**RAG Based Chatbot**

**ServiceTitan DS Internship 2024**

**Round2**

**Zaven Avagyan**

**6/18/2024**

**Table of Contents**

1. Description of components of the RAG System
2. Description of the challenges that might encounter
3. Examples of questions that the chatbot will or will not answer

**1.**

For our embedding tool, I would go with [**gte-Qwen2-7B-instruct**](https://huggingface.co/Alibaba-NLP/gte-Qwen2-7B-instruct), as it is has the highest max token limit (131072), it’s excellent for tasks requiring extensive context such as immense pdf files with various concepts and terminology. With the performance average score of 70.24, it comes second to [**SFR-Embedding-2\_R**](https://huggingface.co/Salesforce/SFR-Embedding-2_R) with the performance average score of 70.31 on [**Massive Text Embedding Benchmark (MTEB) Leaderboard**](https://huggingface.co/spaces/mteb/leaderboard) as of June 18, 2024. **SFR-Embedding-2\_R** is possibly the best all-rounder yet our task is specified on PDFs, that’s why my first choice would be **gte-Qwen2-7B-instruct**. It offers the highest number of max tokens for usage and considering the versatile nature of the data that is placed in PDFs, especially if not ordered properly, we will need tons of tokens to do the job and 131072 max tokens should be well enough. Moreover, with the classification average score, clustering average score and other scores presented in the benchmark leaderboard, it is greater or equal than **SFR-Embedding-2\_R**, which further strengthens our claims. As an alternative, we can take either **SFR-Embedding-2\_R** or [**NV-Embed-v1**](https://huggingface.co/nvidia/NV-Embed-v1), as they are pretty similar to each other, heavy all-rounders and are top-notched embedders. **NV-Embed-v1** also performs well across classification, pair classification, and STS, making it suitable for tasks needing strong semantic understanding. It should be mentioned, however, that these models are one of the heaviest models to use and depending on the size of the data that should be interacted with the model and on the company’s resources, we might want to change **gte-Qwen2-7B-instruct** to less effective ones, such as [**gte-large-en-v1.5**](https://huggingface.co/Alibaba-NLP/gte-large-en-v1.5) or [**gte-base-en-v1.5**](https://huggingface.co/Alibaba-NLP/gte-base-en-v1.5). Overall, General Text Embedding model family proves to be the best model family for our task. I would like to mention, that I would not use openai’s models as their best model [**text-embedding-3-large**](https://platform.openai.com/docs/guides/embeddings/embedding-models) falls back with its resuts and max tokens capacity and is not a free, open-source model, although for other models we may need to get a permission to use it in a real product, as some of them are research purpose only.

For PDF Processors, I would choose [**PyMuPDF**](https://pymupdf.readthedocs.io/en/latest/) for its multiusage and various strength. As mentioned on their website, **PyMuPDF** is a high-performance Python library for data extraction, analysis, conversion & manipulation of PDF (and other) documents. Not only it is easy to use, but also can come handy in several common yet harsh cases, such as having not clean / corrupt data. In such cases, dinamically invoking **PyMuPDF** should help one solve such problems. Although it is a great tool to use in our RAG based chatbot, there are other alternatives. [**PDFMiner**](https://pypi.org/project/pdfminer/) also offers bonanza of features and easy-to-use techniques. It handles layout information better yet lacks the speed of **PyMuPDF**. Another alternative is [**Adobe PDF Sevices**](https://developer.adobe.com/document-services/docs/overview/). Although these are various, paid services, they can offer more, since these services are not open-source and are mostly picked by companies. These models are great and hard-to-pick, but, as mentioned earlier, **PyMuPDF** can be helpful more for its immense features and mostly data extraction, cleaning and transformation. Depending on the data that the models will work with (PDF manuals), if the layout information is crucial, **PDFMiner** can perform better. **Adobe PDF Services** may outperform **PyMuPDF**, however may have usage limits or require more complex integration. Overall, **PyMuPDF** is the top pick.

As a text preprocessing model, I would consider [**NLTK**](https://www.nltk.org/) and [**SpaCy**](https://spacy.io/). According to <https://dev.to/krishnaa192/spacy-vs-nltk-12e5> , SpaCy does not include many built-in datasets and is not as comprehensive as NLTK. Considering all the pros and cons of these models, I would pick SpaCy, as the data that the model will work may seem extremely huge, it is well narrowed to certain topics and can be processed relatively easy. Another benefit is its speed and optimization. We need to have a chatbot, which should be as fast as possible while performing great results and SpaCy is the go-to model for this task.

