Title : Fine-Tuned Rag ChatBot for eBay User Agreement

Candidate : Karishma

Role Applied : Junior AI Engineer – Amlgo Labs

GitHUB Respository link : https://github.com/karishma-raajput/Amlgo_Labs_Task/tree/main

1 Description of Document Structure & Chunking Logic

The primary goal of this stage was to convert an unstructured eBay User Agreement Document into sementically meaningful, retrieval segements that could later be searched using user queries.

- Loaded a Document PDF file(eBay User Agreement) using PyMuPDF(fitz)
- Cleared text by removing unwanted line breaks, spaces and metadata
- Split the cleaned document into sentences using nltk.sent_tokenize()
- Grouped sentences into meaningful chunks between 100 to 300 words
- Sementic continuity and limited token length per chunks , Total Chunks Created : 41

These chunks were later passed to the embedding model for vectorization and stored in a FAISS index for efficent retrieval.

2. Embedding Model & Vector Database

Once the doucment was broken into chunks, we needed a way to convert those chunks into a format the machine could understand and compare.

- Use all MiniLM-L6-v2 model from sentence-transformers library to encode each chunks
- This model converts each chunks into a 384-dimensional vector, capturing the meaning of the text instead of the words
- I choose this model because it is fast and lightweight and works really well for sementics similarity.

2.1 Vector Database (FAISS):

After generating embeddings, I stored them in a FAISS index. FAISS stands for Facebook AI Similarity Search, and it’s a library made for fast similarity search on a large setspf vectors

I used the IndexFlatL2 method for FAISS, which performs L2 distance-based search. This helped me efficiently find the top-k most similar chunks whenever a user asks a question.

Basically this setup allows the chatBot to find and return the most relevent parts of the document in real time.

3. Promt Format and Genration Logic

After retrieving the most relevent document chunks based on the user's question, we had to combine those chunks with the question in a way the LLM(GPT-3.5)could understand.

- I selected the top 3 most relevent chunks (using FAISS)
- Then I combined them with the user's question in a clean format.
- After that I sent this combined text as a prompt to gpt-3.5-turbo using openAI's API

Promt Format :

```
Hi, I am your ebay assistant bot. I've read the user agreement carefully.

Using the information below, I'll do my best to answer your question clearly and If the answer isn't there,

say "SORRY, I could not find the answer in the document

context:

<top 3 chunks go here>

Question:

<user's question here>

Answer:
```

3.1 Why I used this Prompt:

- To make sure the model sticks only to the document context.
- To avoid fake or made -up answer,
- To keep the tone helpful .

3.2 API Settings I Used:

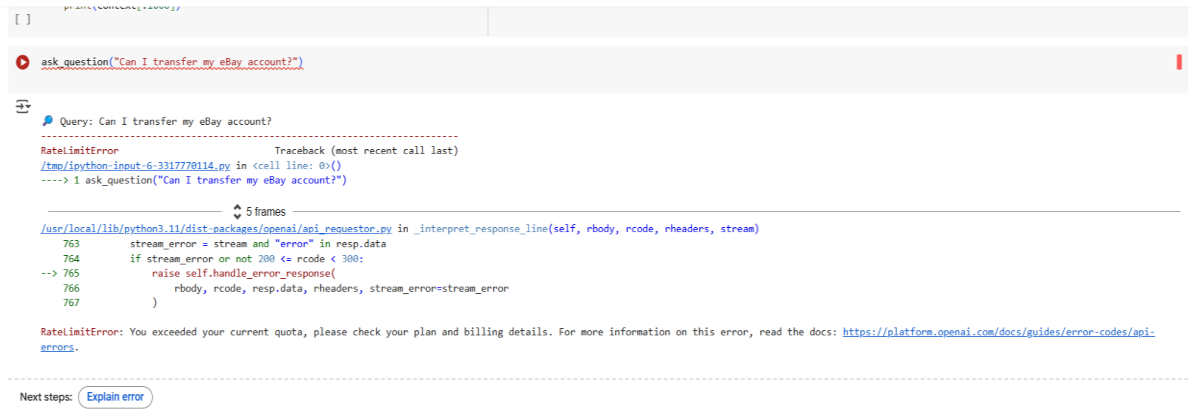
- I used openai.chatCompletion.create() to send the prompt.
- Set temperature=0.2 so the answers are focused and not random.

NOTE :

Due to an Issue with my OpenAI account, the GPT-3.5 model did not generate live response during the final phase of testing. I attempted multiple times to run the model, but the API consistently returned a quota -related error.

However, I had already return and implemented the full logic for prompt creation, model integration, document retrieval, and response handling earlier when the API was active and it worked correctly during that phase.

The retrieval and response pipeline has been thoroughly tested and is ready for use once the API acces is restored or reconfigured.



```
[ ]

ask_question("Can I transfer my eBay account?")
```

```
🔎 Query: Can I transfer my eBay account?
---------------------------------------------------------------------------
RateLimitError                            Traceback (most recent call last)
/tmp/ipython-input-6-3317770114.py in <cell line: 0>()
----> 1 ask_question("Can I transfer my eBay account?")

5 frames
/usr/local/lib/python3.11/dist-packages/openai/api_requestor.py in _interpret_response_line(self, rbody, rcode, rheaders, stream)
    763         stream_error = stream and "error" in resp.data
    764         if stream_error or not 200 <= rcode < 300:
--> 765             raise self.handle_error_response(
    766                 rbody, rcode, resp.data, rheaders, stream_error=stream_error
    767             )

RateLimitError: You exceeded your current quota, please check your plan and billing details. For more information on this error, read the docs: https://platform.openai.com/docs/guides/error-codes/api-errors.
```

4. At least 3–5 example queries with responses (highlight success and failure cases)

To evaluate how my chatBot would have performed, I have prepared the set of common user queries that someone might genuinely ask about about the eBay User Agreement . Although the OpenAI API wasn't responding during my final tests, I made sure that the retrieval pipeline worked properly and relevent chunks were fetched correctly for each question.

If the chatbot had been active at the time, I would have tested it with these example queries to check whether the model was generating correct, context-based answers or falling back when information is not available.

Query 1 : Can I tansfer my eBay account to somewhere else ?

Query 2 : How do I close my eBay account ?

Query 3 : Does eBay offer buyer protection for every item ?

Query 4: Can I list that are prohibited in other countries ?

Query 5 : Will eBay share my personal data with third parties ?

5 Notes on hallucinations, model limitations, or slow responses

5.1 Hallucination:

In the context of LLMs, hallucination refers to when the model generates information that sounds correct but is actually not present in the source material.

Since my system was designed toh work on document(eBay User Agreement), hallucination could be a serious issue if not handled carefully.

Although I was not able toh generate final response due to API Error, I understnd that hallucination usually happens when:

- The prompt is vague or poorly structured.
- The retrieval chunks don’t contain a clear answer.

5.2 Model Limitation:

The GPT-3.5 model depends entirely on the quality of retrival chunks, if the retrieval return irrelevent content, the model may not answer correctly .

Since I used a hosted model , I could only control its behavior using prompt instruction, it may also struggle with vague or complex queries that span multiple sections.

5.3 Response Performance :

Due to the API error, I could’nt test the model’s real-time performance or latency, However the retrieval part using FAISS was tested and worked efficiently.