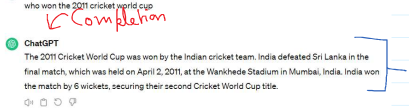
Prompt Engineering

28 March 2024 11:55 PM

Prompt ⇒ The input that you send into a generative model (text, images, videos etc)

Completion ⇒ Output

Example :



Prompt

You
who won the 2011 cricket world cup

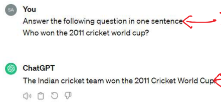
Completion

ChatGPT
The 2011 Cricket World Cup was won by the Indian cricket team. India defeated Sri Lanka in the final match, which was held on April 2, 2011, at the Wankhede Stadium in Mumbai, India. India won the match by 6 wickets, securing their second Cricket World Cup title.

Very Chatty

Prompt Structure:-

Instruction ⇒

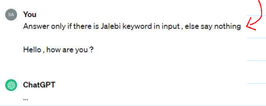


You
Answer the following question in one sentence ← Instruction
Who won the 2011 cricket world cup?

ChatGPT
The Indian cricket team won the 2011 Cricket World Cup. ← single sentence response

Conditional Prompts:-

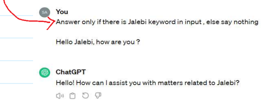
Condition



You
Answer only if there is Jalebi keyword in input , else say nothing

Hello , how are you ?

ChatGPT
...



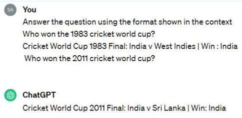
You
Answer only if there is Jalebi keyword in input , else say nothing

Hello Jalebi, how are you ?

ChatGPT
Hello! How can I assist you with matters related to Jalebi?

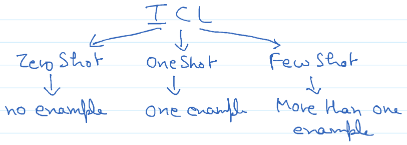
In-Content Learning :-

Allows model to perform taste with a minimal amount of training examples.



You
Answer the question using the format shown in the context
Who won the 1983 cricket world cup?
Cricket World Cup 1983 Final: India v West Indies | Win : India
Who won the 2011 cricket world cup?

ChatGPT
Cricket World Cup 2011 Final: India v Sri Lanka | Win: India

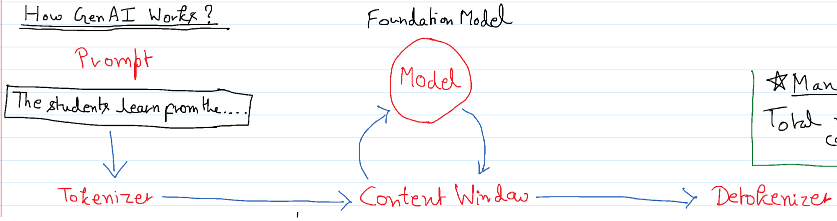


Tokens:-

The smallest unit into which text data can be broken down for all AI model to process.

OpenAI Tokenizer

How GenAI Works ?



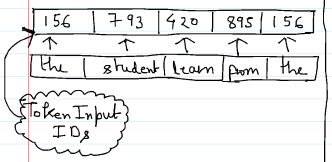
☆ Max Token:-

Total no. of tokens Generative AI can generate in response.

A very basic mechanism to keep

can generate in response.

Tokenizer → Content Window → Detokenizer



Max Amount of tokens, model consider while generating response

In practical terms, the content window limits how much previous dialogue the model can remember during an interaction.

This includes both prompt provided by user & model's generated text. } → Short Term Memory

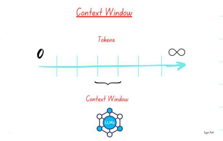
If the interaction exceeds the content window, the model loses access to the earliest part of the conversation.

It is a barrier to long conversation/complex task

A very basic mechanism to keep model response short.