For vector databases, after throughout research, [this](https://www.aporia.com/learn/best-vector-dbs-for-retrieval-augmented-generation-rag/) page summed up main information about top models. While having 5 top models (Milvus, Pinecone, Weaviate, Elasticsearch, and VespaMilvus), my main choice is Pinecone for its pros. Having a flexible, scalable and high performance vector db is a key to our chatbot. Also, considering the huge embedder we chose previously, being fast and scalable is the main priority. Other models can be considered as alternatives for having better scalability or for not having limitations for organizations preferring on-premise solutions, yet these models mostly suffer from immense data, have slow-growing learning curves and can decrease speed of the chatbot greatly.

For such massive information, we will need a knowledge base, based on SQL or its variations. MongoDB and PostgreSQL are the top picks for this project. MongoDB is a better pick for having a distributed database for modern transactional and analytical applications and have fast changing, multi-structured data. PostgreSQL, on the other hand, is an object-relational database management system that can be used to store data as tables with rows and columns. Knowing the nature of PDF files that our chatbot will work with, files may not be fast-changing, but the data overall is multi-structured. Also, MongoDB is widely used for such purposes and therefore can be relatively easy to implement. Considering these differences, MongoDB is the choice. However, if there are other PDFs that include huge amount of data that need to be put in tables for easy use, we can use both of these databases and combined, they will become our KB.

For search algorithms, we can either use sematic search or vector search. Vector search is a great pick if we need to have an exact information or relation between two data points, but sematic search aims to provide personalized and contextually accurate results aligned with user intent, which is the whole point of having a chatbot, that’s why sematic search is a better pick for LLMs generally, and this chatbot specifically.

Finally, to make the aforementioned aspects work together and show results, we will need a LLM and a RG (Response Generator). The best LLM models re widely known, especially GPT-4o and its predeccessors, Gemini, Orca, Llama, Vicuna, etc. My first pick would be GPT-4o for its performance and results, yet is not the fastest and is not free. If our project needs to use a free, open-source LLM, we can refer to the [**Open LLM Leaderboard**](https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard)and choose the best fitting model for our task. As of now, [**Rhea-72b-v0.5**](https://huggingface.co/davidkim205/Rhea-72b-v0.5) is the leader of this leaderboard and should be used with API as it is too large to download. For a better match, the datasets that these models were tested should be reviewed carefully and more detailed information should be retrieved, however, these models are similar and picking an open-source model that can be used in a working enterprice. We will also need to have a Query Processor and a Response Generator, but depending on the LLM we picked and the User Interface we built, NLP tools like SpaCy can be used here too. To keep the user interface and the whole project intact and reduce the risk of malfunction, we will need a proper backend infrastructure, the details of which are too vast to indulge in, in the scope of this paper.

**2.**

This project can bump into several challenges and mostly for the data it encounters or the models it uses.

One challenge can occur during the PDF data handling process. Due to its large sizes and complex nature, combined, the data retrieved from such files can mislead the model and result to a wrong answer to the user. To make it less challenging, we need to ensure that our model gets the data in a more simplified and not entangled way.

Another challenge can be the implementation of the process. Channeling and setting up a gargantuan embedder that works with a vector database and has certain search algorithms and query performances on top of that can seem a herculean task to complete and it would be a fair guess, however, every problem has its solution, we just need to find it.

**3.**

Last but not least, the 10 questions.

Five Questions the Chatbot Can Answer:

* How do I change the filter in my air conditioner?
* What are the installation steps for the kitchen sink?
* Can you explain the troubleshooting steps for my water heater?
* What maintenance tips are provided for the air conditioning unit?
* Where can I find the warranty information for the sink?

Five Questions the Chatbot May Struggle With:

* How can I integrate my air conditioner with a smart home system?

Reason: The manuals may not cover advanced integrations.

* Why is my water heater making a specific noise?

Reason: The manuals may not detail specific, unusual noises.

* Can you provide alternative installation methods for the sink?

Reason: Manuals typically provide standard procedures.

* What are the best air conditioners on the market right now?

Reason: Manuals focus on installation and maintenance, not market comparisons.

* How do I modify the sink installation to fit a custom counter?

Reason: Custom modifications may not be covered in the standard manual.