Brance ML Engineer Task

Name: Aman Rai
Linkedin Profile: https://www.linkedin.com/in/aman-rai-0465081b4/
Date Challenge Received: 5 July, 2023
Date Solution Delivered: 9 July, 2023

---

1. Problem Statement

Building and hosting a QnA model capable of catering 1M requests daily.

2. Approach

There are few assumptions involved with this approach -

1. The answer is present in a consecutive sentence and not split over non sequential paragraphs.
2. The user query is extensive, meaning it consists of all the required contexts, in the case of follow-up question scenarios.
   Examples for extensive query are -
   - How can I enroll for a PAN card?
   - How can I link PAN to my AADHAR?

   Examples of non extensive look like -
   - How can I enroll for it?
   - How can I link them?

3. Solution

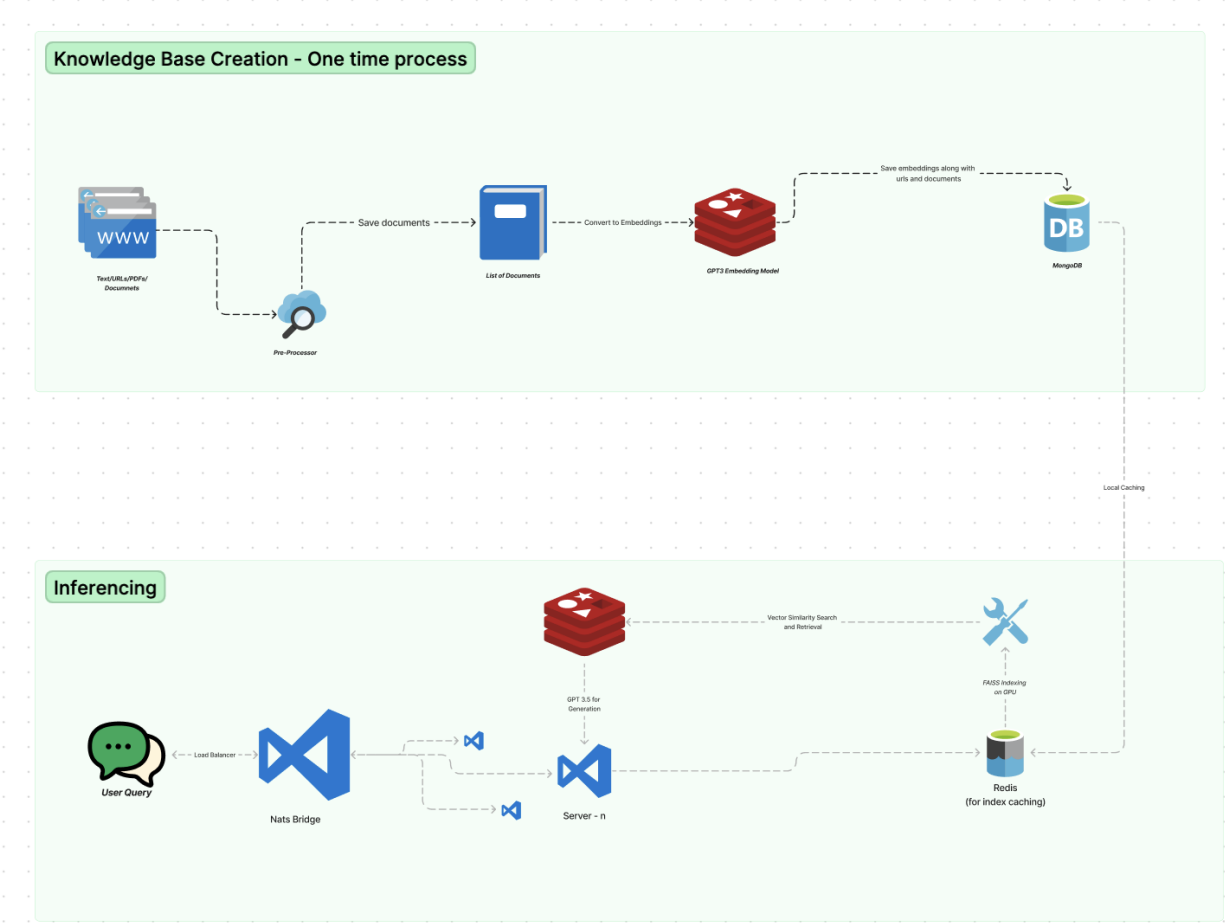
This implementation works around the simple idea of using gRPC for the client server architecture.

This works in two parts -

1. **Embedding creation -**
   This is the first step of creating the QnA model. In this stage, all the structured and unstructured documents are crawled and dumped in a text file. This file is then used to create the Knowledge Base for the chatbot. The text is converted into numbers referred to as embeddings. Here, the “ada” model from OpenAI is used to get the embeddings. These embeddings are then stored into a vector database. For this implementation, the FAISS vector database is used to store and index the mappings. FAISS, provides gpu utilization, which is essential for large matrix multiplication while calculating semantic/cosine similarity.

2. **User Query Inferencing -**

During Inferencing, the user query is converted into embeddings the same way the text is converted to embeddings in the first step and matched across the indexes. Top matching index is retrieved and sent to GPT-3.5 to generate a knowledge grounded answer based on the question.



4. Future Scope

Thoughts on how you could have improved the solution.

The improved version of this implementation would involve the following -

1. Since this project is based on a two and fro messaging idea, using NATS as an architecture instead of gRPC is ideal.
2. We can use top-k matching indexes, instead of just one to frame the answer. This improves the result when the context of the answer is split over multiple paragraphs, may or may not be together.
3. This approach creates an index and stores it locally, which is not ideal. Instead, the indexes should be stored in a globally available mongo database. These indexes can then be cached locally using Redis offering high-availability and low latency.
4. Including history and managing sessions, so users can have seamless experience.

a. Users can ask follow up question for the last one
b. Users can come back a few days later and pick-up on their last conversation.