V.V.I. ⇒ Performance Impact

Tokens ↑ Inference Time ↑



Greedy Vs. Random Sampling :-

Prompt → Model → Probability Distribution across all tokens in the model's known vocabulary

↓

Choose a single token

Greedy → Choose next token with heighest probability

The sky is ____

Probability	Token
0.3	blue
0.4	Limitless
0.2	Clear
⋮	⋮

Random

The sky is ____

Probability	Token
0.3	blue
0.4	Limitless
0.2	Clear
⋮	⋮

Top P & Top K Random Sampling :-

Top K → Choose token randomly from only top k tokens with heighest probability

Probability	Token
0.3	blue
0.4	limitless
0.2	Clear
⋮	⋮

⟶

Probability	Token
0.4	limitless
0.3	blue
0.2	Clear
⋮	⋮

} $k = 2$ (limitless, blue)

$\boxed{\text{Top } K = 1} \Rightarrow$ Greedy Sampling

Top P → Randomly sampling from the set of tokens whose cumulative probability do not exceed p, starting from heighest probability and working down to the lowest.

$p = 0.7$

Probability	Token
0.3	blue
0.4	limitless
0.2	Clear
⋮	⋮

⟶

Probability	Token
0.4	limitless
0.3	blue
0.2	Clear
⋮	⋮

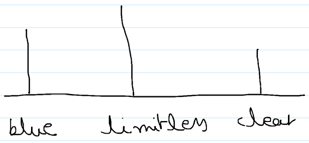
} $p = 0.7$ (limitless, blue)

Temperature :-

top-p / top-K → affects next token prediction after probability distribution is generated

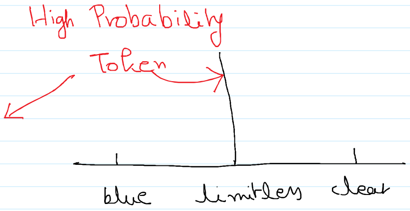
temperature → changes next-token probability distribution → ultimately affect next token prediction.

Probability	Token
0.3	blue
0.4	limitless
0.2	Clear
⋮	⋮



Temp ↑ Randomness ↑

Probability	Token
0.01	blue
0.5	limitless
0.02	Clear
⋮	⋮



$\underbrace{\text{Temp} \downarrow \quad \text{Randomness} \downarrow}$

low temp ⇒ Strong peaks

Low temp, Low Top-p ⇒ highly focused on narrow range of high probability token.

Probability	Token
0.02	blue
0.5	limitless
0.01	Clear
⋮	⋮

$\xrightarrow{\text{sorting}}$

Probability	Token
0.5	limitless
0.02	blue
0.01	Clear
⋮	⋮

High Temp, High Top-p ⇒ very high randomness.

temp=2, p=1

Probability	Token
0.3	blue
0.4	limitless
0.2	Clear
⋮	⋮

$\xrightarrow{\text{Sorting}}$

Probability	Token
0.4	limitless
0.3	blue
0.2	Clear
⋮	⋮

$p = 1$

Low Temp, High Top-p ✓

Probability	Token
0.02	blue
0.5	limitless
0.01	Clear
⋮	⋮

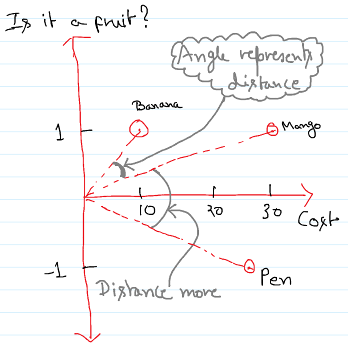
$\longrightarrow$

Probability	Token
0.5	limitless
0.02	blue
0.01	Clear
⋮	⋮

high p

Embedding Vectors :-

	Is it a fruit?	Cost?	
Mango →	1	30	→ [1, 30]
Banana →	1	10	→ [1, 10]
Pen →	−1	25	→ [1, 25]



Cosine graph

$\vec{a} \cdot \vec{b} = |a|\,|b| \cos\theta$ ← Mother Equation

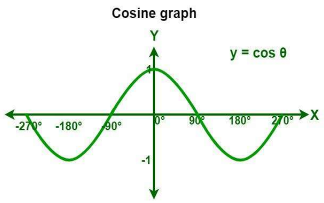
$$\boxed{\cos\theta = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}|\,|\vec{b}|}}$$

