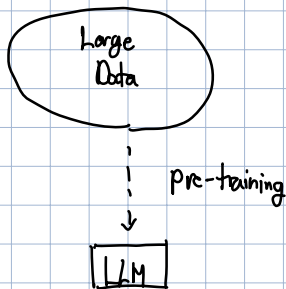


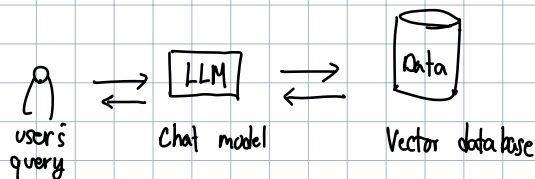
Before RAG (traditional LLM models)



Problems:

- ① knowledge cut-off: If user's query required the most recent news where the pre-trained model haven't been exposed to, the model simply didn't know or "hallucinated".
- ② hallucination: LLMs are "probability machines". They prioritize smoothness over accuracy. Without a source to check, they often made up fake facts.
- ③ Privacy & Business data: inability to teach entire new large sets of data without retraining the entire model.

Solution: RAG: Retrieval-Augmented Generation



- ① up-to minute accuracy: No need to train the LLM. only upload documents that the LLM needs to reference.
- ② Citations: tracking sources is possible (seeing if hallucination occurred or not). by referencing a tool online called langsmith.
- ③ Cost-effective: Since retraining the model itself is not required, uses much less computational energy cost.

RAG Implementation

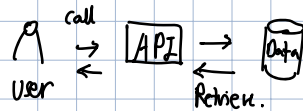
• 3 key words

- ① API
- ② Huggingface
- ③ Langchain

① API: Application Programming Interface

- set of rules and protocols allowing different software applications to communicate, share data, and utilize the functionality.

ex.)



ex.) `repo_id = "google/gemma-2-9b-it"` ← one of huggingface's free models.

② Huggingface

- Online opensource for pre-trained ML models.

can be used simply for 2 types of model.

- ① Embedding model
- ② LLM model.

ex.)



has different types of models that performs embedding, NLP, Object detection, etc.

③ Langchain.

- Opensource Framework for the ease of application development.

3 main key highlights about langchain.

① Abstraction

- Simplifies complex RAG development tasks into few lines of code.

ex.) developing RAG:

1. Upload document
2. Splitting text
3. Embedding to vector
4. Saving to Vector Data Base.

↳ traditionally, this would take many lines of code, but using langchain tools, it makes it much quicker.

② Standardization

- creates a "standard shape" for different AI models.

LLM sources

- Chat GPT
- Grog
- Gemini
- Claude
- ⋮

Standardization
→

Chat model

- ChatAnthropic
- ChatGrog
- Chat Google Generative AI.
- ⋮

- Since most LLMs have different endpoints, APIs, changing one model would require many changes in the code.

↳ langchain allows to change single line of code

③ Chaining

- Most components are designed with input → output structure. and using langchain can "chain" multiple sources under few/single of code.

ex.)

Prompt	chat-LLM	Str Output Parser()
--------	----------	---------------------

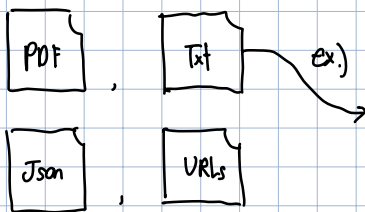
↑
pipe

Implementation Flow diagram -

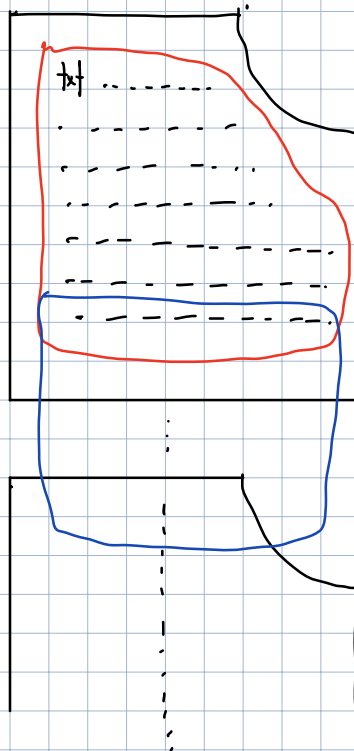
① Indexing.

- Upload documents
- Split text within the document
- Embed each splits
- Save to a vector database.

a.) Load.



b.) Split text within document

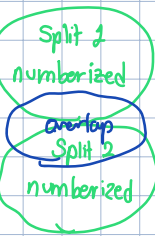
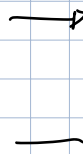
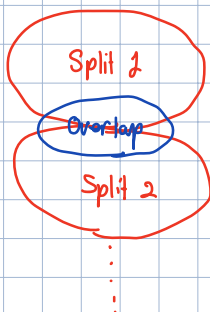


Chunk size = 1000
overlap = 50

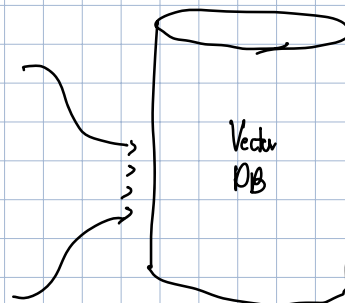
- doing so allows consistency check for the LLM.

(can use free LLM from HuggingFace)

c.) Embed splits.

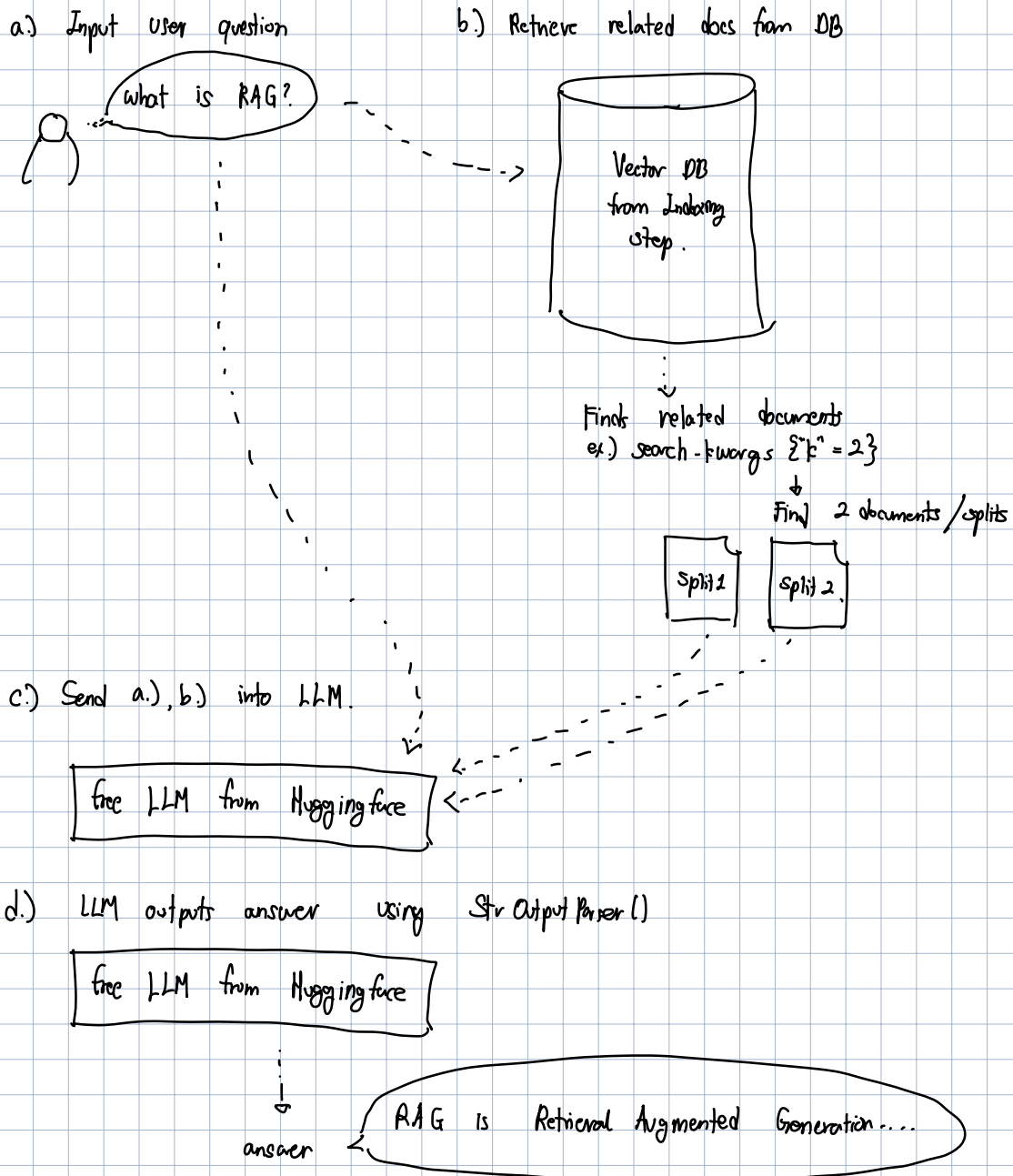


d.) Store to Vector database



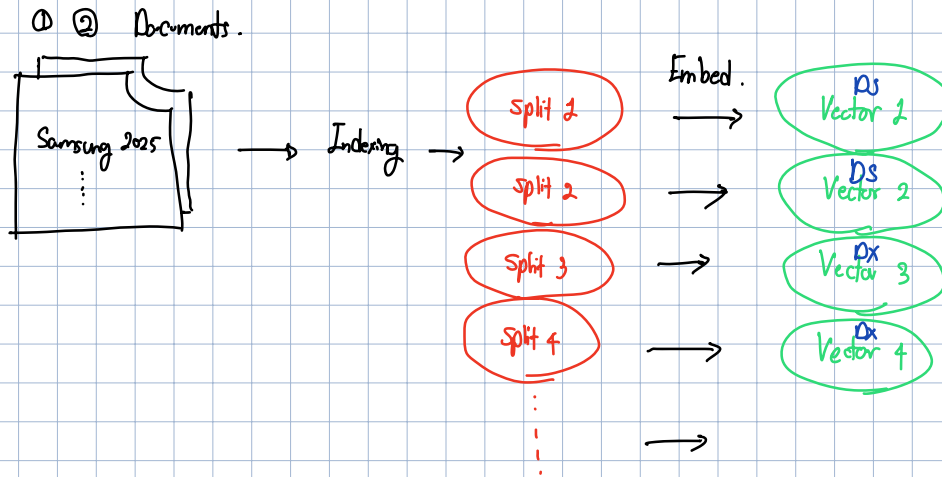
② Query

- a.) Input user question
- b.) Retrieve from Vector DB
- c.) Send step a.), b.) into LLM
- d.) LLM outputs answer.

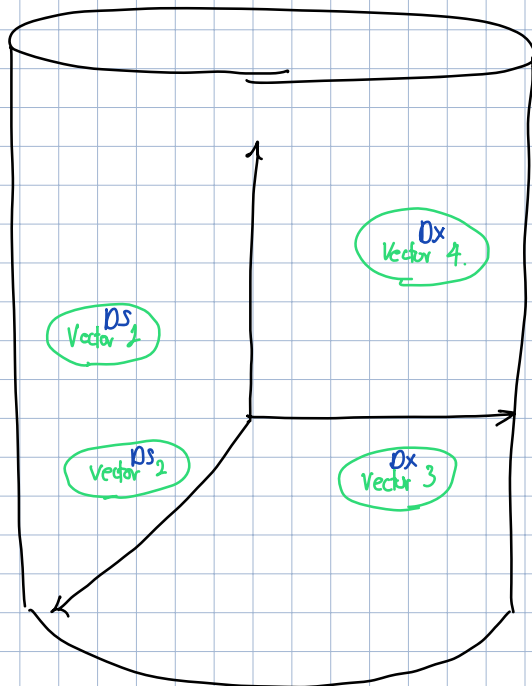


Visualized simple example.

Documents : ① " Samsung 2025 Q4 earnings call PDF"
② " Samsung 2025 Q4 earnings call Presentation "

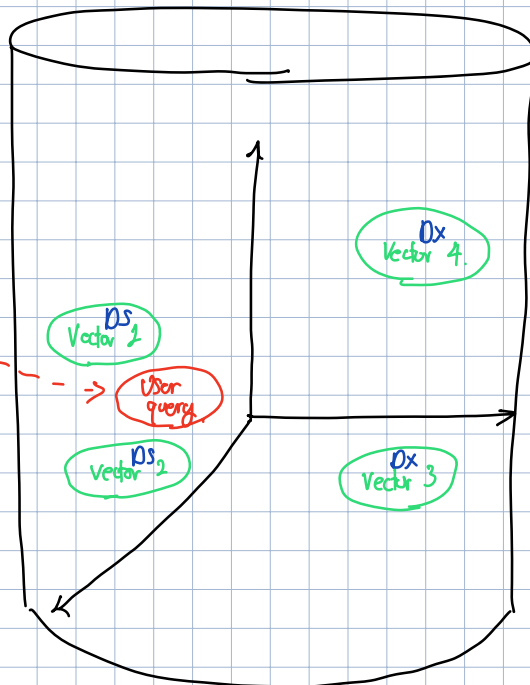


Inside Vector DB

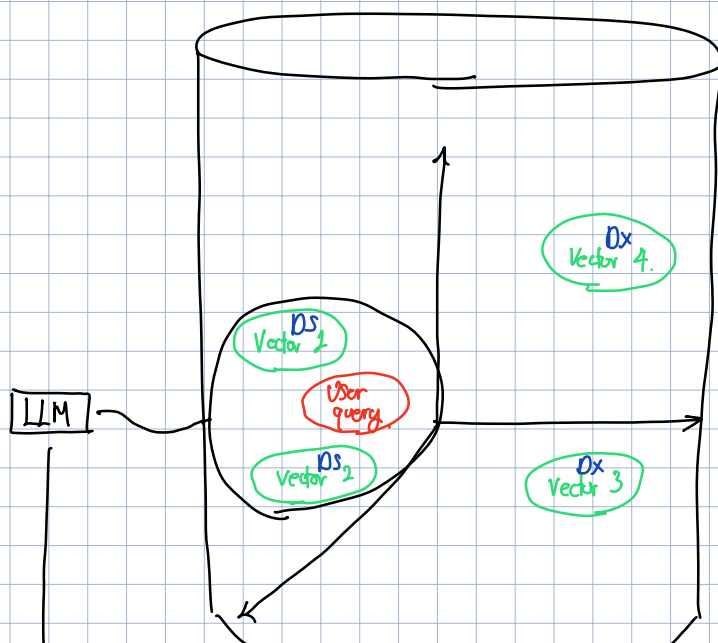


User's Query: "What is the Samsung Electronics 2025 Q4 Earnings in US department?"

Embedded into Vector



- LLM finds the two closest neighbors (splids) with regards to where the user's query is located and fetches them.



↓
Summarize in natural language
and return answer

