To construct a RAG-enabled system there are several components that need to be assembled. This includes creation and maintenance of the non-parametric memory, or a knowledge base, for the system. Another pipeline facilitates real-time interaction by sending the prompts to and accepting the response from the LLM, with retrieval and augmentation steps in the middle. Evaluation is yet another critical component, ensuring the effectiveness and accuracy of the system. All these components of the system are supported by a robust service infrastructure.

Design of a RAG-enabled system:
We have a need for two different pipelines:

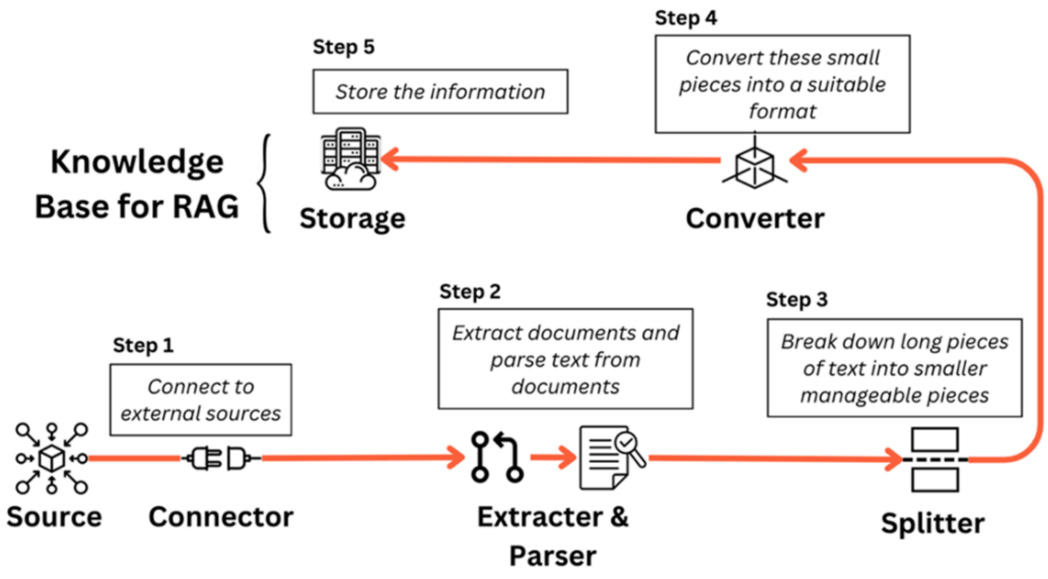
- We will call the pipeline that creates the knowledge base as the indexing pipeline
- The other pipeline that allows real time interaction with the LLM will be referred to as the generation pipeline.

Our system needs to facilitate five steps :

Step 1: User asks a question to our system
Step 2: The system searches for information relevant to the input question
Step 3: The information relevant to the input question is fetched, or retrieved, and added to the input question
Step 4: This question + information is passed to an LLM
Step 5: The LLM responds with a contextual answer

