Retrieval-Augmented Generation

A Seminar Report by

SINGH SHUBHAMKUMAR NANDAN

(PRN : 8023010736)

MSc Information Technology

Semester-II

Under the guidance of

Mrs. Radha Teredesai

Department of Computer Applications, Faculty of Science,
The Maharaja Sayajirao University of Baroda, Vadodara, Gujarat - 390002

ABSTRACT

RAG is an AI framework for retrieving facts from an external knowledge base to ground large language models (LLMs) on the most accurate, up-to-date information and to give users insight into LLMs' generative process.

Large language models can be inconsistent. Sometimes they nail the answer to questions, other times they regurgitate random facts from their training data. If they occasionally sound like they have no idea what they're saying, it's because they don't. LLMs know how words relate statistically, but not what they mean.

Retrieval-augmented generation (RAG) is an AI framework for improving the quality of LLM-generated responses by grounding the model on external sources of knowledge to supplement the LLM's internal representation of information.

RAG has additional benefits. By grounding an LLM on a set of external, verifiable facts, the model has fewer opportunities to pull information baked into its parameters. This reduces the chances that an LLM will leak sensitive data, or 'hallucinate' incorrect or misleading information.

RAG also reduces the need for users to continuously train the model on new data and update its parameters as circumstances evolve. In this way, RAG can lower the computational and financial costs of running LLM-powered chatbots in an enterprise setting.

CERTIFICATE

This is to certify that the work contained in this seminar report entitled **"Retrieva Augmented Generation "** submitted by **SINGH SHUBHAMKUMAR NANDAN**
(PRN : 8023010736) to the Department of Computer Applications, Faculty of Science, The Maharaja Sayajirao University of Baroda towards the partial requirement of Master of Science in Information Technology has been carried out by him under my supervision and that it has not been submitted elsewhere for the award of any degree.

Radha Teredesi
Seminar Guide

Department of Computer
Applications, Faculty of Science,
The Maharaja Sayajirao University of Baroda

Acknowledgement

Many people have contributed to the success of this. Although a single sentence hardly suffices, I would like to thank Almighty God for blessing us with His grace. I extend my sincere and heart felt thanks to Prof.
Prashant K. Mehta, Offg. Head, Department of Computer Applications , The Maharaja Sayajirao University of Baroda, for providing us the right ambience for carrying out this work.

I am profoundly indebted to my seminar guide, Mrs. Radha Teredesai for innumerable acts of timely advice, encouragement and I sincerely express my gratitude to her.

I express my immense pleasure and thankfulness to all the teachers and staff of the Department of Computer Applications The Maharaja Sayajirao University of Baroda for their cooperation and support.

Last but not the least, I thank all others, and especially my classmates who in one way or another helped me in the successful completion of this work.

Singh Shubhamkumar Nandan

8023010736

Index

INTRODUCTION

In recent years, the field of **Natural language processing (NLP)** has witnessed remarkable advancements, propelled by the emergence of innovative techniques and models. Among these, **Retrieval-Augmented Generation (RAG)** stands out as a paradigm-shifting approach that combines the strengths of **retrieval-based** methods with **generative models**, unlocking new frontiers in language understanding and generation.

At its core, **RAG** represents a **fusion of two** fundamental pillars of NLP: **retrieval and generation**. Traditional retrieval-based systems excel at accessing relevant information from large corpora but often struggle with generating fluent and contextually relevant responses. Conversely, **generative models**, such as **transformer-based architectures**, demonstrate remarkable proficiency in **generating human-like** text but may **lack the ability to incorporate external knowledge effectively.**

RAG bridges this gap by **integrating** retrieval mechanisms into generative models, thereby empowering them with the capacity to access and utilize external knowledge during the generation process. By leveraging pre-existing knowledge sources, such as large-scale text corpora or structured databases, RAG enhances the coherence, relevance, and factual accuracy of generated outputs.

The foundation of RAG lies in the synergy between retrieval and generation: while **retrieval mechanisms** provide **access to a vast reservoir of contextual information**, **generative**

models employ this information to produce nuanced and contextually appropriate responses. This symbiotic relationship enables RAG systems to exhibit a nuanced understanding of diverse topics and domains, surpassing the capabilities of traditional generative models.

The **applications of RAG** span a broad spectrum of domains, ranging **from question answering** and **dialogue systems** to content creation and **personalized recommendation engines**. Whether assisting users in retrieving relevant information from vast knowledge repositories or generating engaging and informative responses, RAG systems have demonstrated their utility across various real-world scenarios.

However, despite its transformative potential, RAG also poses **several challenges** and **considerations,** including **data quality and biases**, **computational complexity**, and **ethical implications**. Addressing these challenges necessitates ongoing research and development efforts to ensure the responsible and ethical deployment of RAG technologies.

In this report, we delve into the intricate workings of Retrieval-Augmented Generation, exploring its theoretical foundations, practical applications, implementation strategies, challenges, and future directions. Through a comprehensive examination of RAG, we aim to provide insights into this cutting-edge approach and its implications for the future of natural language processing and artificial intelligence.

Introduction to Large Language Models (LLMs):.

A large language model (LLM) is a type of artificial intelligence (AI) algorithm that uses deep learning techniques and massively large data sets to understand, summarize, generate and predict new content. The term generative AI also is closely connected with LLMs, which are, in fact, a type of generative AI that has been specifically architected to help generate text-based content.

Over millennia, humans developed spoken languages to communicate. Language is at the core of all forms of human and technological communications; it provides the words, semantics and grammar needed to convey ideas and concepts. In the AI world, a language model serves a similar purpose, providing a basis to communicate and generate new concepts.

The first AI language models trace their roots to the earliest days of AI. The Eliza language model debuted in 1966 at MIT and is one of the earliest examples of an AI language model. All language models are first trained on a set of data, and then they make use of various techniques to infer relationships and then generate new content based on the trained data. Language models are commonly used in natural language processing (NLP) applications where a user inputs a query in natural language to generate a result.

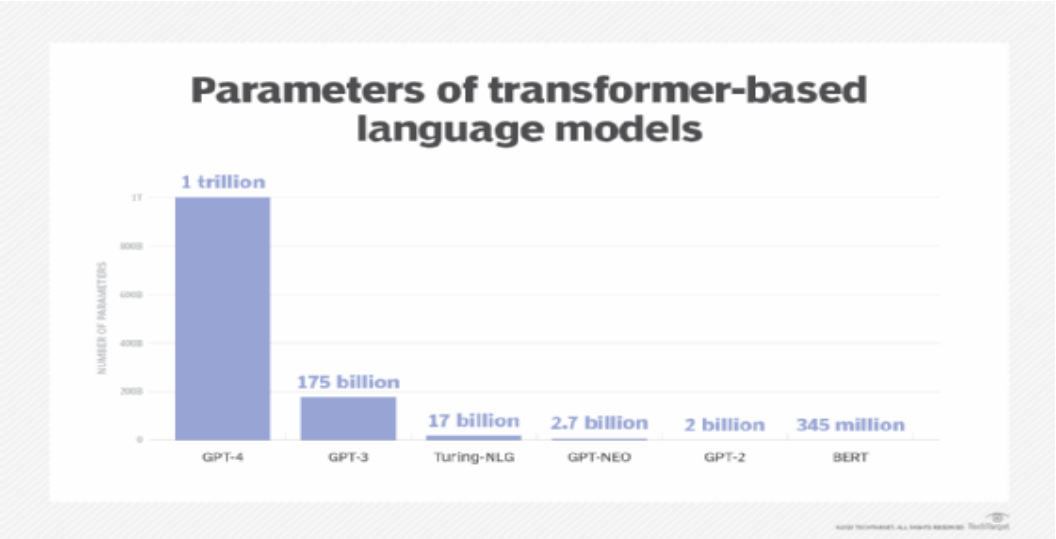
An LLM is the evolution of the language model concept in AI that dramatically expands the data used for training and inference. In turn, it provides a massive increase in the capabilities of the AI model. While there isn't a universally accepted figure for how large the data set for training needs to be, an LLM typically

has at least one billion or more parameters. Parameters are a machine learning term for

the variables present in the model on which it was trained that can be used to infer new content.

Modern LLMs emerged in 2017 and use transformer models, which are neural networks commonly referred to as transformers. With a large number of parameters and the transformer model, LLMs are able to understand and generate accurate responses

rapidly, which makes the AI technology broadly applicable across many different domains. Some LLMs are referred to as foundation models, a term coined by the Stanford Institute for Human-Centered Artificial Intelligence in 2021. A foundation model is so large and impactful that it serves as the foundation for further optimizations and specific use cases.



How Large do language models work?

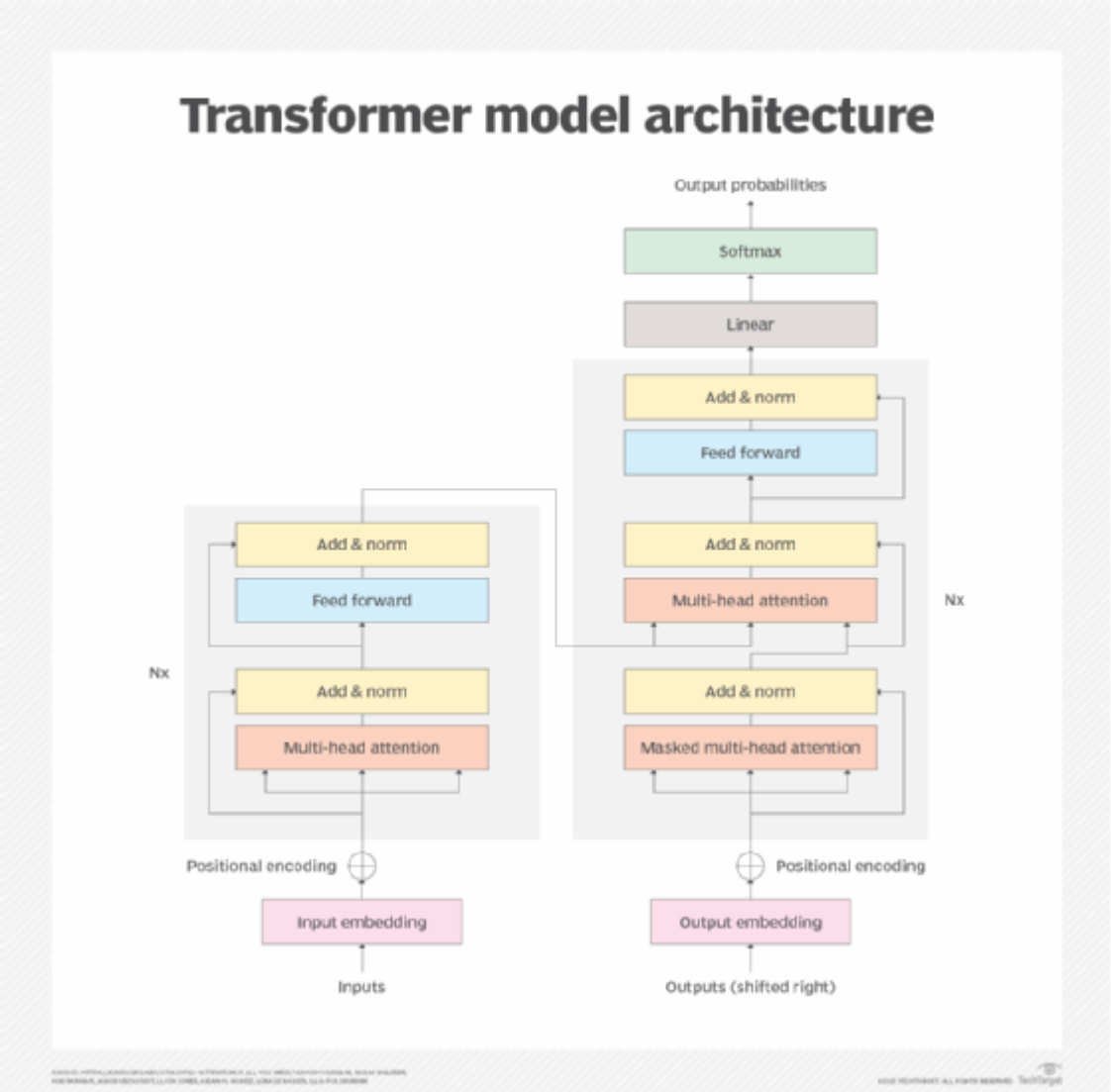
LLMs take a complex approach that involves multiple components.

At the foundational layer, an LLM needs to be trained on a large volume -- sometimes referred to as a corpus -- of data that is typically petabytes in size. The training can take multiple steps, usually starting with an unsupervised learning approach. In that approach, the model is trained on unstructured data and unlabeled data. The benefit of training on unlabeled data is that there is often vastly more data available. At this stage, the model begins to derive relationships between different words and concepts.

The next step for some LLMs is training and fine-tuning with a form of self-supervised learning. Here, some data labeling has occurred, assisting the model to more accurately identify different concepts.

Next, the LLM undertakes deep learning as it goes through the transformer neural network process. The transformer model architecture enables the LLM to understand and recognize the relationships and connections between words and concepts using a self-attention mechanism. That mechanism is able to assign a score, commonly referred to as a weight, to a given item (called a token) in order to determine the relationship.

Once an LLM has been trained, a base exists on which the AI can be used for practical purposes. By querying the LLM with a prompt, the AI model inference can generate a response, which could be an answer to a question, newly generated text, summarized text or a sentiment analysis report.


Transformer model architecture
Output probabilities
Softmax
Linear
Add & norm
Feed forward
Add & norm
Multi-head attention
Add & norm
Masked multi-head attention
Nx
Add & norm
Feed forward
Add & norm
Multi-head attention
Nx
Positional encoding
Positional encoding
Input embedding
Output embedding
Inputs
Outputs (shifted right)

What are large language models used for?

- **Text generation**. The ability to generate text on any topic that the LLM has been trained on is a primary use case.

- **Translation**. For LLMs trained on multiple languages, the ability to translate from one language to another is a common feature.

- **Content summary**. Summarizing blocks or multiple pages of text is a useful function of LLMs.

- **Rewriting content.** Rewriting a section of text is another capability.

- **Classification and categorization**. An LLM is able to classify and categorize content.

- **Sentiment analysis**. Most LLMs can be used for sentiment analysis to help users to better understand the intent of a piece of content or a particular response.

- **Conversational AI and chatbots.** LLMs can enable a conversation with a user in a way that is typically more natural than older generations of AI technologies.

What are the challenges and limitations of large language models?

- **Development costs.** To run, LLMs generally require large quantities of expensive graphics processing unit hardware and massive data sets.
- **Operational costs.** After the training and development period, the cost of operating an LLM for the host organization can be very high.
- **Bias.** A risk with any AI trained on unlabeled data is bias, as it's not always clear that known bias has been removed.
- **Explainability**. The ability to explain how an LLM was able to generate a specific result is not easy or obvious for users.
- **Hallucination.** AI hallucination occurs when an LLM provides an inaccurate response that is not based on trained data.
- **Complexity.** With billions of parameters, modern LLMs are exceptionally complicated technologies that can be particularly complex to troubleshoot.
- **Glitch tokens.** Maliciously designed prompts that cause an LLM to malfunction, known as glitch tokens, are part of an emerging trend since 2022.

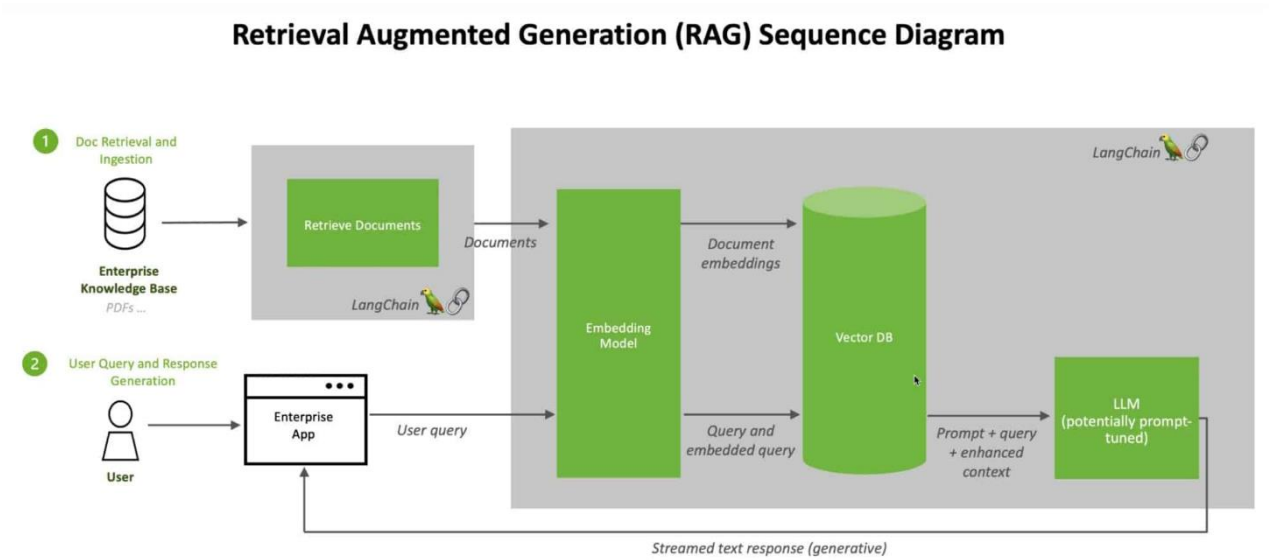
How RAG solves the limitation of LLm

1. **Mitigating Hallucinations:**
    - Hallucinations refer to instances where LLMs generate text that is factually incorrect or irrelevant to the context. RAG addresses this by incorporating retrieval mechanisms that fetch relevant context from external knowledge sources.
    - By retrieving contextually relevant information, RAG can ensure that the generated text is grounded in factual accuracy and context, reducing the likelihood of hallucinations.

2. **Access to Real-time Data:**
    - LLMs typically lack access to real-time data, which can lead to outdated or inaccurate responses. RAG overcomes this limitation by integrating retrieval mechanisms that fetch up-to-date information from dynamic data sources.
    - Through retrieval, RAG can access the latest information available, ensuring that generated text remains relevant and accurate in real-time scenarios.

3. **Enhancing Explainability:**
    - Explainability is a crucial aspect of natural language processing systems, allowing users to understand the reasoning behind generated outputs. LLMs often lack transparency in their decision-making processes.
    - RAG improves explainability by incorporating retrieval mechanisms that provide supporting evidence or context for generated text. Users can trace back retrieved sources to understand how the model arrived at its output, enhancing transparency and interpretability

Working of Retrieval-Augmented Generation (RAG)



- **Enterprise Knowledge Base:** This is a large collection of text documents that the RAG system can access to find relevant information .

- **Doc Retrieval and Ingestion:** This is the first step in the RAG process. The system retrieves documents from the knowledge base that are likely to be relevant to the user's query .

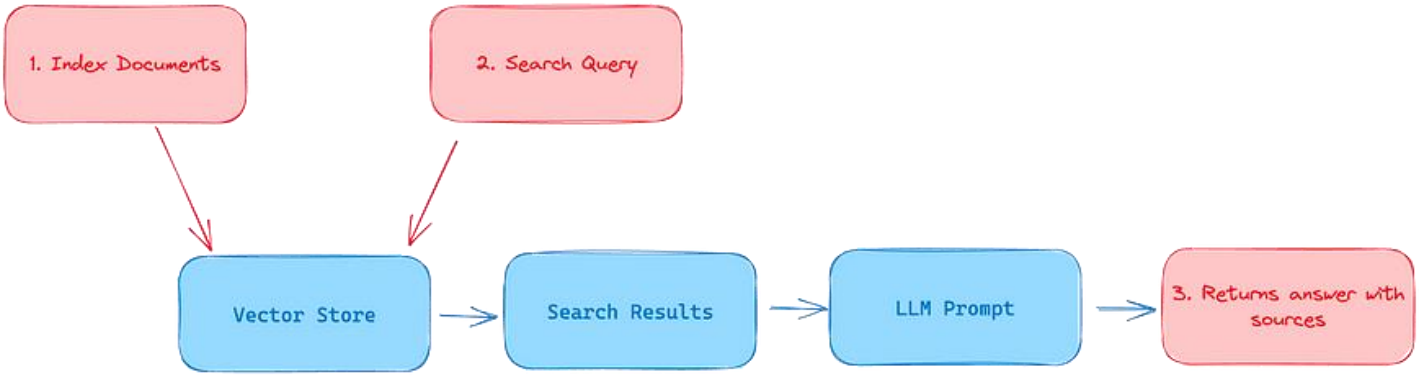
- **Document Embeddings:** The retrieved documents are then converted into a format that the RAG system can understand. This format is called a document embedding .

- **User Query and Response Generation:** The user enters a query, and the RAG system retrieves documents from the knowledge base that are relevant to the query. The system then uses the document embeddings to inform the generation of a response to the user's query .

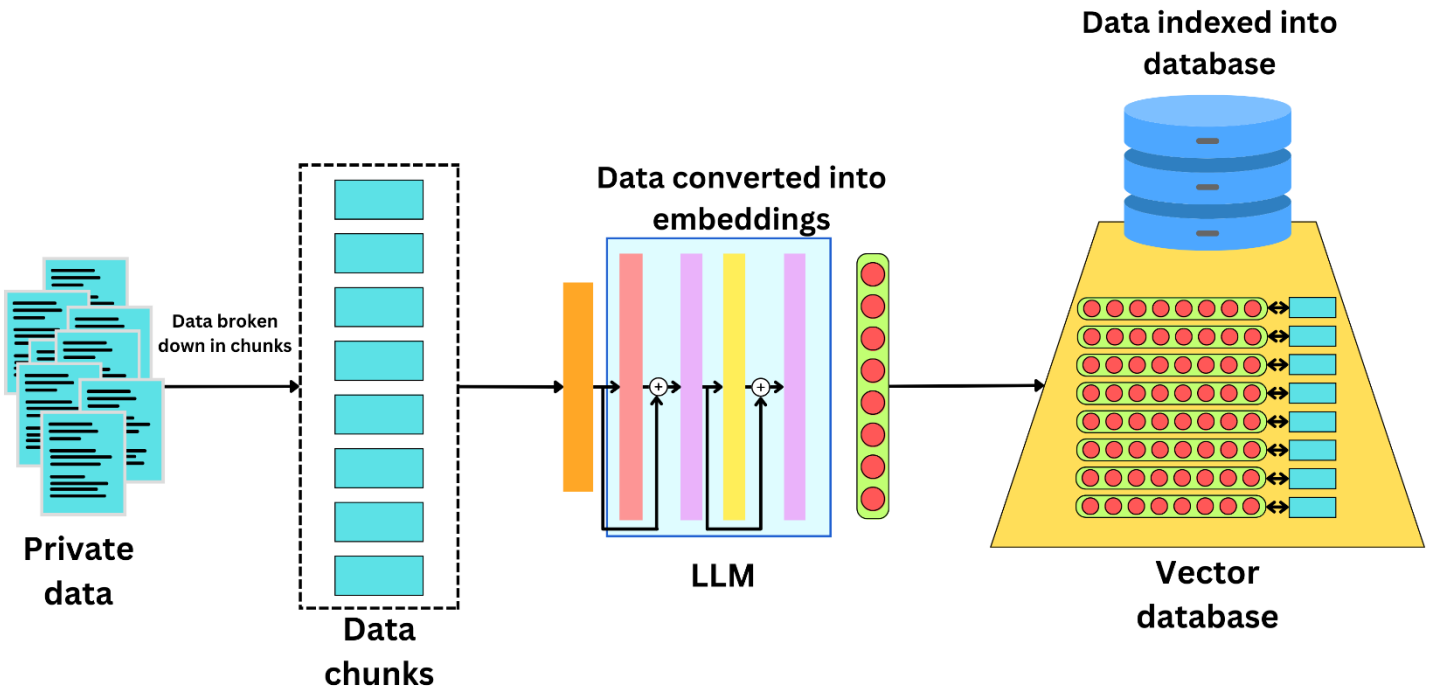
There are two main components to a RAG system:

- **Retriever:** This component is responsible for finding the documents in the knowledge base that are most relevant to the user's query .

- **Generator:** This component is responsible for generating a response to the user's query, using the information retrieved by the retriever [1].



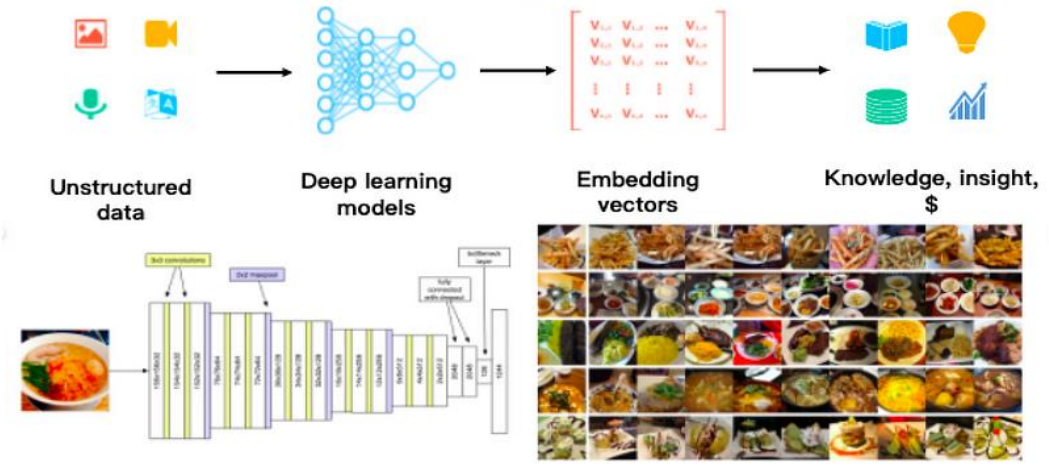
How Data are prepared for Retrieval-Augmented Generation (RAG)



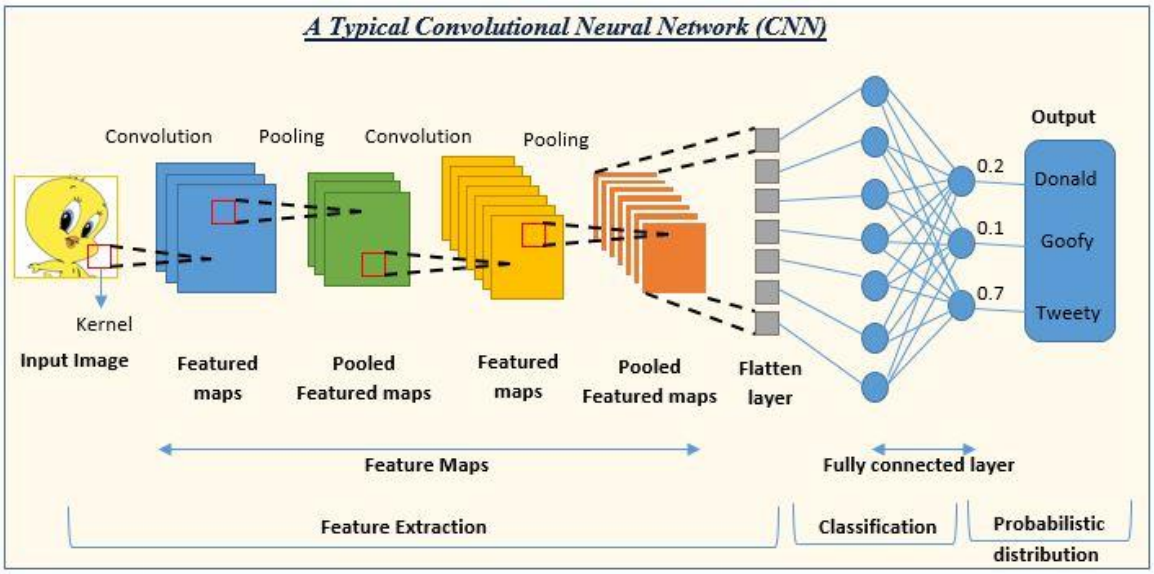
- The documents or PDF are first chunked or partitioned based on paragraphs or some delimiter.
- Then, these chunks or partitions are passed to the sentence transformer for obtaining embeddings.
- After obtaining embeddings from the sentence transformer, they are saved to a vector database for storage and retrieval.

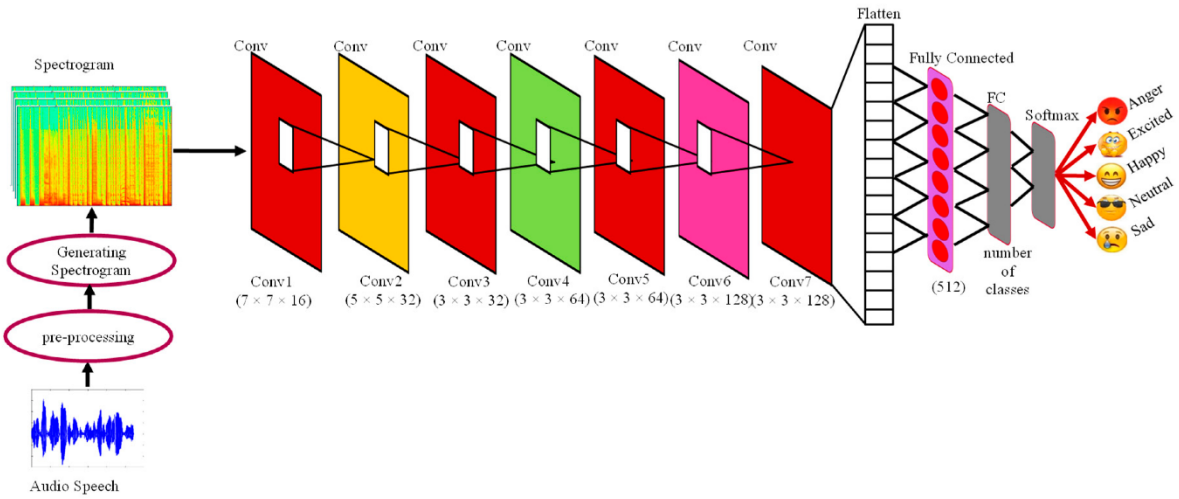
Working with different types of Data

In the context of Retrieval-Augmented Generation (RAG), the framework can be extended to

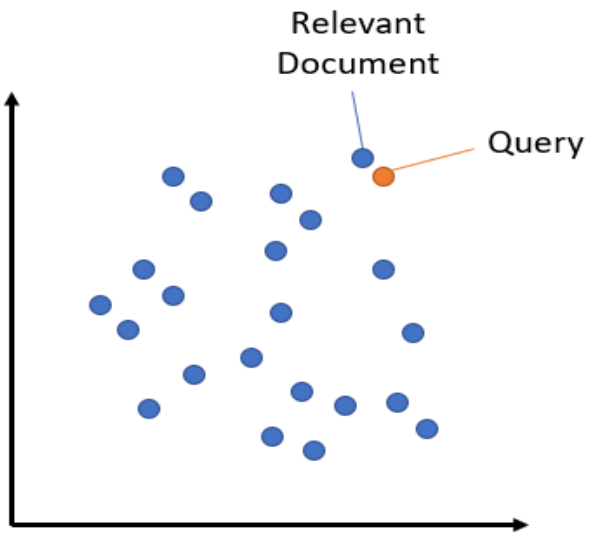


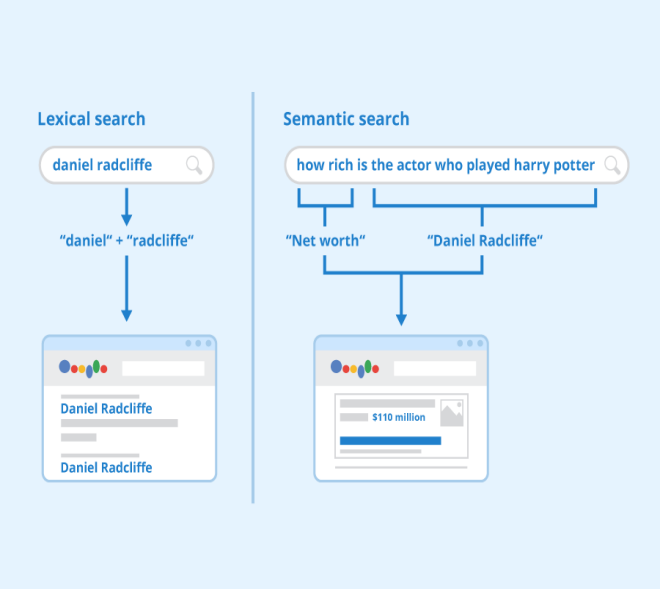
work with not only text but also other modalities such as images and audio. To integrate these modalities into the RAG model, their embeddings can be generated by passing them through Convolutional Neural Networks (CNNs) and extracting the values of the fully connected layers as their embeddings. This approach leverages the capabilities of CNNs in extracting meaningful features from images and audio, which can then be used as additional context for the generation process. By incorporating visual and auditory information into the RAG framework, the model gains access to a richer and more diverse set of inputs, enabling it to generate more contextually relevant and multimodal outputs. This extension opens up new possibilities for applications such as image captioning, multimodal dialogue systems, and content generation, where combining textual, visual, and auditory information can enhance the quality and richness of generated content.


A Typical Convolutional Neural Network (CNN)
Convolution
Pooling
Convolution
Pooling
Output
0.2
0.1
0.7
Donald
Goofy
Tweety
Kernel
Input Image
Featured maps
Pooled Featured maps
Featured maps
Pooled Featured maps
Flatten layer
Feature Maps
Fully connected layer
Feature Extraction
Classification
Probabilistic distribution


Spectrogram
Generating Spectrogram
pre-processing
Audio Speech
Conv
Conv1 (7 × 7 × 16)
Conv2 (5 × 5 × 32)
Conv3 (3 × 3 × 32)
Conv4 (3 × 3 × 64)
Conv5 (3 × 3 × 64)
Conv6 (3 × 3 × 128)
Conv7 (3 × 3 × 128)
Flatten
Fully Connected
(512)
FC
number of classes
Softmax
Anger
Excited
Happy
Neutral
Sad

