Brance <Position>Task

Name:Parth Gupta

Linkedin Profile:[**https://www.linkedin.com/in/parth-gupta-171614184/**](https://www.linkedin.com/in/parth-gupta-171614184/)

Date Challenge Received:25/07/2023

Date Solution Delivered:27-07-23



1. Problem Statement

The task was to build a RAG(Retrieval Augmented Generation) chatbot. For user questions, the RAG module would retrieve context from knowledge document and generation phase LLM would personalize answers using retrieval knowledge from the retrieved context.

2. Approach

The task is to get the context about the query from existing knowledge, find the best answers, and then generate a coherent response to the question. So, the problem can be broken down to 3 broad categories

* **Knowledge Document Organization**
* **Context Retrieval**
* **Response Generation**

Knowledge Document Organization: How to structure and store the knowledge document so as to be able to efficiently search over it about the query at the same time making sure that the context is not lost in the structurization.

Context Retrieval:If we have a storage of knowledge, how to find the most relevant contexts to find the answer to the query out of all the knowledge.

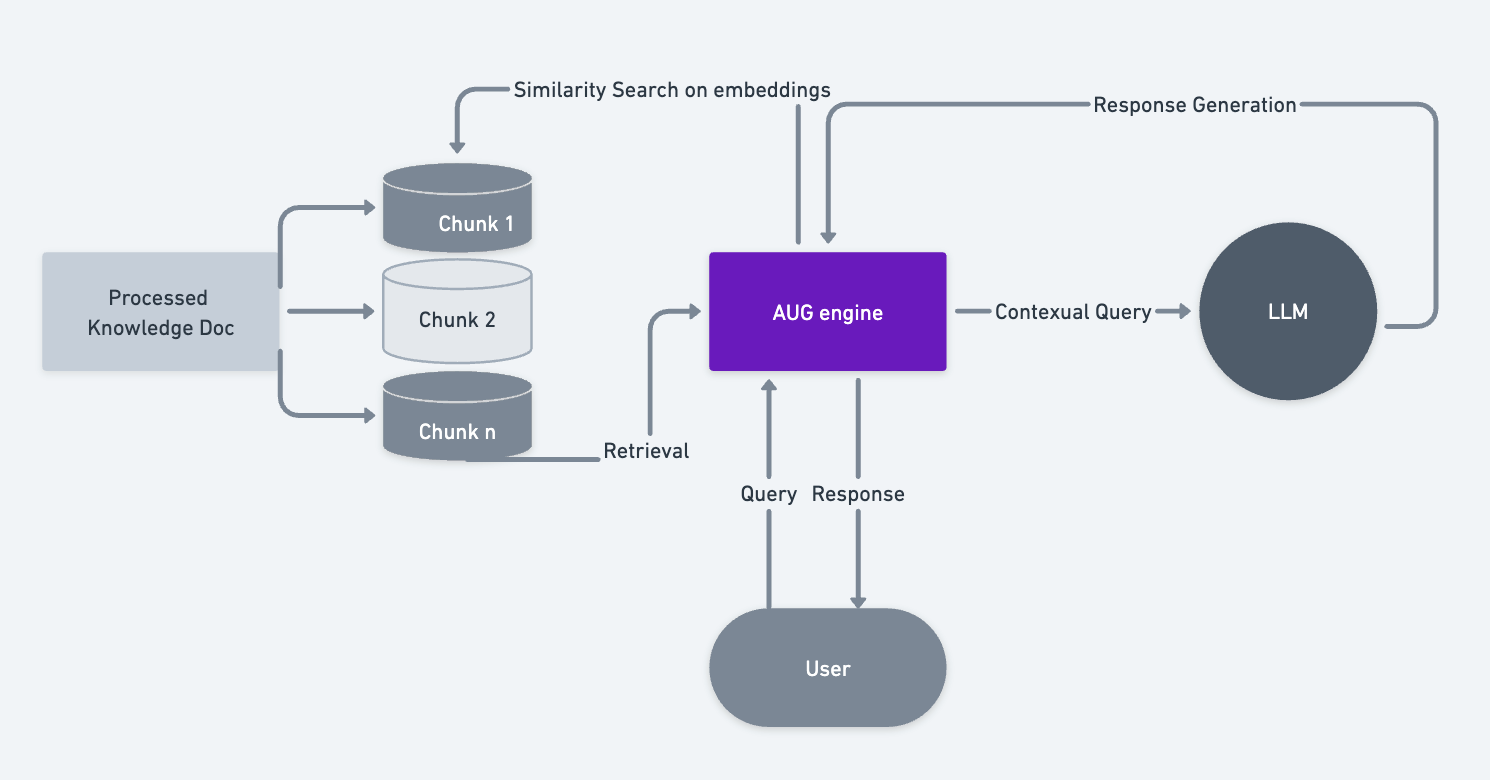
Response Generation: After getting the sources of contextual information from the knowledge, how to generate a good response, while minimizing hallucinations.