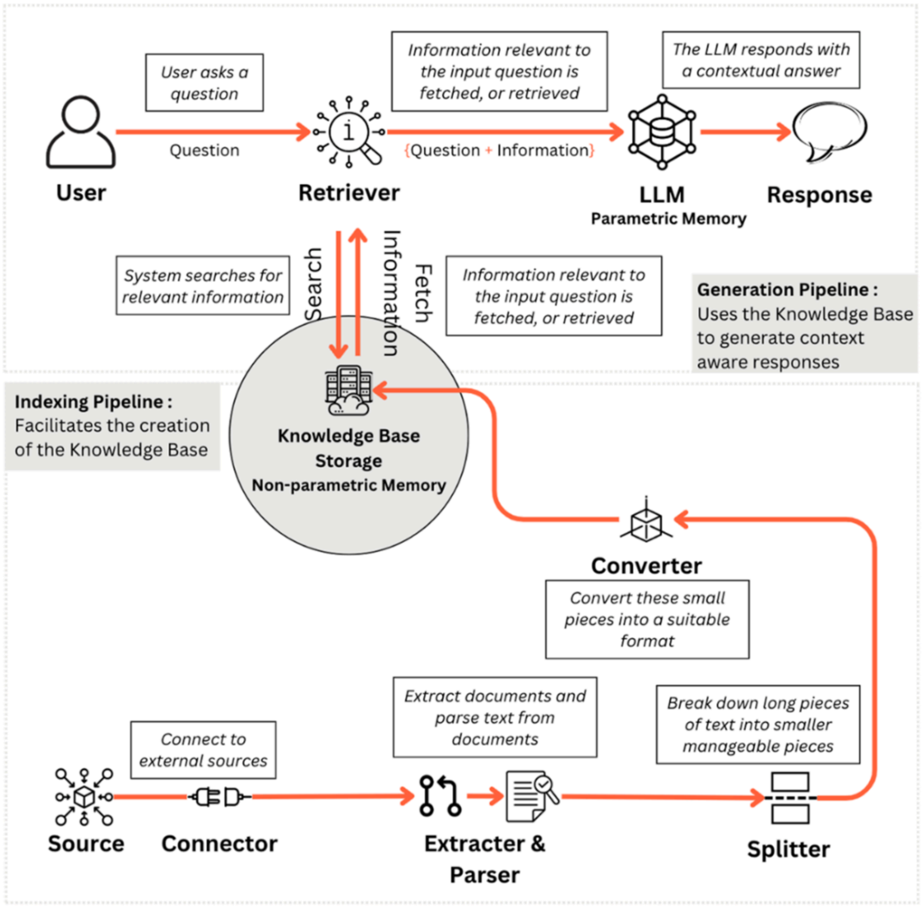
before this Generation Pipeline can be put in place. For that, key questions regarding the external source of information need to be answered.

What is the location of the external source of information?
Is it the open internet? Or are there some documents in the company's internal data storage? Is the information present in some third party databases? Are there multiple sources we want to use.
Why is this important?
We will need to know, in advance, where to look. We will also need to establish connections to all these disparate sources.
What is the nature of the information at source?
Are these word documents, pdf files? Is the information accessed via an API and response is in json format? Will we find answers to a question in one document or is the information distributed in multiple documents?
Why is this important?
We will also need to know the format and nature of data storage to be able to extract the information from the source files.

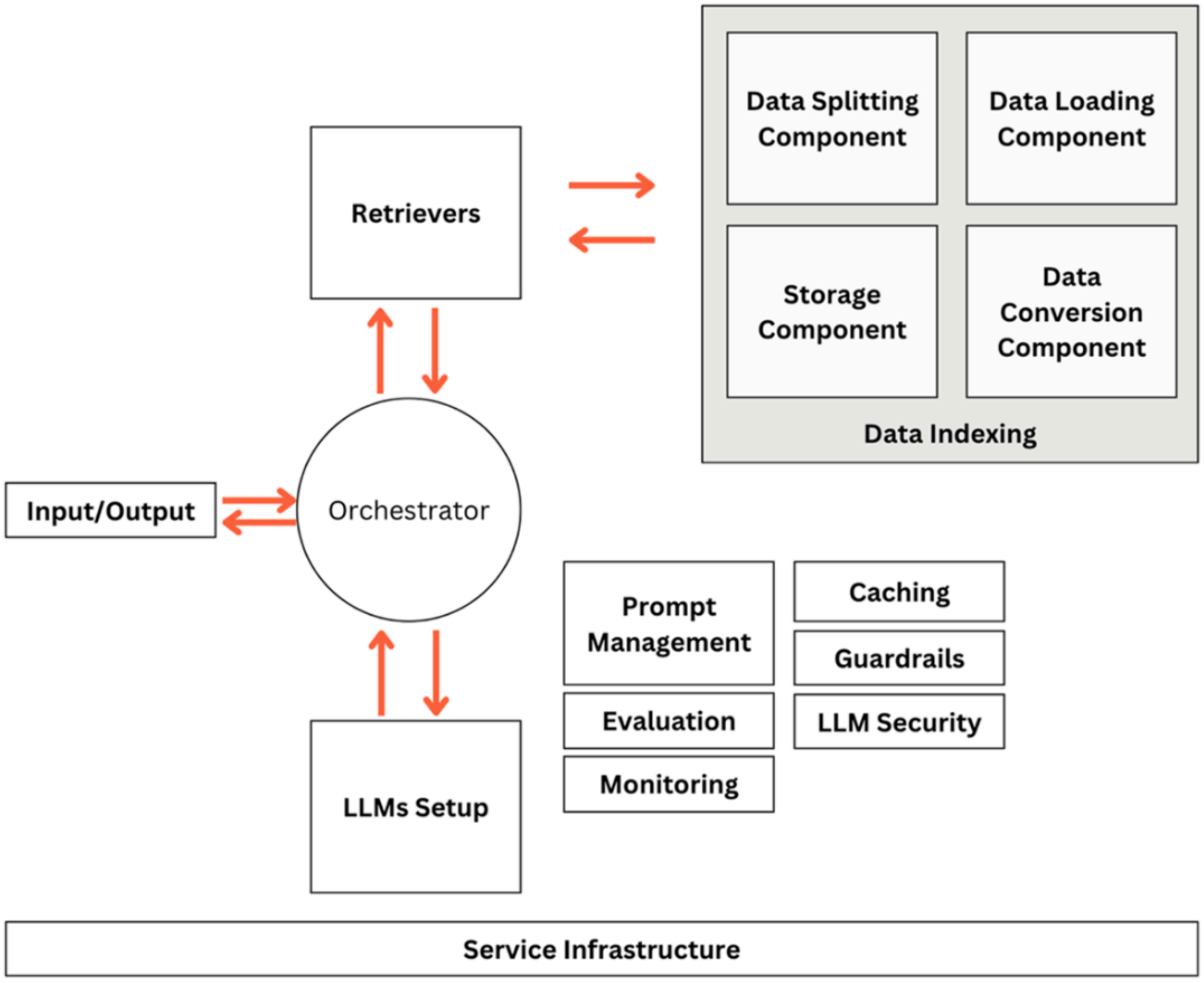

Step 5
Store the information
Step 4
Convert these small pieces into a suitable format
Knowledge Base for RAG
Storage
Converter
Step 2
Extract documents and parse text from documents
Step 3
Break down long pieces of text into smaller manageable pieces
Step 1
Connect to external sources
Source
Connector
Extracter & Parser
Splitter

Indexing Pipeline ^

Indexing + Generation Pipeline:


User asks a question
Information relevant to the input question is fetched, or retrieved
The LLM responds with a contextual answer
Question
{Question + Information}
User
Retriever
LLM
Parametric Memory
Response
System searches for relevant information
Search
Information
Fetch
Information relevant to the input question is fetched, or retrieved
Generation Pipeline : Uses the Knowledge Base to generate context aware responses
Indexing Pipeline : Facilitates the creation of the Knowledge Base
Knowledge Base Storage Non-parametric Memory
Converter
Convert these small pieces into a suitable format
Extract documents and parse text from documents
Break down long pieces of text into smaller manageable pieces
Connect to external sources
Source
Connector
Extracter & Parser
Splitter

Components of a Production Ready RAG system:



- **Data Loading** component : connects to external sources, extracts and parses data
- **Data Splitting** component : breaks down large pieces of text into smaller manageable parts
- **Data Conversion** component : converts text data into a more suitable format
- **Storage** component : stores the data to create a knowledge base for the system

These first four components complete the indexing pipeline

- **Retrievers** : are responsible for searching and fetching information from the Storage
- **LLM Setup** : is responsible for generating the response to the input
- **Prompt Management** : enables the augmentation of the retrieved information to the original input

These next three components complete the generation pipeline

- **Evaluation** component : measures the accuracy and reliability of the system before and after deployment
- **Monitoring** : tracks the performance of the RAG-enabled system and helps detect failures
- The **Service Infrastructure** : in addition to facilitating deployment and maintenance, ensures a seamless integration of various system components for optimal performance.

Other components include **caching** which helps store previously generated responses to expedite retrieval for similar queries, **guardrails** to ensure compliance with policy, regulation and social responsibility, and **security** to protect LLMs against breaches like prompt injection, data poisoning etc.

All these components are managed and controlled by a central **orchestration layer** which is responsible for the interaction and sequencing of these components. It provides a unified interface for managing and monitoring the workflows and processes.

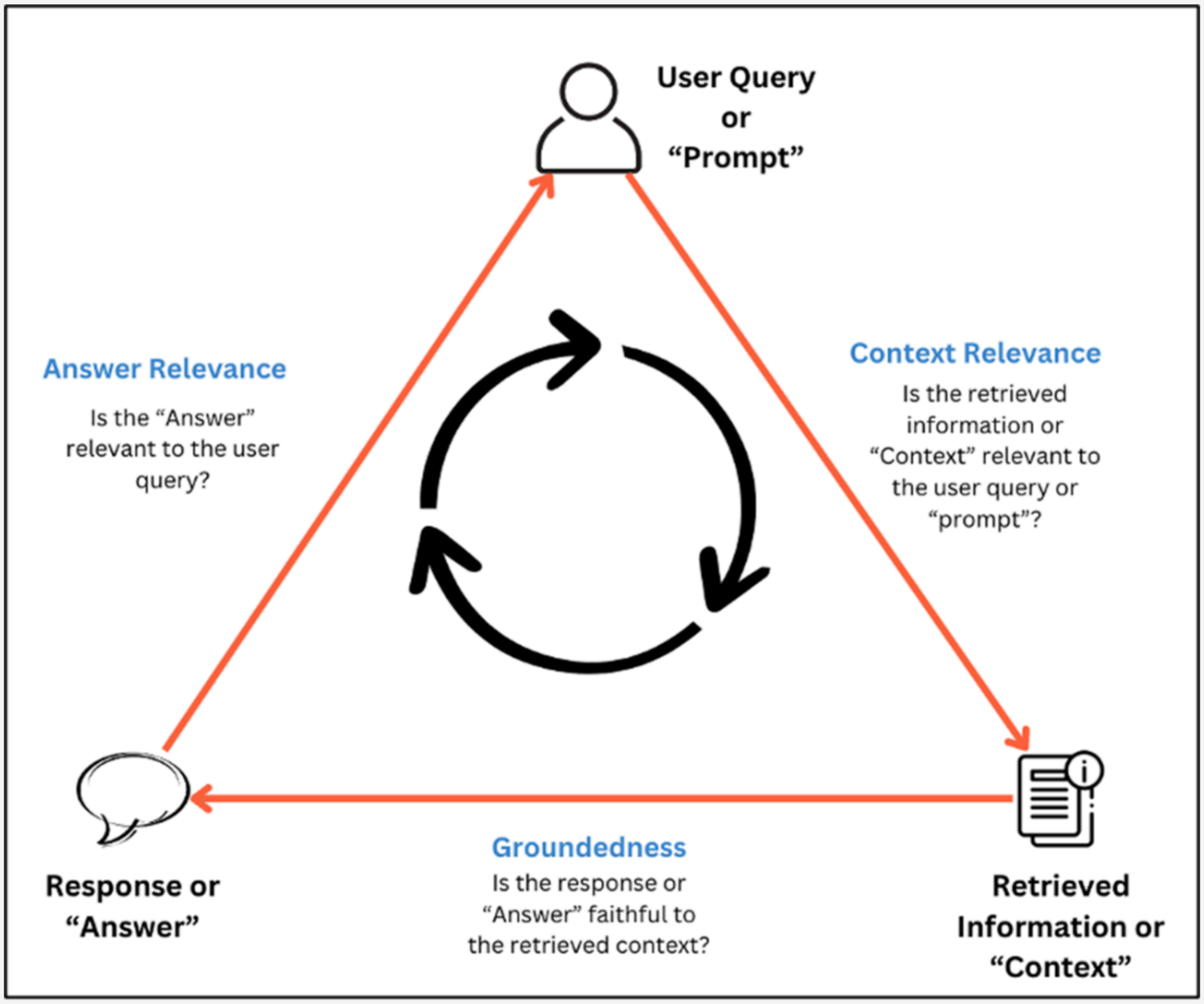
The Retriever : This is arguably the most critical component of the entire system. Using advanced search algorithms, the retriever scans the knowledge base to identify and retrieve the most relevant information based on the user's query. The overall effectiveness of the entire system relies heavily on the accuracy of the retriever. Also, search is a computationally heavy operation and may take time. Therefore, the retriever, also contributes heavily to the overall latency of the system.

Prompt Management: Once the relevant information is retrieved by the retriever, it needs to be combined, or augmented, with the original user query. Now, this may seem a simple task at the first glance. However, the construction of the prompt makes significant difference to the quality of the generated response. This component also falls in the gambit of prompt engineering.

LLM Setup: At the end, Large Language Models (LLMs) are responsible for generating the final response. A RAG enabled system may rely on more than one LLM. The LLMs can be the foundation (base) models that have been pre-trained and generally available either open source, like those by Meta or Mistral, or through a managed service, like OpenAI or Anthropic. LLMs can also be fine-tuned for specific tasks. Fine-tuning involves training pre-existing LLMs on specific datasets or tasks to improve performance and adaptability for specialized applications.

Evaluation and Monitoring

The triad of RAG evaluation proposed by TruEra


User Query or "Prompt"
Answer Relevance
Is the "Answer" relevant to the user query?
Context Relevance
Is the retrieved information or "Context" relevant to the user query or "prompt"?
Groundedness
Is the response or "Answer" faithful to the retrieved context?
Response or "Answer"
Retrieved Information or "Context"

There are several metrics that help assess each of these three dimensions. For some of the metrics a “ground truth” dataset is warranted. Ground truth datasets provide a benchmark for evaluating the accuracy and effectiveness of RAG-enabled systems by comparing generated responses to manually curated references.

Service Infrastructure

he infrastructure can be understood in several layers.

1. **Data Layer** : Tools and Platforms used to process and store data in form of embeddings
2. **Model Layer** : Providers of proprietary or open-source LLMs.
3. **Prompt Layer** : Tools offering maintenance and evaluation of prompts
4. **Evaluation Layer** : Tools and frameworks providing evaluation metrics for RAG
5. **App Orchestration** : Frameworks that facilitate invocation of different components of the system
6. **Deployment Layer** : Cloud providers and platforms for deploying RAG enabled LLM based apps
7. **Application Layer** : Hosting services for RAG enabled LLM based apps
8. **Monitoring Layer** : Platforms offering continuous monitoring of RAG enabled LLM based apps

Caching, Guardrails and Security

Finally, there are certain other components that are used frequently in RAG enabled systems. These components address the issues of system latency, regulatory and ethical compliances among other aspects.

- **Caching** : Caching is the process in which certain data is stored in cache memory for faster retrieval. LLM caching is slightly different from regular caching. The LLM responses to queries are stored in a semantic cache. Next time, a similar query is asked, the response from the cache is retrieved instead of sending the query through the complete RAG pipeline. This improves the performance of the system by reducing the time it takes to respond, by reducing the cost of LLM inferencing and by reducing the load on the LLM service.
- **Guardrails** : For several use cases, in practice, there will be a set of boundaries within which the output needs to be generated. Guardrails are pre-defined set of rules that are added in the system to comply with policies, regulations and ethical guidelines.
- **Security** : LLMs and RAG enabled LLM based applications have observed new kind of threats like prompt injections, data poisoning, sensitive information disclosure and others. With evolving threats, the security infrastructure also needs to evolve to address concerns around security and data privacy of RAG enabled systems.