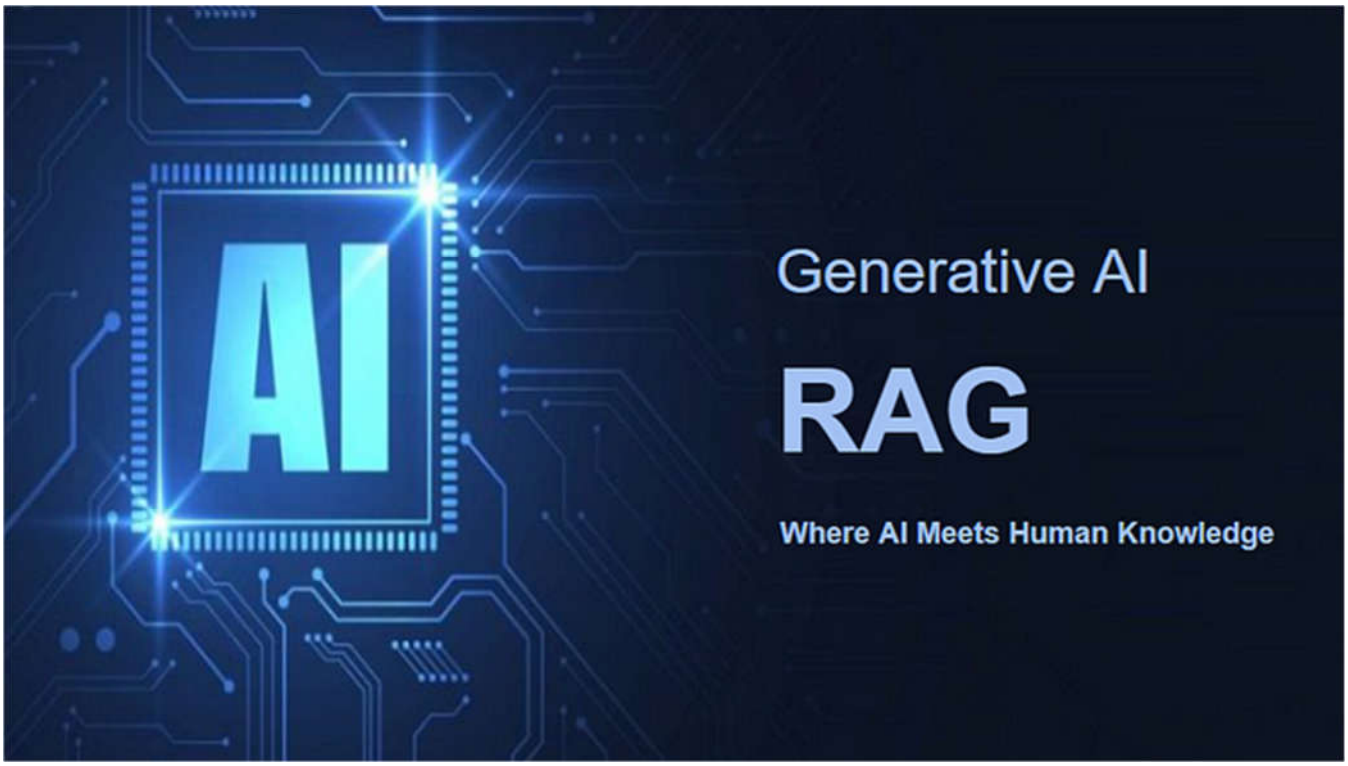
RAG Architecture

Uğur Öz Mar 23, 2024



What is RAG?

