1. Problem Statement

Building a chatbot using Retrieval Augmented Generation (RAG). The RAG module should be capable of receiving user questions as input and will need to output an answer to it based on the knowledge provided in the document.

Understanding perspective its simple task, but if we want to build such service as product many aspects come in like

* Should produce human comparable results.
* Deployment should be scalable and cost effective.
* As automatic as possible, like auto-finetuning of models, auto-deployments, auto-scalability based on user load on system.

2. Approach

Assumptions:

* Embeddings model and LLM are fine tuned for the given domain knowledge, in our case PAN card related knowledge.

Git Repo link:

1. OpenAI API based solution: [**https://github.com/JagdishKolhe/qa-chat-rag**](https://github.com/JagdishKolhe/qa-chat-rag)
2. Opensource dolly-v2 and sentence transformer based: [**https://github.com/JagdishKolhe/qa-chat-rag/blob/main/experiments/RAG-open-source.ipynb**](https://github.com/JagdishKolhe/qa-chat-rag/blob/main/experiments/RAG-open-source.ipynb)

Initially I thought of using an open-source model like dolly-v2 for LLM. I implemented the code, but it was pretty slow (4 to 8 mins) for answering one question on my local machine, also dolly-v2-3b model was not giving as good results as that of OpenAI GPT model. For word embeddings I was using Hugging Face embeddings which uses Sequence Transformer models.

I switched to Google Colab to get better speed on GPU, and tried to use dolly-v2-7b and dolly-v2-12b model which has higher parameters means more knowledge has been learned by these models. But such big models were not able to load in Google Colab env hence could not test QA with these bigger models.

Finally, I stick to OpenAI API for developing this chat service. I developed simple GUI for this service using frameworks and technologies like WebSocket and Flask. I chose WebSocket because it improves network efficiency and user experience.

3. Solution

A diagram of a diagram

Description automatically generated

The flow for retrieving RAG based answer form LLM is presented in above diagram, where we store the knowledge from source documents in chunked format in vector database. While querying to LLM we wrap the relevant splits along with prompt to LLM. Prompting is mechanism through which we tell LLM what to do along with relevant context (relevant splits)

Factors affecting accuracy and efficiency.

* Is LLM trained enough to answer the domain specific answer.
* Quality of embeddings generation while storing chunks in vectordb
* The split mechanism of source knowledge data. The chunks should not lose the original context and meaning after splitting.

Metric calculation:

I used simple dot product of embeddings as a metric, higher the value better the answer.

A diagram of a question

Description automatically generated

Corner Case:

* There is a scope of improving the chunking specially for FAQ section in provided Knowledge document, after Markdown Chunking mechanism, which I used, I get all FAQ in single chunk, we can split it further to improve quality of answers.
* We should make sure how much embedding shape is supported by embedding model if it is small, we need to create smaller chunks, In our case as I used OpenAI embeddings which supports chunk size of 2048, we can go for bigger size chunks. Whereas Hugging Face Embeddings supports only 780 chunk size, so we need to further split the chunks.

4. Future Scope

Thoughts on how you could have improved the solution.

* We can play around various ways chunking is done, which might improve the answering capability.
* Fine tune existing LLM with domain knowledge, in our case PAN card related policies and articles crawled from government websites like RBI, Income Tax Department of India, Foreign Ministry, etc. This way we will be able to launch products very quickly as most of commercial LLM has given API to tune their model for specific cases with all infrastructure ready in place.
* Use well trained Domain specific Small Language Model instead of LLM, which will save inference cost in terms of time and money.
* There are various techniques by which we can evaluate the QA system like BERT\_score, perplexity, QAEval, etc. We can evaluate these scoring mechanisms.
* For matrix calculations, I am using human curated QA’s which might not be possible in all cases. So, we can use Auto generation of questions using LLM for given knowledge documents, later we can review those form human experts.

5. Concussions:

* GPT-4 tend to perform better than GPT-3.5-turbo
* Open source LLM like jolly-v2 are not prod ready, we need to work on them quite a lot before putting them in production.
* Popular Open-Source embedding models support small chunk size, we need to retrain them to support for bigger chunk size. Bigger chunk size is better for similarity search on bigger docstrings.
* There is lot to do with generic mechanism for evaluating QA system. Although domain specific metric development would be comparatively easy to achieve.