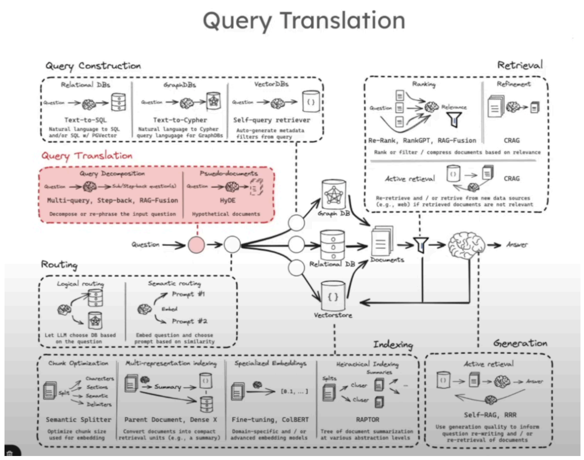
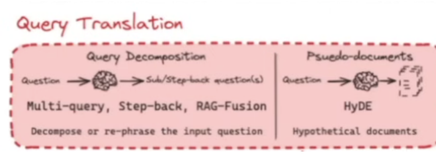
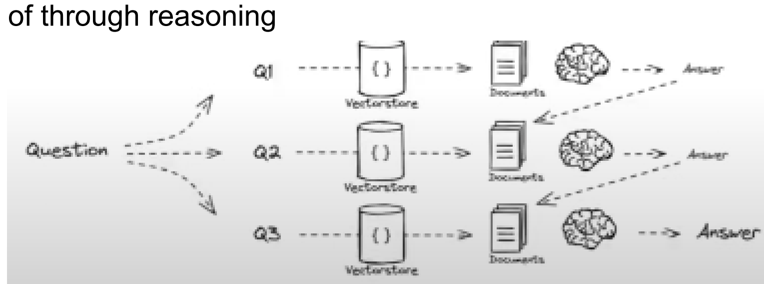
Learn RAG From Scratch – Python AI Tutorial from a LangChain Engineer

- https://youtu.be/sVcwVQRHIc8?list=PLCkMWeGDjQDKWu1IFdhykSMcjnn0SJA0x

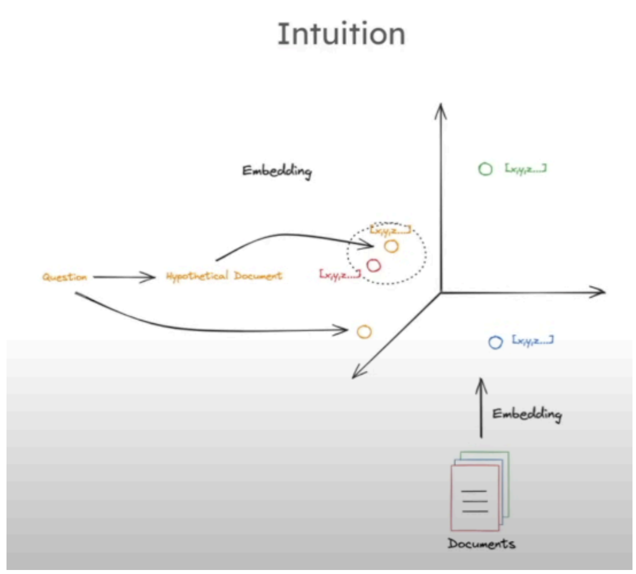
- # Query Translation
    - 



    - Rewriting strategies
        - Multi Query
            - Enter a query and ask a LLM to rewrite it in 4-5 different ways then normal RAG with all retrieved documents
        - RAG Fusion
            - Same as multi query but rank the document and take the top 5
                - Ranking is given out by which documents are retrieved the most frequent from each of the queries in the Retrieval phase
    - Decomposition or Least to most (by google)
        - Chain of through reasoning
            - 

            - 1. Break question into smaller sub questions (ex.3)
                - Have to use specific prompt to get questions that build on each other, can help one another, and be answered in full.
            - 2. Answer first question with rag

        - 3. Answer second question with rag + (first question + answer) into LLM
        - 4. Answer third question with rag with (first question + answer AND second question + answer) into LLM
        - 5. Using rag on original question + (all context above) into LLM
    - Alternate way
        - 