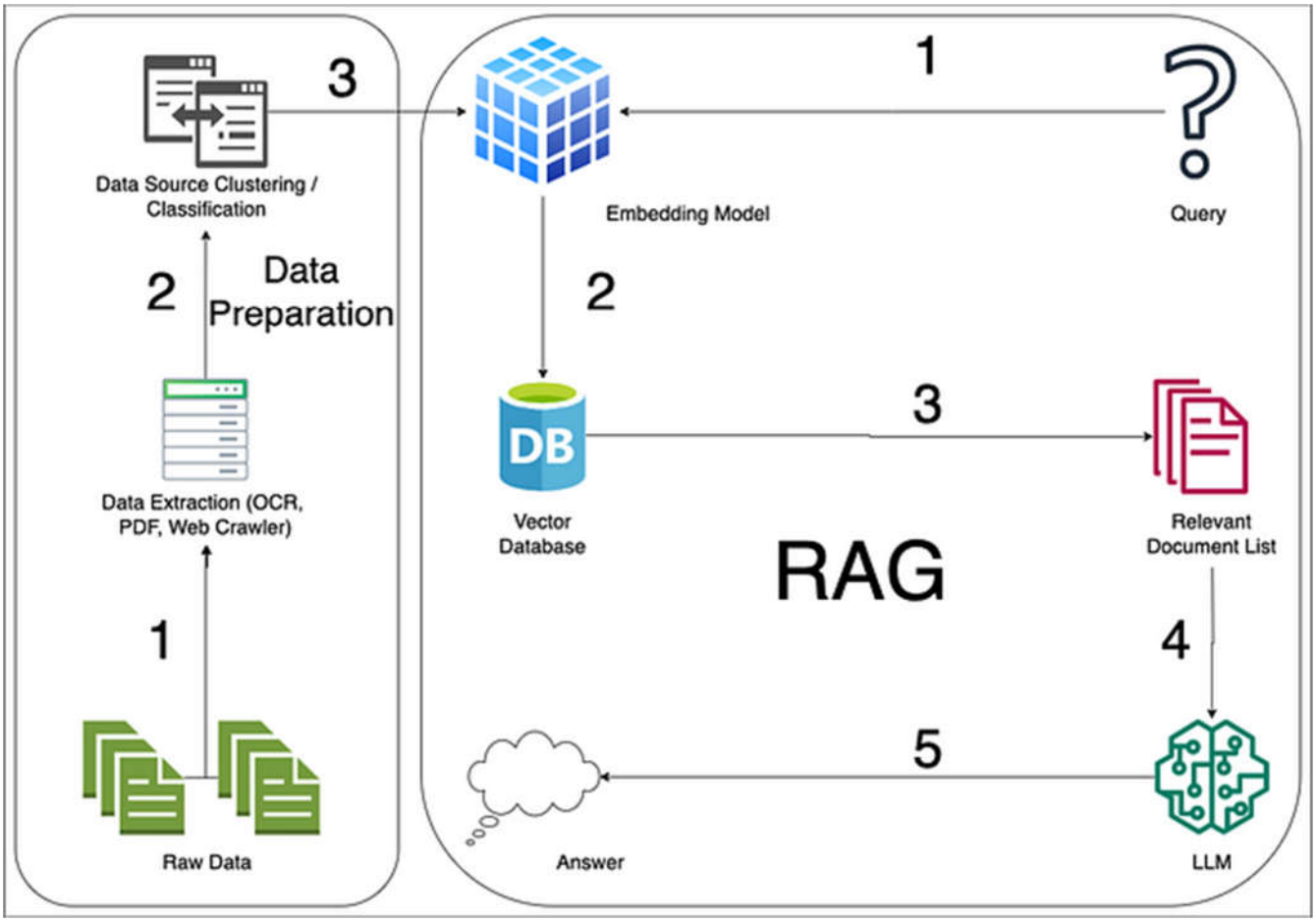
The concept known as ‘Retrieval Augmented Generation’, abbreviated as RAG, first entered our lives with an academic study published by Meta in 2020. Although the concept has a short history, it has revealed serious potential when combined with large language model technology and is now at the center of generative artificial intelligence, offering us opportunities as the largest commercial use case that can be benefited from in this field.

RAG augments the capability of large language models with masses of data outside the base model, enabling model responses to generate more real, more individual, and more reliable outputs. For this reason, for RAG; We can say that it is a framework that provides functionality to improve the performance of large language models. Thanks to this flexible and powerful framework, RAG has achieved significant growth in the applications of large language models in the corporate field in just 3 years. According to the Retool Report published in 2023, an impressive 36.2% of enterprise big language model use cases now use the RAG framework. RAG combines the power of large language models with structured and unstructured data, making enterprise information access more effective and faster

than ever before. In this way, more competent and successful artificial intelligence services can be produced without the need for data science processes such as data preparation required by traditional virtual assistants, while spending minimum labor.