**ASSUMPTIONS:**

* **Most knowledge document structure is similar (by annotation or preprocess) as that of the provided knowledge document with some sections being able to be meaningfully separated by a delimiter(‘###’ here), as that structure is exploited for better chunking.**
* **If the structure is not similar(meaningfully separable by ‘###’), the delimeter(‘###’) is not present in multiple random positions.**
* **The size of the documents is not significantly larger than the given document (Otherwise # chunks would be very high leading to high search time and calls for vector DB).**
* Gpt 3.5 is used for generation (Different model would have different max token length)

3. Solution

Details about your solution. Illustrate performance and design with diagrams.



High Level Diagram of Process

**Process Remarks:**

* Document is parsed and preprocessed to remove noise and unwanted spaces, ‘/n, delimeters, emojis, etc.
* **Chunking Strategy:**
  + Different strategies to split the doc into different chunks was considered so as to have small enough chunks so that the answer is precise and also the scope of **hallucination is reduced in smaller chunks,**  while not losing **global context and local context.**
  + Multiple chunks were used in the context sent to LLM.
  + **Given doc has a pattern of different questions being separated by a delimiter (“###”), and also the data did not seem to have much relative context among neighboring section**. So rather than chunking on the basis of size, it is chunked on the basis of that delimiter first.
    - **If the chunk does not have that structure, the chunk is made on the basis of size** upto within MAX TOKEN SIZE = 500
  + Since some chunks can become big, big chunks were split into multiple small chunks.
  + Title is added to chunks to improve global context(Metadata should also be included)
* Chunks were embed using OpenAI embeddings
* Cosine similarity is used to find most similar chunks.
* Throttling used to get embeddings so as to not hit rate usage (Can also be done on queries).
* **Hallucination Considerations:**
  + LLM was prompted to be precise and not answer if the context is not present.
  + Chunks are short.
  + Temperature is low.

4. Future Scope

Thoughts on how you could have improved the solution.

A lot more can be done to improve the performance, not a lot could be covered in this writeup. Here are some of them:

* Different chunking strategies could be explored which also depends on type of data in terms of size of chunk, splitting techniques, global and local context, indexing etc.
* Vector DB for similarity searching would be added.
* Different techniques can be used on vectorDB like indexing, filters, metadata similarity search etc.
* **Ways to reduce Hallucinations will be considered some approaches could be comparing the results back with the ground truth and getting scores like BLEU score or other metrics to see how relevant response is.**
* **Fact Checking, Consistency checking, Ensemble Techniques can be used for Reducing Hallucinations**
* **Classification can be used on top of chunking and query to search better context.**
* **Feedback systems can be integrated (like RLHF) to improve performance.**
* **Different searching metrics can be considered.**
* Supporting multi-linguality: This can be done by using an LLM that supports multiple languages.
* Profanity checks
* Adding speech capabilities: This can be done by using a speech-to-text engine and a text-to-speech engine.

This was a very quick implementation of the bot, a lot could be thought, researched and done on the upside.

**Usage, code and links-**

* A very simple and quick app is used to deploy the script and test.
* Follow instruction on the github to run the app.
* 2 knowledge document are added in the repo to test (**‘PAN’ and ‘BERT5’**)
* The app has very basic utility as of now and the UI does not handle upload of new knowledge document from UI. Follow instructions on the github to upload new document and search from it.

Example Usage of the App:

