enhanced RAG system.

Florin Cuconasu et. al. [17] have considered key factors like relevance of the chunks included as context, their position, and their number. Research observation includes a counter-intuitive finding that retriever's highly relevant documents can negatively impact the effectiveness of RAG systems. Authors also have found that by adding random documents in the prompt improves the LLM accuracy by up to 35%.

Researchers have come up with several new advanced RAG techniques [6] like Query rewriting [7], HyDE [8] etc. which have shown significant performance improvements of RAG systems in terms of previously defined parameters. However, a complete RAG improvement framework is the need of the hour, and the paper tries to address the gap by proposing the same. Without the recommendation system, it would be just choosing the technique randomly and perform trial and error. It may not be a feasible approach as the operating cost will be too high.

In terms of embedding models, prior research work have shown the huge number of state of the art models are available in the market. Researchers also came up with evaluation metrices like Massive Text Embedding (MTEB) benchmarking [9] to evaluate and compare embeddings model with the existing benchmark.

In one of the literature [10], researchers looked into the desired properties of the word embedding and evaluation techniques using two types intrinsic and extrinsic. The study tried to offer a valuable guidance in selecting suitable evaluation methods for different application tasks.

Jean-Baptiste Excoffier et. al. [11] have concluded based on their research that generalist embedding models perform better than specialized ones at short-context clinical semantic search. They curated a textual dataset on clinical code and found that specialized embedding models are more sensitive to small changes in input.

The literature review clearly shows the need of an holistic RAG enhancement framework as no such concrete one is available. This has motivated the current researchers to propose an end-to-end autonomous framework for improving the performance of retrieval augmented generation-based (RAG based) applications with minimal user intervention which can tackle the different failure points [14] in a systematic way. The framework can select the appropriate techniques in each level based on the user requirements and system constraints.

3 Problem Statement

In the field of Generative AI, RAG is one of the most powerful approaches which combines the strength of language models and information retrieval system so that meaningful and contextual responses can be generated. However, several practical challenges are present in the system so that always desired outcome cannot be achieved through RAG methodology. Few such challenges or open questions are presented in the following which have motivated the current researchers to design the proposed framework.

A. What can be the best chunking strategy of documents while reading them as input?

B. How can RAG system interpret the user query optimally?
C. What is the best way to store documents in the database which leads to better retrieval and performance generation?
D. How to assign priority to a specific part of datasets which contain the most relevant and trustworthy information?
E. Is there any way that system can re-rank the retrieved results based on user requirements like semantic similarity, source credibility, or task-specific relevance?
F. How the retrieved output can be more meaningful in the context of user query?
G. How to integrate content from various input documents?
H. Is there any way that redundancy can be removed from the final response?
I. How to receive quality content which has the requisite depth and free from inconsistencies?
J. Is there any way to check whether the correct part of the document extracted in the process?

Additionally, using the embedding model is a must in all RAG based applications which also throws lots of confusion to the users. The following presents a few of them. As a result, an embedding model selection framework is included as a part of the proposed framework.

A. Is there any Embedding Model which can help to get better accuracy for the selected use case?
B. How to address the latency constraint in the project?
C. What is the correct embedding model to support big size documents taken as input?
D. How to choose the embedding model which can handle complex scenarios?
E. Whether all the embedding models need GPU support, or any one is available which does not require it?
F. How to take care of data privacy and security as it should not be exposed to the outside world?
G. How to perform the embedding on non-English text?

4 Failure Point Analysis in Basic RAG based Applications

Retrieval augmented generation, or RAG, is an architectural concept which can enhance the efficacy of large language model (LLM) applications by leveraging custom data. The custom data consists of user provided data in a document which is fed as an input to the system. The following Figure 1 presents the first part of the architecture which is *Ingestion* where user provided documents are divided into smaller chunks, transformed into embedding vector and finally kept in the vector database.

In the second part, depicted in Figure 2 shows the *Retrieval* and *Synthesis* process where appropriate K chunks are retrieved from the vector database based on user query and sent to LLM for evaluation. LLM performs the synthesis based on user query and generates the final response based on the given context.

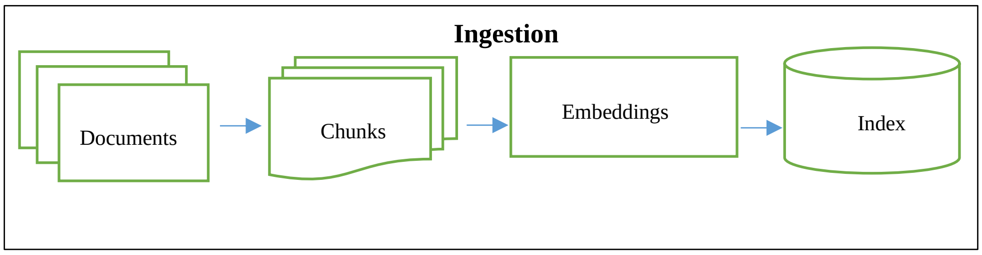


Fig. 1: RAG Architecture: Ingestion

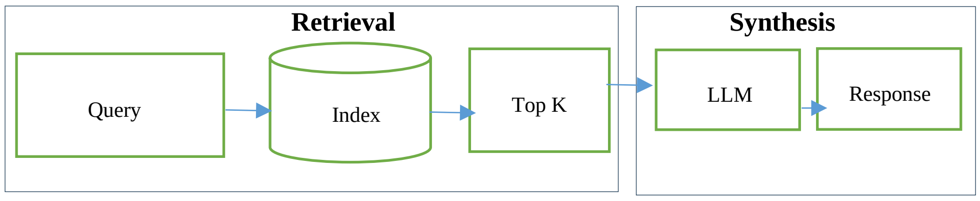


Figure 2: RAG Architecture: Retrieval and Synthesis

But there are several practical challenges in vanilla RAG which are presented in the following. However, the proposed framework will try to address the challenges to a meaningful extend.

A. **Missing Content:** One common case in RAG where incorrect answers can appear, is retrieval of improper chunks as those are not related to the question exactly.

B. **Missing Top Ranked Documents:** The other scenario can be non-retrieval of most relevant documents where relevance can be semantic similarity, source credibility, or task-specific relevance.

C. **Limitation in Consolidation Strategy:** In few cases, multiple documents can be associated for generating the final response where appropriate consolidation takes a vital place as quality of final response depends how effectively the chunks are merged together.

D. **Problem in Extraction Format:** This refers to the problem where LLM output does not adhere the instructions given by the user; which means output response format differs from the requested format.

E. **Incorrect Specificity:** This situation happens when the answer is returned but is not specific enough or is too specific to address the user's need. It occurs when users are not sure about the correct question or provided in a generic format.

F. **Incomplete Response:** It means that answers are not incorrect but does not contain all the relevant points to make it complete though those are present in the input document.

In summary, following are the limitations found in three stages:

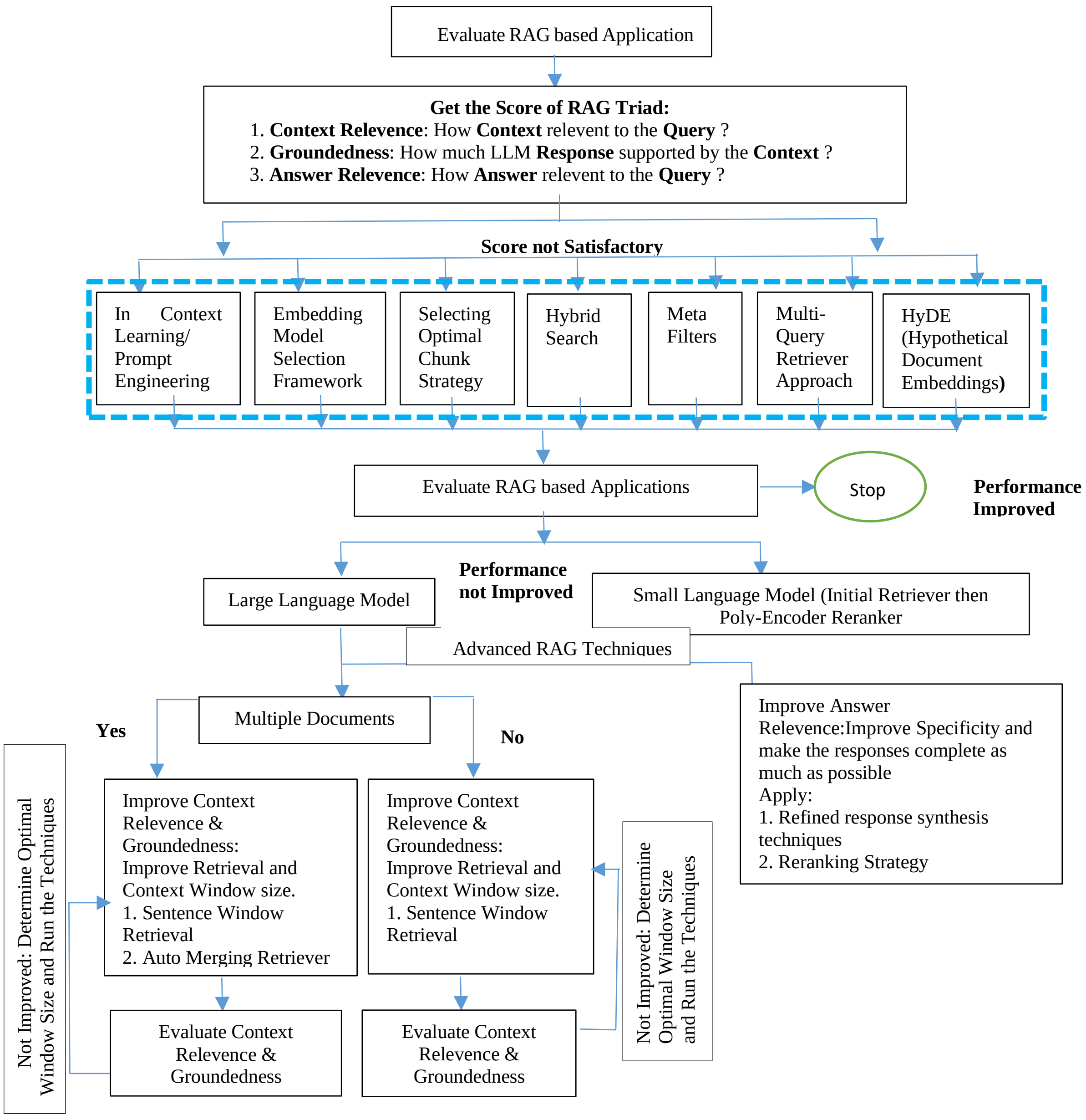
- **Retrieval Stage** - Difficulties in perfect data retrieval and the response relevance.
- **Augmentation Stage** - Emphasizing the complexities in synthesizing multi-document information and managing conflicting data.
- **Generation Stage** - Underscoring the need for responses that are contextually complete and specific.

5 Outline of Proposed Framework

The following two frameworks are proposed to address the above set of problems where an end-to-end recommendation system of various RAG techniques will help the end users to select the appropriate RAG technique based on the evaluation metrices and the other one will recommend the optimal embedding model which is the heart of any RAG based architecture. The end objective is improving the accuracy of RAG based applications.

A. Framework for Improving RAG-based Application Performance

The following Figure 3 presents the outline of the proposed framework for RAG-based application performance improvement.



Evaluate RAG based Application
Get the Score of RAG Triad:
1. **Context Relevence**: How **Context** relevent to the **Query** ?
2. **Groundedness**: How much LLM **Response** supported by the **Context** ?
3. **Answer Relevence**: How **Answer** relevent to the **Query** ?
Score not Satisfactory
In Context Learning/ Prompt Engineering
Embedding Model Selection Framework
Selecting Optimal Chunk Strategy
Hybrid Search
Meta Filters
Multi-Query Retriever Approach
HyDE (Hypothetical Document Embeddings)
Evaluate RAG based Applications
Stop
Performance Improved
Performance not Improved
Large Language Model
Small Language Model (Initial Retriever then Poly-Encoder Reranker
Advanced RAG Techniques
Multiple Documents
Yes
No
Improve Context Relevence & Groundedness: Improve Retrieval and Context Window size. 1. Sentence Window Retrieval 2. Auto Merging Retriever
Improve Context Relevence & Groundedness: Improve Retrieval and Context Window size. 1. Sentence Window Retrieval
Improve Answer Relevence:Improve Specificity and make the responses complete as much as possible Apply: 1. Refined response synthesis techniques 2. Reranking Strategy
Evaluate Context Relevence & Groundedness
Evaluate Context Relevence & Groundedness
Not Improved: Determine Optimal Window Size and Run the Techniques
Not Improved: Determine Optimal Window Size and Run the Techniques

Figure 3. Outline of Proposed framework for RAG-based application performance improvement

B. Embedding Model Recommendation System

The following Figure 4 presents the outline of the proposed framework for Embedding Model Recommendation System.

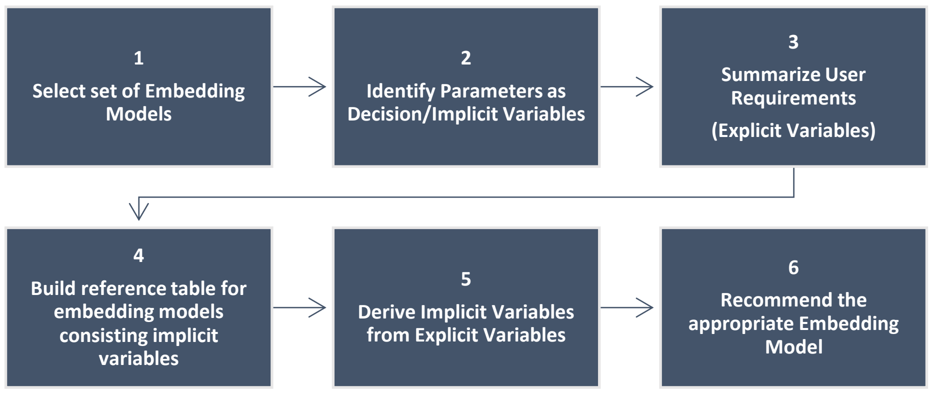

1
Select set of Embedding Models
2
Identify Parameters as Decision/Implicit Variables
3
Summarize User Requirements
(Explicit Variables)
4
Build reference table for embedding models consisting implicit variables
5
Derive Implicit Variables from Explicit Variables
6
Recommend the appropriate Embedding Model

Figure 4. Outline of Proposed Embedding Model Selection Framework

6 Solution Approach

A. RAG based Application Improvement Framework

The proposed framework presented in above Figure 3 has the following major components or approaches.

- **Configuration File:** The configuration file enables end users to select different selection criteria like LLM, embedding model, chunk size, chunking strategy along with other details like document path, document type and vector store persistence path etc. It provides the end users the flexibility to try out different parameters to build a RAG system.

- **Agent based Approach**: Based on an Agentic approach, the Agent will take different routes to populate the vector DB as shown below Figure 5 according to the configuration parameters.

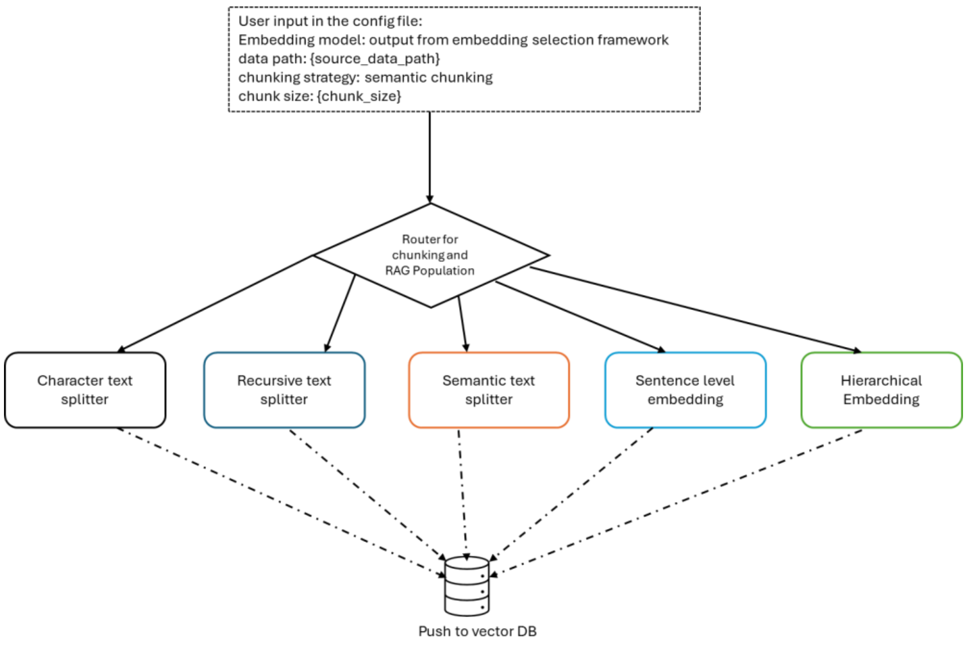

User input in the config file:
Embedding model: output from embedding selection framework
data path: {source_data_path}
chunking strategy: semantic chunking
chunk size: {chunk_size}
Router for chunking and RAG Population
Character text splitter
Recursive text splitter
Semantic text splitter
Sentence level embedding
Hierarchical Embedding
Push to vector DB

Figure 5. Agentic approach to populate Vector DB based on configuration file

- **Evaluation metrices for RAG based applications**: RAGAS [12] package is utilized towards performing the evaluation of the application in the beginning and in-between stages. It contains three metrices which include context relevance, groundedness and answer relevance where context relevance refers to the measurement of relevance of the context to the user query. To verify the groundedness of the application, we can separate the response into individual claims and independently search for evidence that supports each within the retrieved context. Answer relevance is the metric which checks the relevance of the answer to the user query. We can verify this by evaluating the relevance of the final response to the user input.

- **RAG implementation for Large Language Model and Small Language Model:** Implementing RAG techniques for small language model is little bit different from the LLM; hence initial retriever and poly encoder re-ranker are proposed here instead of all other advanced techniques.

- **Challenges in handling multiple documents**: This is one thing in real life applications where multiple input documents are there and the answer to each question comes from more than one document. In these cases, some advanced techniques for auto-merging retriever is proposed which takes care of this problem by returning the parent chunk instead on individual child chunk.

B. Embedding Model Selection Framework

Embedding selection framework goes through a series of six steps, below we will discuss each of the steps in detail.

Step 1 - Selection base of 100 embedding models: Using a popular leader board [10] and after doing some of our research on the popular embedding models we have selected a set of 100 embedding models. Our recommendation would be based on this set of models.

Step 2 – Identify the decision variables or implicit variables: The decision variables are the parameters or features used to describe various aspects of the embedding model. Out of several available parameters, we have considered the following ones primarily in our framework.

- **Model Size:** This refers to the number of parameters in the embedding models. Larger models usually have more parameters, which can lead to better performance but also higher computational costs and latency.
- **Embedding Dimension:** The size of the vector representation (embedding) that the model generates for each input token. Common sizes are 300, 768, 1024, etc. The latency of semantic search grows with the dimension of embeddings. Low dimensional embeddings can be selected to minimize latency.
- **Average:** This refers to the average performance metric across multiple tasks or datasets.
- **Classification Average:** The average performance of the model on classification tasks. This includes accuracy metric and unit is percentage (%).
- **Retrieval Average:** The average performance of the model on retrieval tasks, which involve finding relevant documents or passages based on a query. Metrics include Normalized Discounted

Cumulative Gain @ 10 (nDCG@10) and unit is ratio (range from 0 to 1).

- **Clustering Average:** The average performance of the model on clustering tasks. This involves Validity Measure(V-measure) and unit ratio (0 to 1).
- **STS (Semantic Textual Similarity):** Semantic Textual Similarity is the task of determining how similar two texts are. Metrics include Spearman correlation based on the model's similarity metric (usually cosine) and unit is ratio (range from -1 to 1).
- **Summary Average:** Summarization is the task of generating a summary of a text. Metrics include Spearman correlation based on the model's similarity metric (usually cosine) and unit is percentage (%).
- **Max Token:** The maximum number of tokens (words or subwords) the model can process in a single input. This is important for understanding the model's capacity to handle longer texts.
- **Embedding Latency:** The time it takes for the model to generate an embedding for a given input. This is usually measured in milliseconds and can be influenced by the model size, input length, and hardware used and unit is milliseconds (ms).
- **Embedding Type:** The type of embeddings the model generates. This includes static and dynamic. Static models require inputs of a fixed length, needing padding or truncation whereas dynamic models can handle varying input lengths.
- **GPU Support:** Indicates if the model can leverage GPU acceleration for faster computation. Models that support GPU can significantly reduce latency and improve throughput compared to CPU-only models.
- **Multilingualism:** Multilingual encoder or a translation system can be chosen alongside an English encoder to support non-English languages. Key considerations for choosing a multilingual embedding model include language coverage, dimensionality, and integration ease.
- **Model Type**: Stringent data privacy requirements, especially in sensitive domains like finance and healthcare, may influence the choice of embedding services. Evaluate privacy considerations before selecting a provider. Hence, we can select public models

instead of licensed ones as those can be downloaded privately and utilized.

Step 3 – Capture and Summarize User requirements: This is a very crucial step as user satisfaction ultimately determines the efficacy of the framework. Hence, user requirement should be analyzed carefully to ensure the right model selection. Seven common input parameters are primarily considered which are presented below. They cover most of the queries present in the problem statement section, also referred as explicit variables.

- Use Case
- Latency
- Document Length
- Task Type
- GPU Availability
- Model Type
- Non-English Language Support Required

Step 4 – Build reference table for embedding models consists of implicit variables: Based on the available features of embedding models the process builds the reference dataset which contains the values of implicit variables.

Step 5 – Map decision variables (or implicit variables) from user requirements (or explicit variables): Once we have captured the user requirements, the next step is to map explicit variables to the implicit variables. Following Figure 6 shows the mapping between the two set of variables. Based on that, the rulesets are created for recommendation.