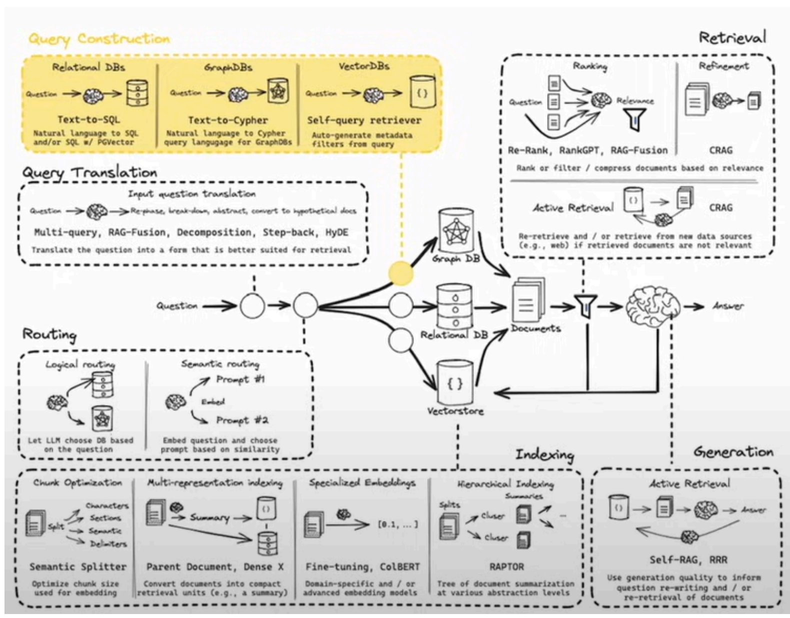
        - Answer each question individually then add together at the end
            - We don't need a super specific prompt the break down the initial question
- Step-Back question (by google)
    - Take a specific question and find a more abstract question that is typically easier for a LLM to answer
        - In the initial prompt provide examples of different questions and a appropriate output can the LLM can figure out the pattern and then replicate it
    - Combines the documents from both the step back question and the documents form the original question to enter in a LLM with the original question as the final response
- HyDE
    - Because Documents and questions are very different in nature the embeddings from each may not match accordingly, thus HyDE will turn a question into a HYPOTHETICAL document then compare the embeddings from this with the embedding to the documents
    - 

    - BUT in the final RAG prompt does not include the hypothetical document

Routing

- Structured output
  - Use a LLM to decide if the prompt should go to which data source
    - Knowledge based approach
- Semantic Routing
  - Embedding prompts or description of the DB such as a physics or math DB to a physics or math prompt and then compare both prompts to the user's question
    - question has to be embedded first before comparison