$\vec{a} = a_x\hat{i} + a_y\hat{j} + a_z\hat{k}$

$\vec{b} = b_x\hat{i} + b_y\hat{j} + b_z\hat{k}$

Pen ⟶ Is it a fruit? −1 | Cost? 25 ⟶ [1, 25]

$\vec{b} = b_x\hat{i} + b_y\hat{j} + b_z\hat{k}$



$$\vec{a}\cdot\vec{b} = a_x b_x + a_y b_y + a_z b_z$$

$$|\vec{a}| = \sqrt{a_x^2 + a_y^2 + a_z^2}$$

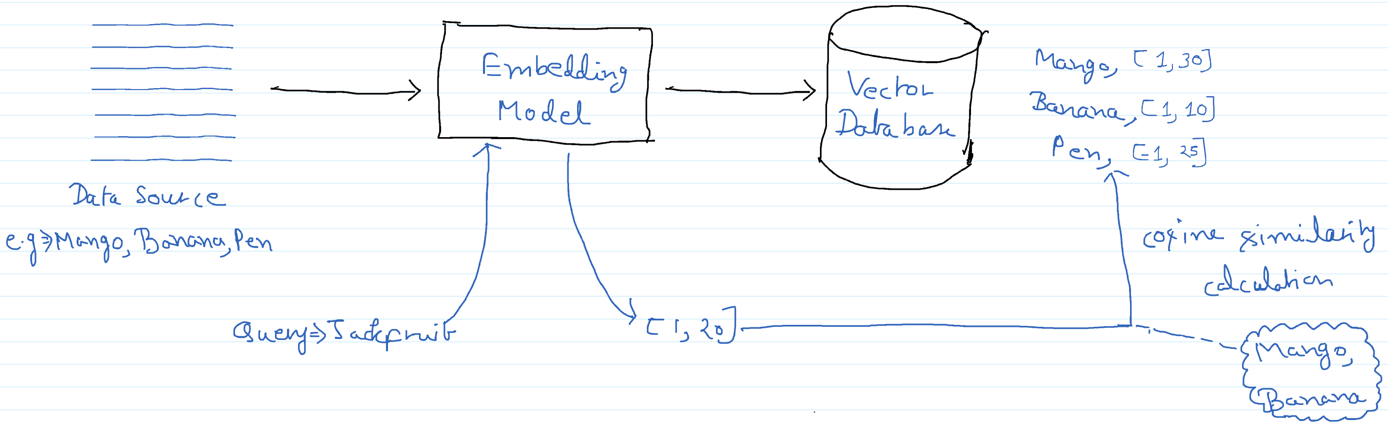
$$|\vec{b}| = \sqrt{b_x^2 + b_y^2 + b_z^2}$$

$$\boxed{\cos\theta = \frac{a_x b_x + a_y b_y + a_z b_z}{\left(\sqrt{a_x^2 + a_y^2 + a_z^2}\right)\left(\sqrt{b_x^2 + b_y^2 + b_z^2}\right)}}$$

Embedding are numerical, vectorized representation of any type, including text, images, audio clips, videos etc.

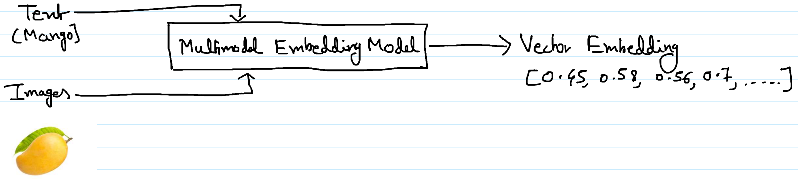
Semantic Search:-

Semantic Search
- ⟶ Traditional keyword matching based output ✗
- ⟶ Return output based on meaning & context behind user queries. ✓

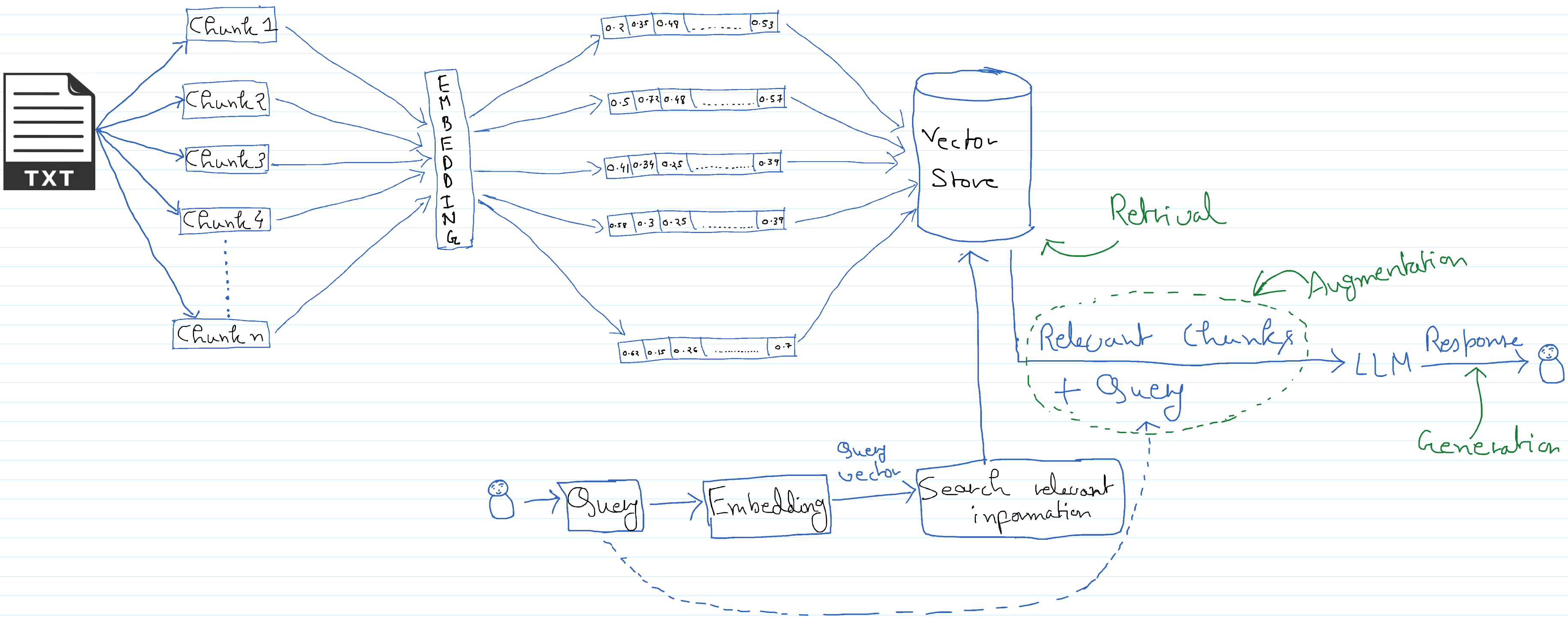


Multimodal Embedding:

↳ Integration of Text, Images & various other datatypes in single vector space.



