Building a GST FAQs App

with Streamlit, Langchain, HuggingFace and VertexAI Palm APIs

Yogesh Haribhau Kulkarni (PhD)
Published in Google Cloud - Community
5 min read · 5 hours ago

Listen Share More

Photo by Danylo Suprun on Unsplash

Goods and Services Tax (GST) is a complex tax system, and it can be difficult to find answers to specific questions. This Medium Story will show you how to build a simple FAQs app with Streamlit, Langchain, HuggingFace and VertexAI Palm APIs.

Demo

Here is a sneak peak of my GST Query Bot, designed to tackle all your GST-related questions effortlessly.

Code for the app is made available at

Sarvadnya/src/ask_gst at master · yogeshhk/Sarvadnya

Contribute to yogeshhk/Sarvadnya development by creating an account on GitHub.

github.com

Code explanation

App uses `streamlit,` a Python framework used for building interactive web applications with minimal code. It allows developers to create user interfaces and handle user input easily. In this code, Streamlit is used to create the frontend of the GST FAQs web application, including the form for asking questions and displaying the answers.

```
import streamlit as st
```

Imports the `VertexAI` , `PromptTemplate` , and `LLMChain` classes from the `langchain` library. These classes are used to build question-answering systems.

VertexAI is a library within LangChain that provides language model services. In this code, it is used to create an instance of the VertexAI language model, which is used in the RetrievalQA model for answering questions.

LLMChain is a class within LangChain that represents a language model chain. It is used to create an instance of the RetrievalQA model, which combines a language model with a retrieval model for question answering.

FAISS is a library within LangChain that provides efficient similarity search and clustering of embeddings. In this code, it is used to build a retrieval model using the embeddings generated from the loaded documents.

HuggingFaceHubEmbeddings is a class within LangChain that is used to generate embeddings using pre-trained models from the Hugging Face model hub. In this code, it is used to generate embeddings for the loaded documents.

RetrievalQA is a class within LangChain that represents a retrieval-based question answering model. It combines a language model with a retrieval model to find the most relevant answer to a given question. In this code, it is used to create an instance of the RetrievalQA model using the language model from VertexAI and the retrieval model built with FAISS.

Also imports the `CSVLoader` , `UnstructuredHTMLLoader` , and `PyPDFLoader` classes from the `langchain` library. These classes are used to load data from CSV files, HTML files, and PDF files.

```
from langchain.llms import VertexAI
from langchain import PromptTemplate, LLMChain
from langchain.document_loaders.csv_loader import CSVLoader
from langchain.document_loaders import UnstructuredHTMLLoader
from langchain.document_loaders import PyPDFLoader
```

Defines a template that is used to generate answers to questions. The template includes a prompt that tells the language model that the answer should be about GST.

```
template = """
        You are a Goods and Services Tax (GST) Expert.  Give accurate answer t
        Under no circumstances do you give any answer outside of GST.

        ### QUESTION
        {question}
        ### END OF QUESTION

        Answer:
        """
```

Sets the title of the app to "GST FAQs".

```
st.title('GST FAQs')
```

Following function loads the GST FAQs data from a CSV file, a PDF file, and an HTML file. The data is then indexed using FAISS, and a RetrievalQA chain is created.

```
def build_QnA_db():
    loader = CSVLoader(file_path='./data/nlp_faq_engine_faqs.csv')
    docs = loader.load()

    # loader = PyPDFLoader("./data/Final-GST-FAQ-edition.pdf")
    # docs = loader.load_and_split()

    loader = UnstructuredHTMLLoader("data/cbic-gst_gov_in_fgaq.html")
    docs += loader.load()

    embeddings = HuggingFaceHubEmbeddings()
    db = FAISS.from_documents(docs, embeddings)
    retriver = db.as_retriever()
    llm = VertexAI()
    chain = RetrievalQA.from_chain_type(llm=llm, retriever=retriver, verbose=F
    return chain
```

The `build_QnA_db()` function is responsible for building the question and answer (QnA) database used in the GST FAQs web application. Here is a step-by-step explanation of its functioning:

1. Loading data from CSV file: The function starts by using the `CSVLoader` class from the LangChain library to load data from a CSV file. The file path is specified as `./data/nlp_faq_engine_faqs.csv`. This CSV file contains frequently asked questions (FAQs) related to GST.

2. Loading data from HTML file: Next, the function uses the `UnstructuredHTMLLoader` class from LangChain to load additional FAQs related to GST from an HTML file. The HTML file path is specified as `"data/cbic-gst_gov_in_fgaq.html"`.

3. Generating embeddings: After loading the data, the function uses the `HuggingFaceHubEmbeddings` class from LangChain to generate embeddings for the loaded documents. These embeddings capture the semantic meaning of the text and are used for similarity search in the retrieval model.

4. Building the retrieval model: The function then uses the `FAISS` class from LangChain to build a retrieval model. The retrieval model is created using the embeddings generated in the previous step. FAISS provides efficient similarity search and clustering capabilities, which are essential for retrieving the most relevant answers to user questions.

5. Creating the RetrievalQA model: Finally, the function creates an instance of the `RetrievalQA` class from LangChain. The `RetrievalQA` model combines a language model with the retrieval model built using FAISS. In this case, the language model is provided by the `VertexAI` class from LangChain. The `RetrievalQA` model is responsible for running the question-answering process by finding the most relevant answer from the QnA database based on the user's input question.

6. Returning the QnA chain: The function returns the created `RetrievalQA` model, which represents the QnA chain. This chain is then stored in the Streamlit session state for future use.

The `build_QnA_db()` function plays a crucial role in setting up the QnA database and the retrieval model used in the GST FAQs web application. It loads the FAQs from different sources, generates embeddings, and builds the retrieval model to enable accurate and relevant question answering.

Once the chain is built we can use it multiple times without repopulating it. The following function thus takes a question as input and returns an answer from the FAQs database. The answer is generated by running the question through the RetrievalQA chain.

```
def generate_response_from_db(question):
    chain = st.session_state["chain"]
    response = chain.run(question)
    st.info(response)
```

The `my_form` form allows users to enter a question about GST. When the user submits the form, the `generate_response_from_db()` function is called to generate an answer.

```
with st.form('my_form'):
    text = st.text_area('Ask Question:', '... about GST')
    submitted = st.form_submit_button('Submit')
    if submitted:
        generate_response_from_db(text)
```

To run the app, you can save the code above as a Python file and then run it from the command line. For example, if you saved the code as `gst_faqs.py`, you would run it as follows:

```
python gst_faqs.py
```

The app will then open in your browser. You can then ask questions about GST, and the app will generate answers from the FAQs database.

References

Goods & Services Tax (GST) | Home

Goods And Services Tax

www.gst.gov.in

Goods & Service Tax, CBIC, Government of India :: Frequently Asked Questions

Registration 1 Does aggregate turnover include value of inward supplies received on which RCM is payable? Refer Section…

cbic-gst.gov.in

Vertex AI Langchain Google Cloud Platform Machine Learning Generative Ai