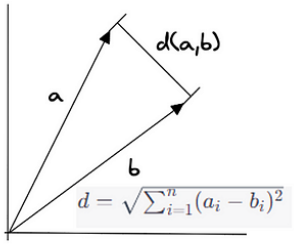
What is Symantic search





Similarity Metrics

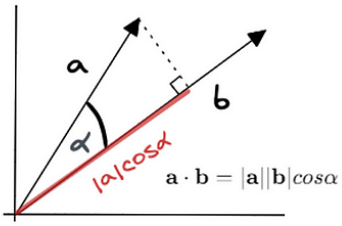
Euclidian (L2) Distance



Range=[0, infinity]

Smaller distance = greater similarity

Dot (Inner) Product

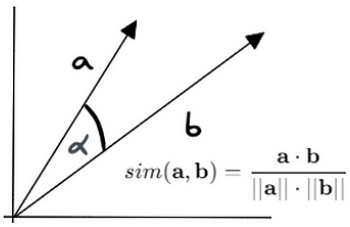


Range=[-infinity, +infinity]

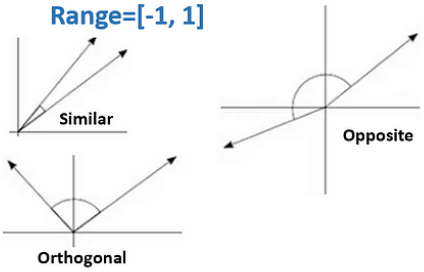
Larger dot product = greater similarity

Both the dot product & and Cosine similarity are closely related concept

Cosine Similarity



Range=[-1, 1]

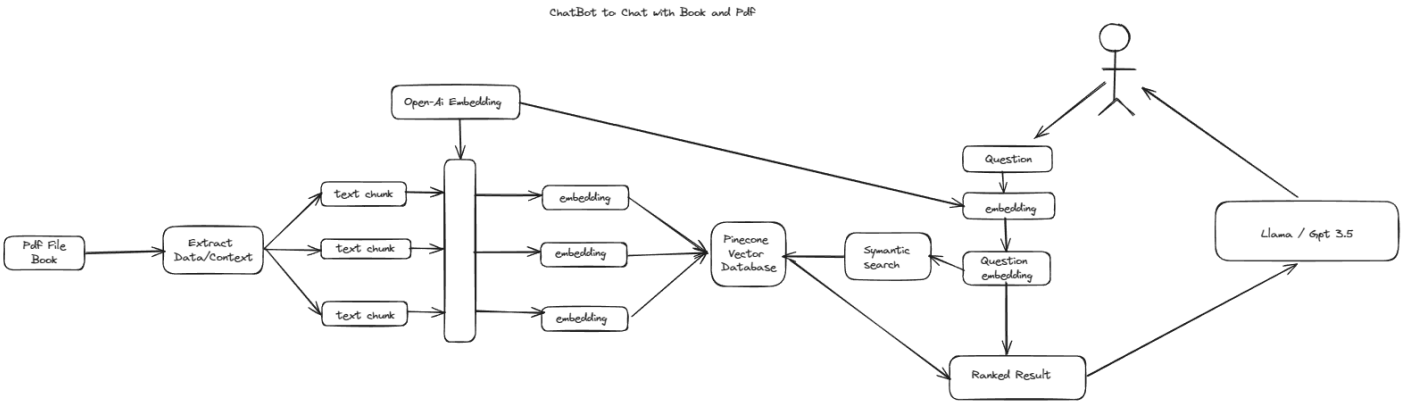


Larger Cosine value = greater similarity

Cosine similarity focuses solely on the angle between two vectors to assess their similarity, *disregarding the vector lengths*. In contrast, the dot product accounts for the individual lengths of each vector, making it an important factor to consider when selecting a similarity metric.

In the RAG framework, after generating embeddings for both the retrieved documents and the query input, the process of matching is typically performed using cosine similarity. Cosine similarity is a metric commonly used to measure the similarity between two vectors by computing the cosine of the angle between them. In this context, the embeddings serve as vector representations of the documents and the query input, encapsulating their semantic meaning and context. By calculating the cosine similarity between the query embedding and the embeddings of retrieved documents, the RAG model identifies the most relevant documents for augmentation. Documents with higher cosine similarity scores are considered more closely aligned with the query, indicating their relevance to the generation task at hand. This matching process ensures that the retrieved documents are contextually relevant to the input query, enhancing the quality and coherence of the generated output.

Developing Chat With Pdf Application



1. **PDF Upload and Chunking with LangChain:**
    - Users upload PDF documents into your application.
    - LangChain, a language processing tool, processes these documents by segmenting them into smaller, more manageable units known as chunks.
    - Chunking breaks down lengthy documents into meaningful sections, improving the efficiency of subsequent processing steps.

2. **Embedding Generation with OpenAI Embedding Model:**
    - Each chunk of the PDF document is passed through the OpenAI embedding model to generate vector representations, or embeddings, for the textual content.
    - The OpenAI embedding model encodes the semantic meaning and context of the text into high-dimensional vectors, capturing its essence in a numerical format.
    - These embeddings serve as compact and informative representations of the text, facilitating downstream tasks such as similarity matching and response generation.

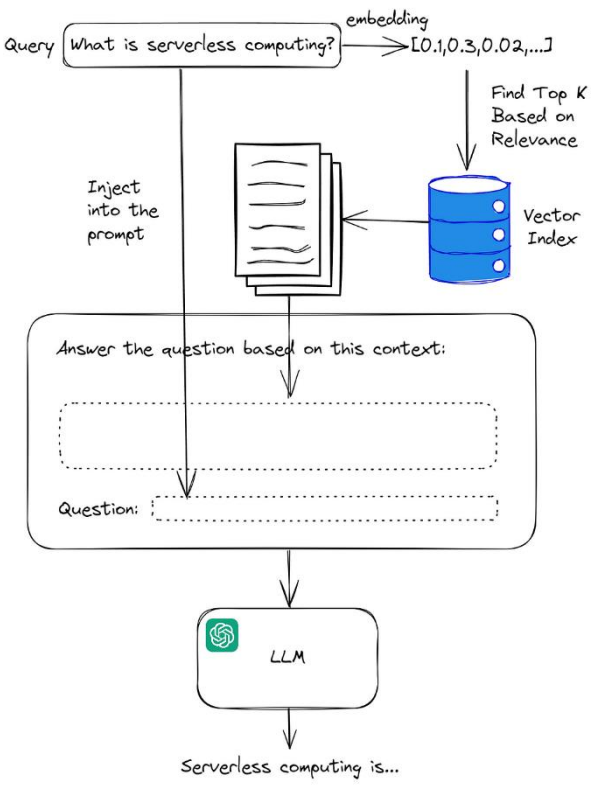
3. **Storage in Vector Database:**

- The generated embeddings are stored in conjunction with their corresponding textual data in a vector database.
- Vector databases are optimized for efficient storage and retrieval of high-dimensional vectors, enabling fast access to relevant information during query processing.
- This database serves as a repository of document embeddings, allowing for quick and scalable retrieval based on user queries.

4. **Query Processing and Embedding Calculation:**
    - Users input queries into the application to retrieve relevant information from the uploaded documents.
    - The query undergoes similar embedding generation using the OpenAI model, resulting in a vector representation of the query text.
    - This vector representation captures the semantic content of the query, enabling comparison with document embeddings to identify relevant matches.
    -
5. **Cosine Similarity Matching for Document Retrieval:**
    - Cosine similarity is computed between the embedding of the query and the embeddings of all documents stored in the vector database.
    - Cosine similarity measures the cosine of the angle between two vectors, indicating their similarity in direction and magnitude.
    - Documents with higher cosine similarity scores are deemed more relevant to the query, as they exhibit greater alignment in semantic space.
6. **Integration with ChatGPT API for Response Generation:**
    - Relevant documents identified through cosine similarity matching are retrieved from the vector database.
    - These documents, along with the query, are passed to the ChatGPT API for

response generation.

- ChatGPT utilizes state-of-the-art natural language processing techniques to generate coherent and contextually relevant responses based on the input query and retrieved documents.
- The generated responses are then presented to the user, providing valuable insights and information extracted from the uploaded PDF documents.



Benefits of Rag

1. **Enhanced Accuracy and Factual Grounding**: RAG verifies information against trusted sources using its rule-based system, integrates databases and fact-checking services, and prioritizes authoritative sources to minimize misinformation dissemination.

2. **Real-Time Information Access and Integration:** RAG connects to various data streams, APIs, and databases to access real-time information. Through continuous updates, it seamlessly integrates the latest data into generated content, ensuring relevance and timeliness.

3. **Reduced Reliance on Manual Training and Updates:** RAG's rule-based approach automates knowledge acquisition and adaptation. By dynamically adjusting to changes in data or requirements, it reduces the need for manual intervention, streamlining operations.

4. **Increased Transparency and Explainability:** RAG provides clear insights into its decision-making process by tracing back each step of its reasoning. It generates explanations alongside outputs, detailing the rules or criteria used, enhancing trust and allowing users to verify accuracy.

Application of Rag

1. **Personalized Customer Service:** RAG generates tailored responses to customer queries based on predefined rules and historical data, enhancing customer experience.

2. **Legal Research and Document Assistance:** RAG provides summaries of legal texts, assists in drafting documents, and offers insights based on established legal standards.

3. **Scientific Research and Literature Review:** RAG summarizes research articles, identifies relevant studies, and helps researchers stay updated with the latest advancements.

4. **Automated Report Generation and Analysis**: RAG processes raw data, generates reports highlighting trends and anomalies, and performs data analysis tasks efficiently.

CONCLUSION

Retrieval Augmented Generation (RAG) represents a significant advancement in Natural Language Processing (NLP). It overcomes limitations of traditional Large Language Models (LLMs) by seamlessly integrating them with external knowledge sources. This approach offers several key advantages:

- **Enhanced Accuracy and Factuality:** RAG mitigates the issue of factual errors and hallucinations common in LLMs. By retrieving relevant information from the knowledge base, RAG ensures responses are grounded in reality and supported by evidence
.
- **Improved Contextual Relevance:** RAG personalizes responses by considering the specific context of a user's query. It retrieves documents most aligned with the user's intent, leading to more informative and focused answers.

- **Domain-Specific Expertise:** RAG can be fine-tuned for specific domains by incorporating specialized knowledge bases. This empowers it to address complex questions within particular industries with greater accuracy and depth.

- **Adaptability and Continuous Learning:** Unlike traditional LLMs that require retraining for new information, RAG's knowledge base can be dynamically updated. This allows it to stay current with evolving information and adapt to new domains.

Beyond these core benefits, RAG unlocks several exciting possibilities:

- **Revolutionizing Search Engines:** RAG can power intelligent search engines that not only retrieve relevant documents but also synthesize information to provide comprehensive answers directly within the search results.

- **Enhanced Chatbots and Virtual Assistants:** Integrating RAG into chatbots and virtual assistants can lead to more informative and helpful interactions. Users can receive accurate and nuanced responses to their questions.

- **Personalized Education and Training:** RAG can personalize learning experiences by tailoring content and explanations to individual needs. This can significantly improve educational and training outcomes.

However, RAG also presents some challenges:

- **Knowledge Base Construction and Maintenance:** Building and maintaining a comprehensive and up-to-date knowledge base requires significant effort and resources.

- **Data Bias and Fairness:** The information stored in the knowledge base can perpetuate biases if not carefully curated. Mitigating bias in RAG systems is crucial.

Overall, Retrieval Augmented Generation offers a powerful and versatile approach to NLP tasks. Its ability to leverage external knowledge and continuously learn positions it as a key technology for shaping the future of human-computer interaction.