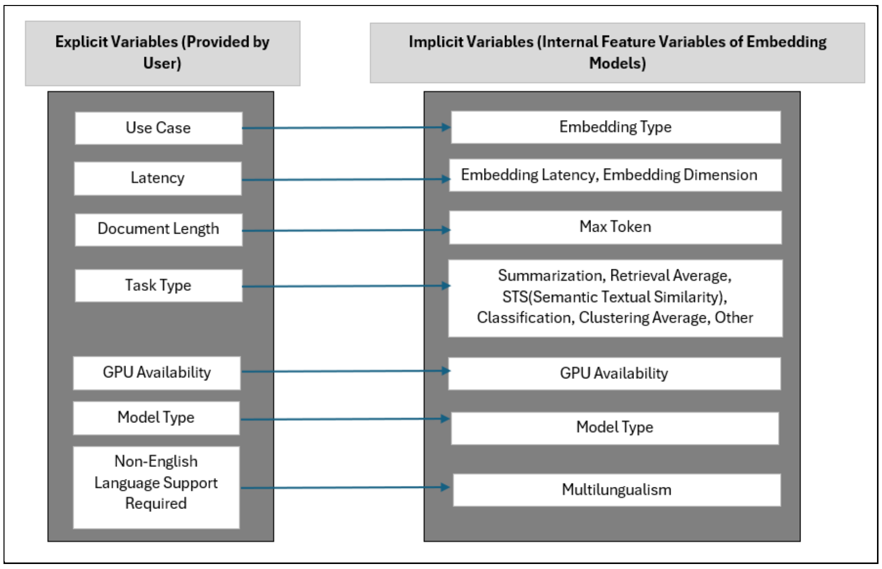

Explicit Variables (Provided by User)
Implicit Variables (Internal Feature Variables of Embedding Models)
Use Case
Latency
Document Length
Task Type
GPU Availability
Model Type
Non-English Language Support Required
Embedding Type
Embedding Latency, Embedding Dimension
Max Token
Summarization, Retrieval Average, STS(Semantic Textual Similarity), Classification, Clustering Average, Other
GPU Availability
Model Type
Multilungualism

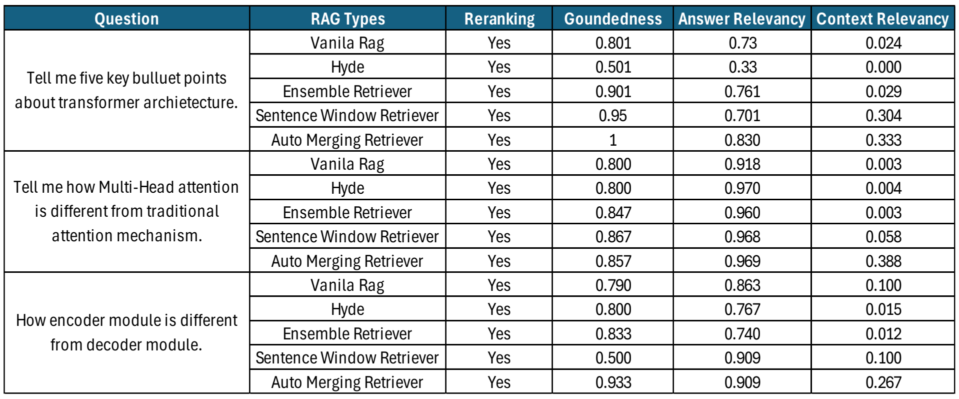
Figure 6. Mapping between implicit and explicit variables

Step 6 – Recommend appropriate embedding models: Based on the rules and mapping between the explicit and implicit variables; finally, filters are applied to come up with set of recommendations. This set of recommendations is also sorted based on the user provided requirements.

7 Experimental Setup and Results

A. Performance improvement of RAG based applications through proposed Framework

Our proposed framework is fully functional which was tested on multiple datasets. It allows us to run different RAG methodologies and provides evaluation scores accordingly. Appropriate recommendations can be chosen from the framework based on the received evaluation score and other user-provided requirements towards improving the same. Below Figure 7 and Figure 8 present a few such sample snapshots of the results.



Question	RAG Types	Reranking	Goundedness	Answer Relevancy	Context Relevancy
Tell me five key bulluet points about transformer archietecture.	Vanila Rag	Yes	0.801	0.73	0.024
	Hyde	Yes	0.501	0.33	0.000
	Ensemble Retriever	Yes	0.901	0.761	0.029
	Sentence Window Retriever	Yes	0.95	0.701	0.304
	Auto Merging Retriever	Yes	1	0.830	0.333
Tell me how Multi-Head attention is different from traditional attention mechanism.	Vanila Rag	Yes	0.800	0.918	0.003
	Hyde	Yes	0.800	0.970	0.004
	Ensemble Retriever	Yes	0.847	0.960	0.003
	Sentence Window Retriever	Yes	0.867	0.968	0.058
	Auto Merging Retriever	Yes	0.857	0.969	0.388
How encoder module is different from decoder module.	Vanila Rag	Yes	0.790	0.863	0.100
	Hyde	Yes	0.800	0.767	0.015
	Ensemble Retriever	Yes	0.833	0.740	0.012
	Sentence Window Retriever	Yes	0.500	0.909	0.100
	Auto Merging Retriever	Yes	0.933	0.909	0.267

Figure 7. RAG improvement framework evaluation results

Below is the comparison of different advanced RAG techniques using the recommended framework. Th result shows that average context relevancy has been improved using some of the advanced RAG recommendations like sentence window retriever or auto merging retriever.