How Does RAG Work?

A typical RAG process, as pictured below, has a large language model at its core, a collection of corporate documents to be used to feed the model, and prompt engineering infrastructure to improve response generation. RAG workflow; It uses the vector database to find concepts and documents similar to the question asked, and prompt engineering techniques to convert the relevant data into the format expected by the model. This process of RAG makes it a powerful tool for companies looking to leverage existing data repositories for advanced decision-making and information retrieval. To look at the application steps in order:



1-Query: The question in text format is sent to the RAG flow through any virtual assistant or interface.

2-(**R**etrieval) Document Search: The model performs a search step to collect relevant information from external sources. These sources may include a database,

a set of documents, or even search engine results. The search process aims to find text fragments or documents containing information relevant to the given input or request.

3-**A**ugmentation: The information obtained during the search phase is then combined with the original input or prompt and enriched by creating a prompt engineering draft that the model can use to create the output. The model is brought to the format expected by the large language model by including external information in this draft created through prompt engineering.

4-**G**eneration: Finally, the model produces the answer by taking into account the received information and the original input. Here, the first form of the question posed to the system, the document obtained from the vector database and other arguments are evaluated together to ensure that the large language model produces the most accurate output text.

5-Answering: New content created by the large language model is transferred to the user.

RAG can be useful in a variety of natural language processing tasks, such as question answering, conversation generation, summarization, and more. By incorporating external information, RAG models demonstrate the potential to provide more accurate and informative answers than traditional models that rely solely on the data on which they are trained.

RAG is a suitable solution for a wide range of industries and use cases. It facilitates informed decision-making processes by helping to reference information from large databases or document repositories in the finance, legal and healthcare sectors. Additionally, regardless of the sector, RAG in the field of customer service; It is used to power virtual assistants and provide accurate and contextually appropriate answers to user queries. Apart from this, it also has an important place in personalized content creation and recommendation systems by understanding user preferences and historical data.

Differences Between RAG and Traditional Methods

Traditional Models Based on Classification
Traditional natural language models are designed to select an appropriate response from a set of predefined responses based on the input query. These models compare input text (a question or query) against a taxonomy set of predefined answers. The system determines the most appropriate response by measuring the

similarity between the input and labeled responses using techniques such as supervised learning algorithms or other semantic matching methods. These classification-based models are effective for tasks such as question answering, where answers are often based on static response types and can be easily found in a structured form.

Next Generation Models Based on RAG-Based Rendering
Unlike traditional methods, productive artificial intelligence models create answers or content from scratch instead of matching existing content. These models use more complex algorithms, often based on neural networks, to generate human-like text or responses. Unlike the method we are used to, they do not need to associate it with any existing classes because they create new content. For this reason, it brings with it all the advantages and strengths of unsupervised learning. With these models, which are based on unsupervised learning, they learn to produce answers by predicting the next word or sequence of words based on the content provided by the query. This ability to generate new, contextually appropriate responses makes generative AI-based models extremely versatile and suitable for tasks such as creative writing, translation, and dialogue where responses must be rich in content.

	Traditional AI Models	**Generative AI Models**
Data Dependency	Relies on the Existence of Predefined Answers in the Dataset.	No Predefined Answers Required; They Produce Unique Responses Based on Learned Patterns.
Response Diversity	Limited to Answers Found in Data Source. Works with Repetitive and Stereotyped Limited Response Types.	Can Produce Diverse and Contextually Rich Responses, Leading to More Engaging and Creative Interactions.
Contextual Understanding	It Focuses on Matching the Introductory Sentence with Existing Responses, Lacking a Detailed Understanding of the Input Sentence.	It can capture subtle detail and improve the quality of interactions by producing responses tailored to specific inputs.
Training Complexity	It is generally easier to train because it involves matching input patterns to predefined responses.	A more complex training process often requires large data sets and complex neural network architectures.

One of the first things to consider when developing a RAG application for your organization is to accurately identify the types of questions that arise in the workflow and data for which you create the RAG framework and what processes will address them.