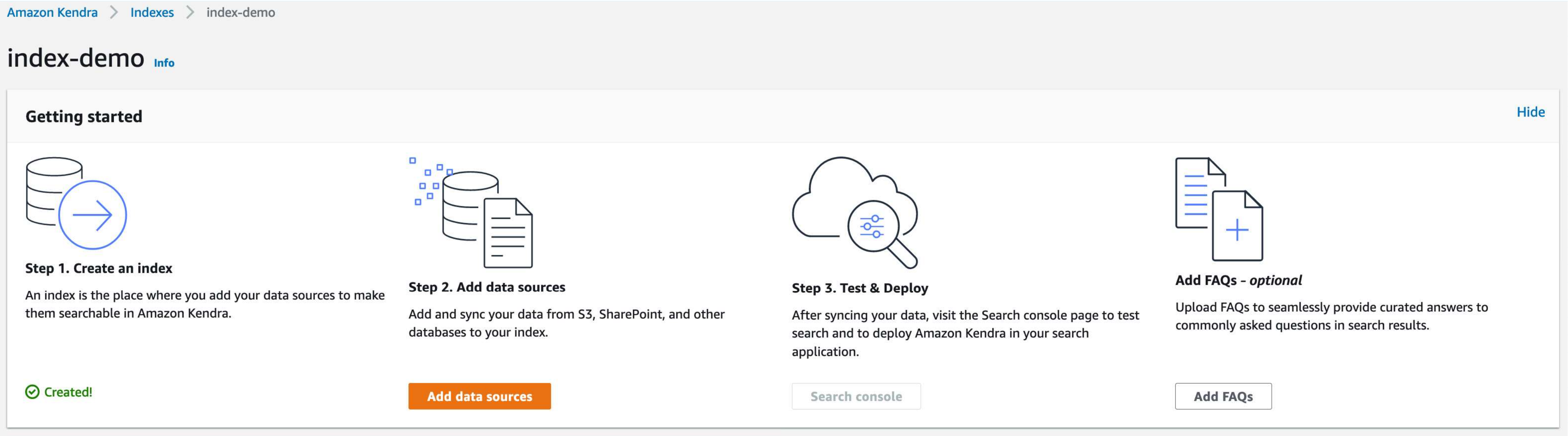
RAG :-



RAG → Combination of 2 memories

model's own prior knowledge

A search engine



Amazon Kendra > Indexes > index-demo

index-demo Info

Getting started

Hide

Step 1. Create an index

An index is the place where you add your data sources to make them searchable in Amazon Kendra.

✓ Created!

Step 2. Add data sources

Add and sync your data from S3, SharePoint, and other databases to your index.

Add data sources

Step 3. Test & Deploy

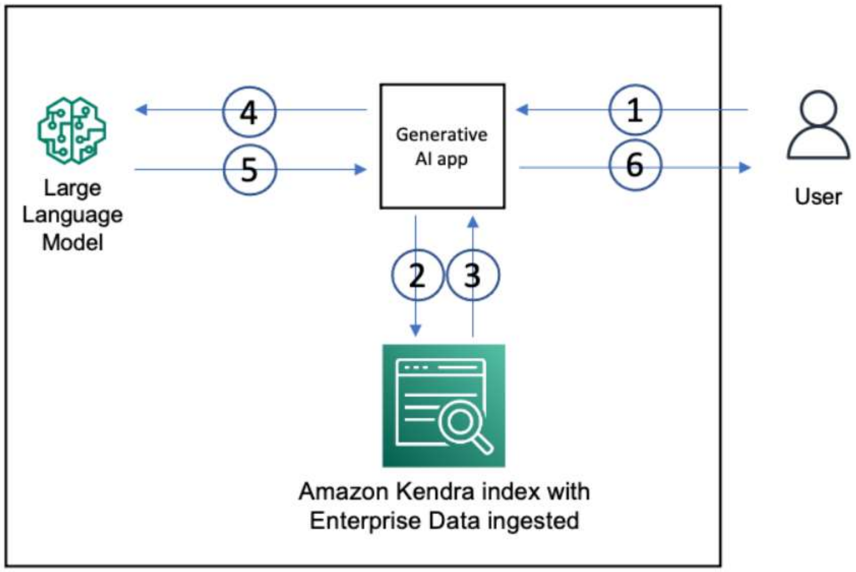
After syncing your data, visit the Search console page to test search and to deploy Amazon Kendra in your search application.