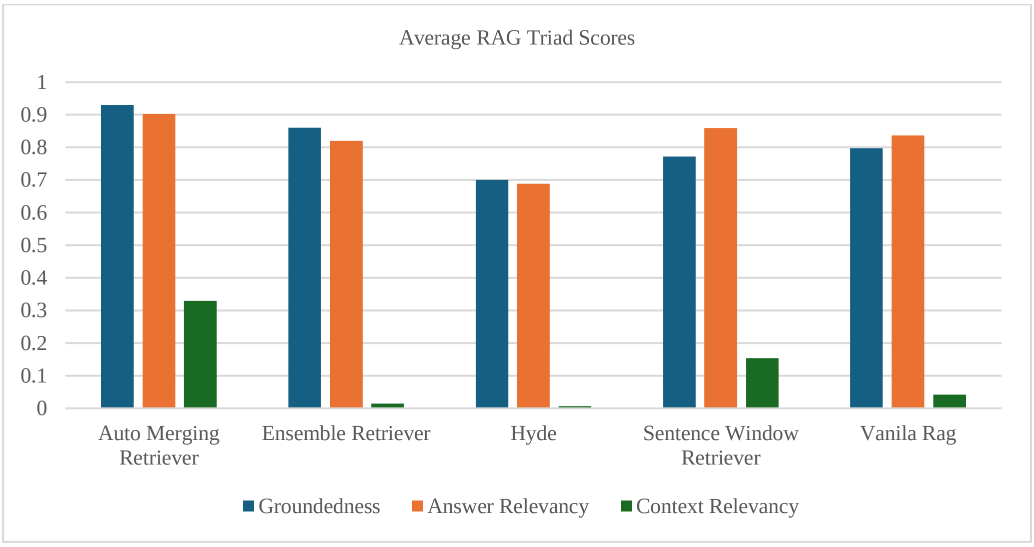
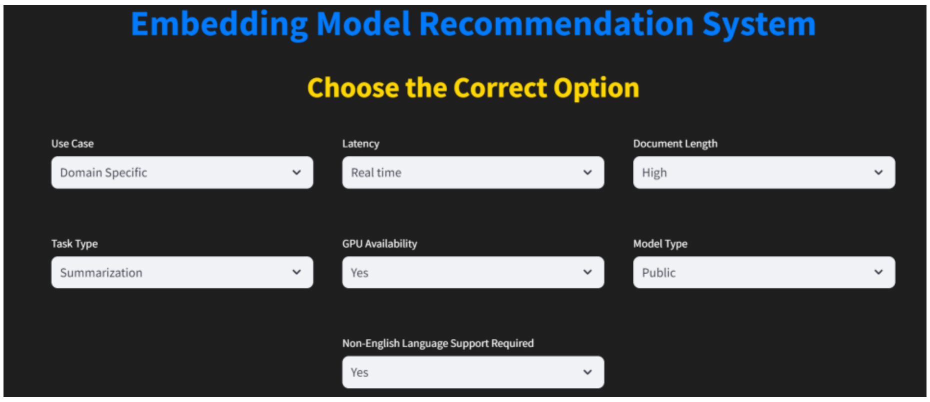


Figure 8. Average RAG Triad score for different RAG techniques

B. Embedding model selection framework

Embedding model selection framework is based on a user interface where users can select different criteria based on the use case which will in turn produce a list of recommended embedding models. Following Figure 9 shows the list of choices present in the user interface whereas Figure 10 presents the recommendations of the embedding models.

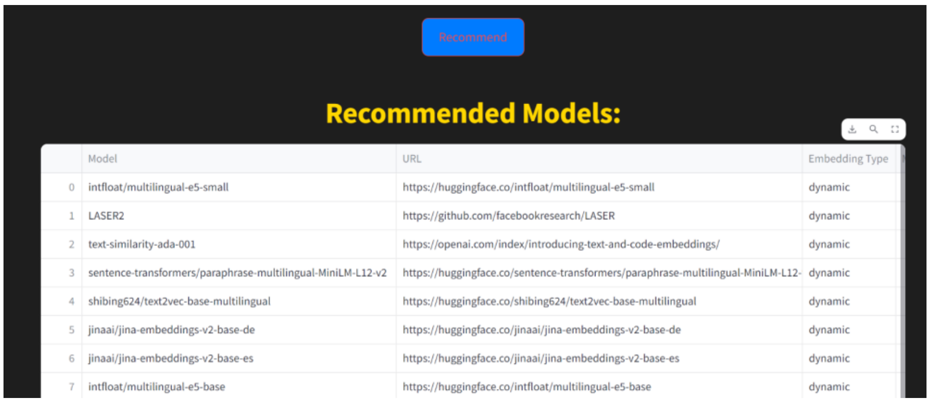


Embedding Model Recommendation System

Choose the Correct Option

Use Case: Domain Specific
Latency: Real time
Document Length: High
Task Type: Summarization
GPU Availability: Yes
Model Type: Public
Non-English Language Support Required: Yes

Figure 9. Embedding model selection framework – user selection criteria



Recommended Models:

	Model	URL	Embedding Type
0	intfloat/multilingual-e5-small	https://huggingface.co/intfloat/multilingual-e5-small	dynamic
1	LASER2	https://github.com/facebookresearch/LASER	dynamic
2	text-similarity-ada-001	https://openai.com/index/introducing-text-and-code-embeddings/	dynamic
3	sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2	https://huggingface.co/sentence-transformers/paraphrase-multilingual-MiniLM-L12-	dynamic
4	shibing624/text2vec-base-multilingual	https://huggingface.co/shibing624/text2vec-base-multilingual	dynamic
5	jinaai/jina-embeddings-v2-base-de	https://huggingface.co/jinaai/jina-embeddings-v2-base-de	dynamic
6	jinaai/jina-embeddings-v2-base-es	https://huggingface.co/jinaai/jina-embeddings-v2-base-es	dynamic
7	intfloat/multilingual-e5-base	https://huggingface.co/intfloat/multilingual-e5-base	dynamic

Figure 10. Embedding model selection framework – model recommendations

8 Conclusions and Future Scope

Currently, RAG is the most widely used approach in the current era of Gen AI; however, the response is not always optimal due to the various limitations in different phases of retrieval, augmentation and generation. Few such limitations are missing content, missing top ranked documents, limitation in consolidation strategy, problem in extraction format, incorrect specificity and incomplete response. An end-to-end RAG method adoption framework is proposed towards addressing this challenge which will recommend the optimal technique in each stage of evaluation based on existing scenarios so that the application performance can be improved. Additionally, one recommendation system is developed which will suggest the most appropriate embedding model based on user requirements as embedding model is the heart of the RAG architecture.

However, there are few areas exist to improve the framework which include developing new advanced approaches to receive better contextual response from the LLM, incorporating agents in the whole eco-system to make the RAG agentic, proposing autonomous agents for improving RAG based application performance, hallucination identification and mitigation with agents and self-corrective process with feedback loop which will select the optimal RAG improvement strategies in automated manner.

References

[1] Lewis, Patrick, et al. "Retrieval-augmented generation for knowledge-intensive nlp tasks." *Advances in Neural Information Processing Systems* 33 (2020): 9459-9474.
[2] Achiam, Josh, et al. "Gpt-4 technical report." *arXiv preprint arXiv:2303.08774* (2023).
[3] Touvron, Hugo, et al. "Llama: Open and efficient foundation language models." *arXiv preprint arXiv:2302.13971* (2023).
[4] Salemi, Alireza, and Hamed Zamani. "Evaluating Retrieval Quality in Retrieval-Augmented Generation." *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2024.
[5] Es, Shahul, et al. "Ragas: Automated evaluation of retrieval augmented generation." *arXiv preprint arXiv:2309.15217* (2023).
[6] Gao, Yunfan, et al. "Retrieval-augmented generation for large language models: A survey." *arXiv preprint arXiv:2312.10997 (2023).*
[7] Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. Query rewriting for retrieval-augmented large language models. *arXiv preprint arXiv:2305.14283, 2023.*
[8] Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. Precise zero-shot dense retrieval without relevance labels. *arXiv preprint arXiv:2212.10496, 2022.*

[9] Muennighoff, Niklas, et al. "MTEB: Massive text embedding benchmark." *arXiv preprint arXiv:2210.07316* (2022).
[10] Wang, Bin, et al. "Evaluating word embedding models: Methods and experimental results." *APSIPA transactions on signal and information processing* 8 (2019): e19.
[11] Excoffier, Jean-Baptiste, et al. "Generalist embedding models are better at short-context clinical semantic search than specialized embedding models." *arXiv preprint arXiv:2401.01943* (2024).
[12] RAGAS package reference: https://docs.ragas.io/en/stable/concepts/metrics/index.html
[13] RAGAS package reference: https://ragas.io/
[14] Barnett, Scott, et al. "Seven failure points when engineering a retrieval augmented generation system." *Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering-Software Engineering for AI*. 2024.
[15] Wang, Xiaohua, et al. "Searching for best practices in retrieval-augmented generation." *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. 2024.
[16] Jeong, Cheonsu. "A Study on the Implementation Method of an Agent-Based Advanced RAG System Using Graph." *arXiv preprint arXiv:2407.19994* (2024).
[17] Cuconasu, Florin, et al. "The power of noise: Redefining retrieval for rag systems." *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2024.