Advantages of Using RAG

RAG offers several advantages, especially in scenarios where access to external information and current data is useful. Some of the key advantages are:

1. Contextual Relevance: RAG models can produce dynamic responses that are more contextually relevant and informative. Text created by combining information from external sources is better grounded in current real-world information, resulting in more accurate and context-sensitive answers.
2. Fact Check and Verification: Since RAG models receive information from reliable external sources, they can perform fact check and verification during the production process. This helps reduce the creation of false or misleading information and ensures the accuracy of the content created.
3. Improved Knowledge Incorporation: RAG models can effectively use external knowledge bases or documents to improve their answers. This is especially useful in question-answering tasks, where the model can access relevant information from a wide range of sources to provide well-informed and accurate answers.
4. Flexibility and Adaptability: The ability to obtain information from various sources makes RAG models more flexible and adaptable. They can address a wide range of topics and tasks without requiring explicit fine tuning for each specific scenario, as long as the search mechanism is designed to retrieve relevant information.
5. Handling Out-of-Distribution Inputs: Traditional text generation models may struggle when faced with out-of-distribution or unusual inputs that are not present in the training data. RAG models, on the other hand, can leverage document stacks via vector databases to find relevant information even for unseen or less common inputs.
6. Controlled Content Production: RAG models can also be used for controlled content production. By guiding the document search process and specifying sources, developers can control the type and quality of information the model uses to generate a response.
7. Reduced Bias: The document search mechanism can help reduce bias in the content created. By using a variety of information sources, the model can provide a more balanced and unbiased response compared to traditional models that can be affected by biases present in the training data.

Although RAG provides significant advantages, it is important to be aware of potential challenges and considerations, such as handling complex information from different sources and balancing the accuracy and efficiency of document search results.

What Points Does RAG Differ From Other Productive Artificial Intelligence Titles?

The main difference is where and how company data is stored and used. When you fine-tune a model, you retrain a pre-existing large language model using your company data and change the model configuration to meet the needs of your use case. RAG, on the other hand, retrieves data from externally stored company documents and provides them to the large language model to guide response generation. Fine-tuning is a long and costly process and is not a good solution for working with frequently changing company documents.

Compute Intensity and Complexity

	Prompt Engineering	RAG	Fine Tuning	Pre-Train From Scratch
What Purpose Does It Serve?	It ensures the development of claims by guiding LLM to give the most accurate and successful results.	Large Language Model Gains the Ability to Produce Responses Specific to the Purpose of Use by Evaluating External Current and Institution-Specific Data	It is the maturation of the model by training the previously trained large language model in a way that can be expanded with new data sets.	Creating a New Large Language Model with Raw Data from Scratch.
How Does It Do?	Creating Text Based Templates; Query Contains Description of Similar Document and Output	Content Obtained from Dynamic Datasets and External Document Sources is Feeded into the Model	Data Prepared with Deep Learning Algorithms is Added to the Existing Model and the Neural Network is Expanded	Algorithms are run on GPU resources with high energy and resource consumption and a new model is obtained.
What Are Its Highlights?	Fast, Economical, No Training or Resources Required	Ability to Work with Dynamic and Current Data, Increased Accuracy, Field-Specific Working	Ability to Create Unique Models with Granular Control, High Expertise, Mature and High Parameter Set	Maximum Control Ability Prepared for Special Needs
What Does It Need?	-	Document Repository or Third Party External Data Source	Thousands of Customized Data or Document Examples	Large Datasets in the Tens or Hundreds of TB

RAG Implementation Challenges

Although RAG is a very powerful tool, it also brings with it some challenges in terms of implementation and governance;