Query construction

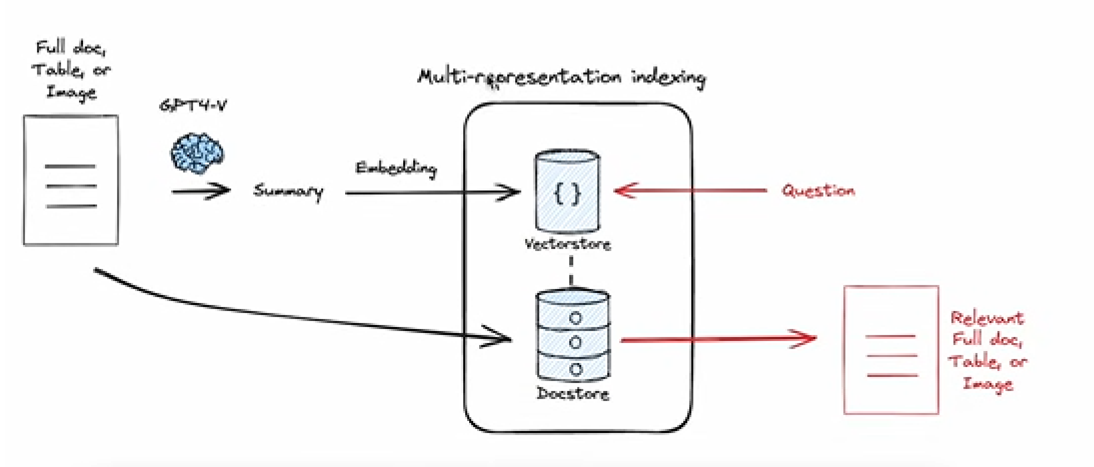
- 

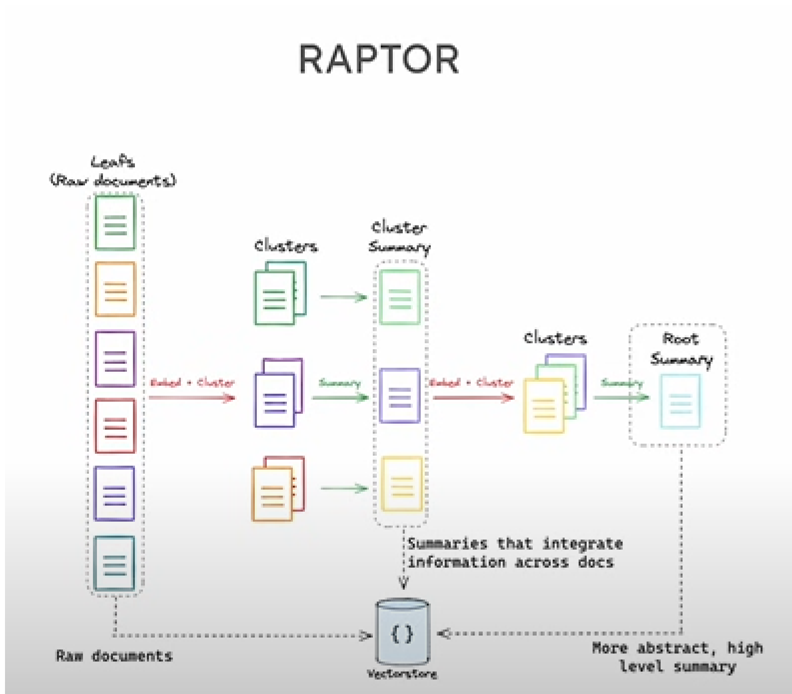
- From the prompt you have to turn the question into a acceptable format to query each of the DB's so it would have to be in
  - {"content_search": ….., "document_date:" ….., (OTHER PARMATERS)}
  - You can Acchive this by using a LLM with example output prompts

Indexing

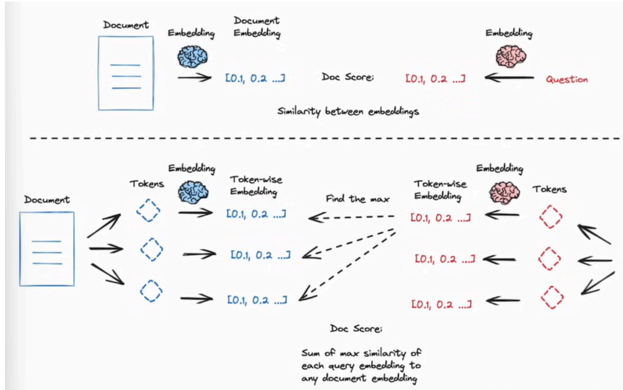
- Multi Representation indexing
  - Lol just making a summary of each document before indexing

- But the prompt to LLM when summarizing is asked in a way to optimize the summary for indexing

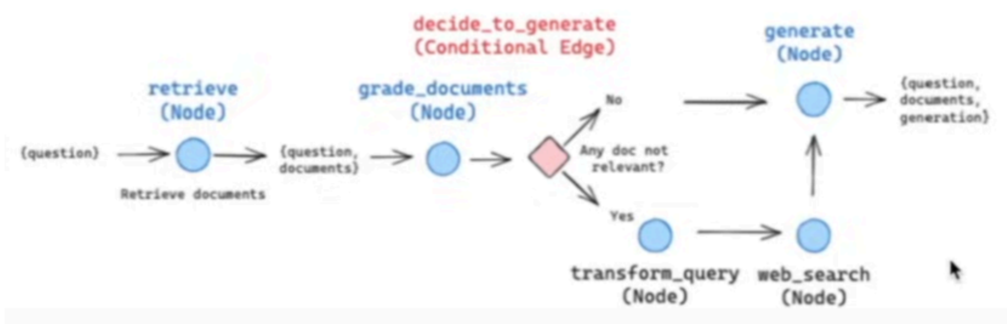


- RAPTOR
  - Create a hierarchy of documents and summaries throught he use of recursively joining similar chunks/documents and summarizing them
    - This helps to answer a wide range of high and low level questions
  - 

- ColBERT
  - Turn both documents and question into tokens and then compare these tokens to one another and then score them by *???*

- ■ 

https:/

- CRAG
  - DO retrieval but if no documents are relevant you web search and use that context instead
    - ■ Deciding if the documents are relevant are done by a “grader”
  - 

- Adaptive RAG
  - Have unit testing at certain points in RAG
    - ■ Each unit test will have a fall back like a re-run or switching paths
  -