Search console

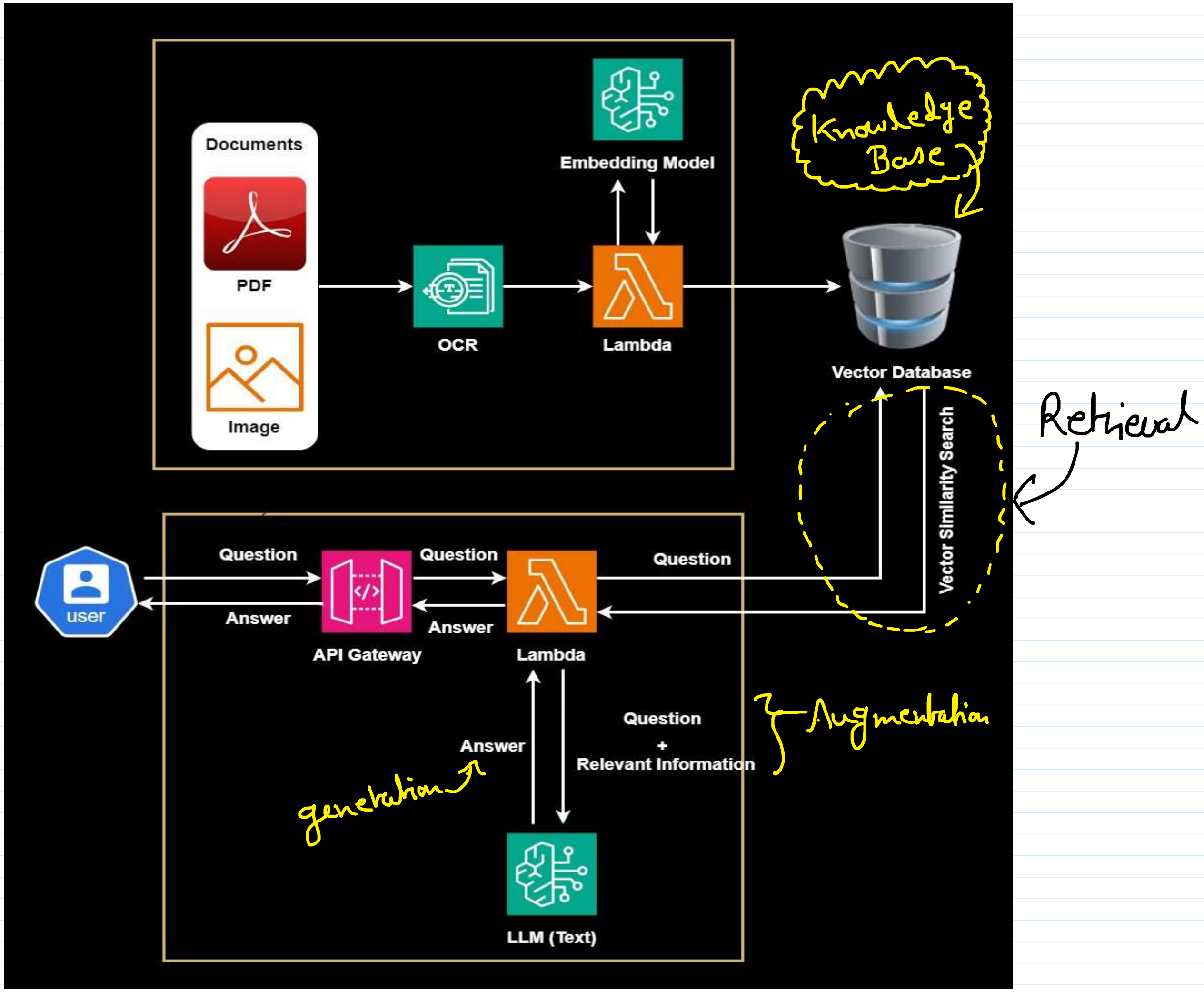
Add FAQs - *optional*

Upload FAQs to seamlessly provide curated answers to commonly asked questions in search results.

Add FAQs



Rag Implementation:-


Documents
PDF
Image
OCR
Embedding Model
Lambda
Vector Database
Knowledge Base
Vector Similarity Search
Retrieval
user
Question
Answer
API Gateway
Question
Answer
Lambda
Question
Answer
Question
+
Relevant Information
Augmentation
generation
LLM (Text)