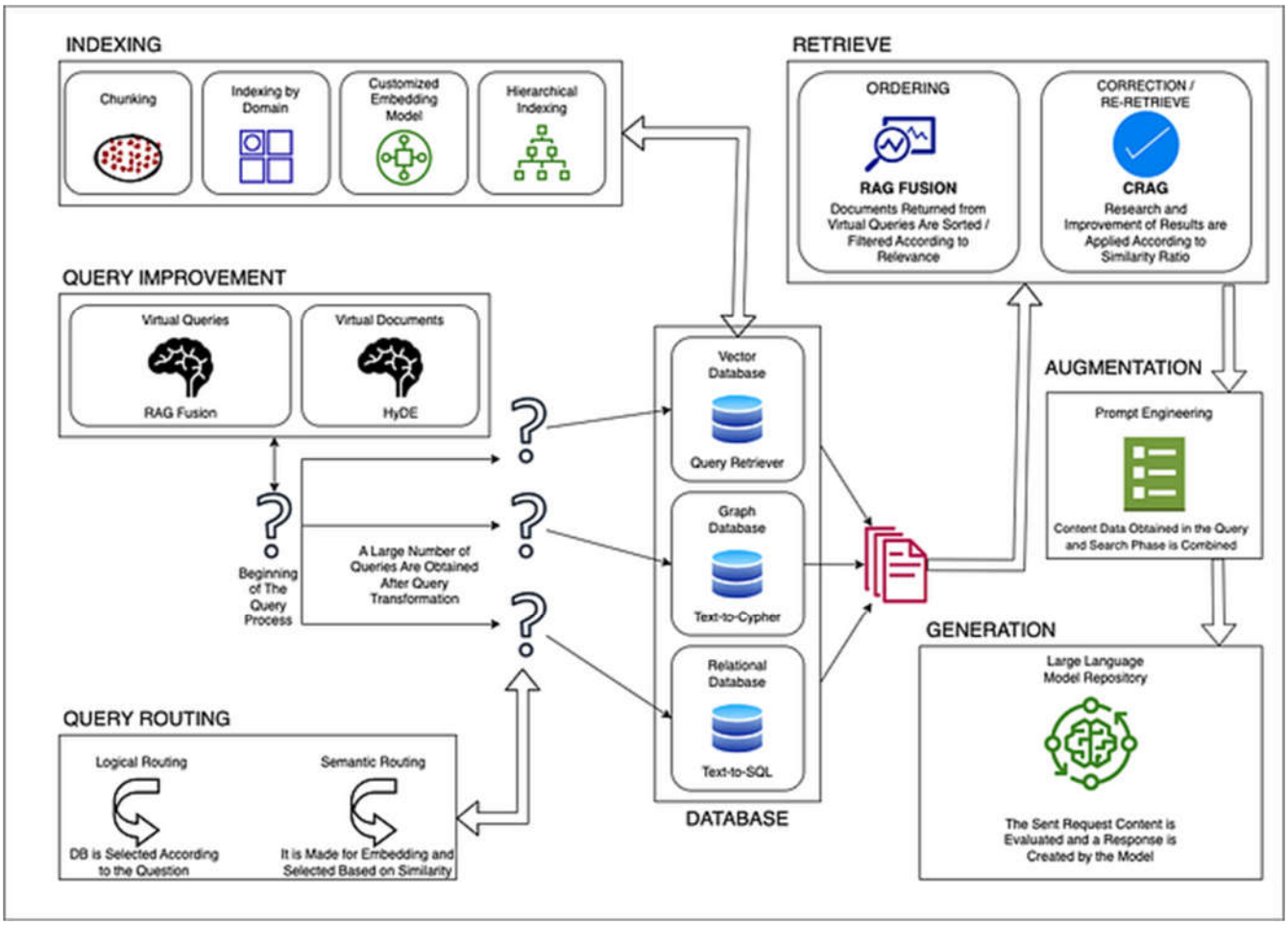
Multi-Channel/Source Integration: Complexity increases when there are multiple external data sources in different formats. To overcome this problem, it is important to pre-process or check these data to avoid duplication between datasets.

Data Quality: Data should be consistent and well representative. If the quality of the source content accessed by the application is poor, the answers produced will not be accurate. Therefore, it is necessary to improve data quality before integrating data sources.

Scalability: As the amount of data increases, it becomes difficult to manage the performance of the RAG system. To manage this problem more easily, solutions such as vector databases should be used.

Search Optimization: In order for the models to produce correct and desired outputs, the first step, the similarity search from the vector database, must work with high performance. Here, selecting missing or incorrect content as a result of the vector search will result in the failure to create a proper request engineering draft in the next step. The sketch is used as input to the large language model query, and an inefficient input will result in an equally inefficient output.

Complex and Advanced RAG Systems



Document Chunking

In the context of natural language processing, “chunking” means dividing text into small, concise, meaningful pieces. A RAG system can find relevant context more quickly and accurately in smaller pieces of text than in large documents. How can you make sure you choose the right part? The effectiveness of your fragmentation strategy depends largely on the quality and structure of these fragments. Determining the optimal chunk size is about striking a balance that means capturing all the important information without sacrificing speed. While larger chunks can capture more context, they introduce more noise and require more time and computational costs to process. Smaller parts have less noise but may not fully capture the necessary context. Overlapping parts is a way to balance both of these constraints. By overlaying parts, a query will likely retrieve enough related data over multiple sets of vectors to create a properly correlated answer. One limitation is that this strategy assumes that all the information you need to get can be found in a single document. If the required context is split across multiple different documents, it may be worth considering solutions such as document hierarchies and knowledge graphs. Here, we can think of the logic of extracting meaning piece by piece from

multiple documents as the same as bringing together information from different fields in relational databases to make it more valuable and meaningful.

CRAG

We can think of CRAG as verifying or improving search results, which is the first step involved in a RAG workflow. The general problem in all RAG processes is that the desired results cannot be obtained from the document stack during the search phase. Failure to obtain the most accurate documents needed leads to incomplete expression of the request and misdirection of the language model in the last step, which is the query to the large language model. A search problem experienced at the very beginning produces more clear and deviation-effective results in the following steps. In this case, it is critical to be able to check whether the search results actually yield the correct documents. The CRAG approach, developed as a solution to this problem, is a relatively new usage method. It is set up to classify the results as 'true', 'false' or 'unclear' results by defining an upper and lower threshold value for all search results. If the closeness ratio of at least one document to the query is above the threshold value, the search process is considered correct. If the score of all documents found is below the lower threshold value, it is assumed that the search process was incorrect. The uncertain state covers other situations that lie between the upper threshold value and the lower threshold value. For documents where the search result is assumed to be correct, another shredding is applied and the document is strengthened by cleaning the texts containing noise and enriching the content. In cases where it is incorrect, a web search is performed using the content and query, and more detailed information is obtained from online environments and correction work is carried out for misclassified documents. In cases of uncertainty, the processes applied for both right and wrong are repeated and carried out simultaneously.

Here, the similarity ratio determination for the documents obtained as a result of the search; It is provided with smaller language models compared to the 'Creation' step (literature studies use the t5-large model provided by Google with 770 million parameters). This model is called 'search evaluator'. In order for the search evaluator to evaluate the content correctly, it is important to include the relevant data in the fine-tuning process and turn it into a mature model for the efficiency of the CRAG process. On the other hand, the potential bias introduced by web searches for negative results is also concerning. The quality of Internet sources can vary significantly, and the inclusion of such data without sufficient consideration can introduce noise or misleading information in the outputs produced. Additionally, CRAG can add additional workload and cumbersomeness when not well optimized.

Assuming that a large number of documents will be selected for a search result and that each one will be queried separately with this small model, this may pose extra risks in terms of resource consumption and delay. For institutions using in-house systems, having to go online to improve incorrectly evaluated results may present another challenge.

Despite all the reservations and difficulties, the advantages and use of CRAG offer very strong benefits and can be applied in different fields; By assessing the quality of documents received, CRAG ensures that only relevant and reliable information is used and minimizes the risk of errors or misleading results, resulting in more accurate and reliable answers. CRAG facilitates communication between humans and machines by finding applications in automated content creation, question-answering systems, translation services, and educational tools, enabling more understandable empowered communication.

RAG Fusion

In traditional search systems, users typically enter a single query to find information. Although this approach is simple, it has limitations. A single query may not cover the full scope of the user's topic of interest or may be too narrow to yield comprehensive results. This is where creating multiple queries from different perspectives comes into play. RAG Fusion overcomes the inherent limitations of RAG by creating multiple user queries and reordering the results. While the method of duplicating queries from different perspectives is used in the pre-search process, it overcomes the inherent limitations of RAG by combining and reordering the results in the post-search processes. RAG-Fusion aims to bridge the gap between what users explicitly ask and what they want to ask, moving closer to uncovering transformative knowledge that often remains hidden. In this way, it provides the best understanding of the problem by diversifying it with a large number of questions. To put it more simply, think of RAG Fusion as the person who insists on getting everyone's opinion before making a decision. But in this case it is not annoying, but useful. The more there are, the more joyful, or in this case, the more correct decisions will be made.

RAG Fusion leverages reciprocal sequence fusion technology to provide this powerful support. Reciprocal Ranking Fusion (RRF) is a technique for combining the rankings of multiple search result lists to create a single combined ranking. RRF, developed in collaboration with the University of Waterloo and Google, "provides better results than any individual system and better results than standard reordering methods," as its authors put it. When you use RAG Fusion, the depth of your search is not only enhanced but also strengthened. The reordered list of relevant documents means that you are not only getting superficial information but also

diving into an ocean of perspectives. Structured output is easier to read and feels intuitively reliable; This is crucial in a world that is skeptical of AI-generated content. Additionally, the System not only interprets but also refines user queries. RAG Fusion improves search result accuracy by performing implicit spelling and grammar checks through the creation of multiple query variations.

In summary; Think about the information you didn't know you needed until you came across it. RAG Fusion enables this serendipitous discovery. By using a wider range of queries, the system opens up the possibility of uncovering information that, although not explicitly sought, becomes a pivotal moment for the user. This distinguishes RAG Fusion from other traditional search models.
The sensitive point that needs to be taken into consideration is; The depth of RAG-Fusion can sometimes lead to a flood of information. The outputs can be so detailed as to be overwhelming. Think of RAG-Fusion as an informative companion that over-explains things, potentially distracting from the focus of the user's original intent. To mitigate this, it can be kept in check by instructing the model to give more weight to the original query in request engineering.

HyDE

Sometimes, when you are faced with a question that lacks detail or easily identifiable elements to derive an answer from a specific context, it can be quite challenging. For example, you want to receive calls from a contact center that includes many brands; When you ask 'the price of a phone that can take good photos' with a very short, simple question devoid of subject, the system should be able to understand that the product in question is a phone that takes good photos, even if the word phone is not used. The difficulty here is that the main item to be purchased is indirectly implied rather than clearly expressed in words. Therefore, searching for documents becomes problematic. To solve this problem, a temporary and virtual answer or document is created using the help of different language models, and this auxiliary document is later used to clarify the statement in order to obtain accurate results. These virtual contents are then examined in a vector resource based on semantic similarity, helping to search for relevant information. HyDE, in its simplest form, provides us with a new and flexible method defined as answer-to-answer instead of the classical query-to-answer method.

However, this approach has a disadvantage; because it may not always give good results. For example, if the topic under discussion is completely foreign to the language model, this method is not effective and may lead to increased cases of misinformation. Therefore, the created virtual content must be consistent and supportive of the original query.

Hierarchical Indexing

Document hierarchy is a powerful way to organize your data to improve information access. You can think of the document hierarchy as the table of contents of your RAG system. It organizes parts in a structured way that allows RAG systems to retrieve and process relevant data efficiently. Document hierarchies play a crucial role in the effectiveness of RAG by helping it decide which parts of large language models contain the most relevant data to extract.

Hierarchical indexing associates chunks with nodes and associates nodes as parent — child. Each node contains a summary of the information contained within it; This makes it easier for the RAG system to quickly flip through data and figure out which pieces to remove.

Once the hierarchical indexing method is understood so far, a question may come to mind; Why do we need a document hierarchy if large language models are assumed to be able to understand the content in a document? Think of the document hierarchy as a table of contents or file directory. Although a large language model can extract relevant text fragments from a vector database, you can increase the speed and reliability of the search process by using the document hierarchy as a preprocessing step to find the most relevant text fragments. This strategy may help reduce hallucinations caused by fragment extraction problems by increasing search reliability and speed.

Let’s consider the issue with the example of an international institution. Let’s say a company has many offices in different parts of the world, and each office has regulation and management processes specific to its region, but uses the same standard template to document these policies. As a result, each office’s documents are in roughly the same format, but each section will detail country-specific policies for rules, governance, regulations. In a vector database, each management or corporate culture-related paragraph block will look very similar to each other. In this case, a vector query could retrieve much of the same, useless data, leading to hallucinations. Using a document hierarchy, a RAG system can more reliably answer a question about regulations for a US office by first searching for documents relevant to the US office.

Written by Uğur Özker

Computer Engineer, MSc, MBA, PMP®, Senior Solution Architect IBM


LLM
LARGE LANGUAGE MODEL