**Tax Guidance Chatbot**

Develop a Tax Guidance chatbot using Hugging Face open-source LLMs (Large Language Model), which will give guidance and help the user to get his queries answered on Tax. Users have to upload a document which would contain all the regulations regarding the income tax and then can interact with the chatbot by asking relevant queries.

**Pre-Requisite**

The user should have Python>3.10 installed.

**Installation**

* Download the zip file and extract to the desired location.
* Open the terminal and run the following command “pip install requirements.txt”. This would install all the required libraries for the application to run.
* Then user can open that folder in any desired code editor and run the command “streamlit run app.py”. This command would run the application on your localhost.

**Methodology**

I used RAG (Retrieval Augmented Generation) architecture where knowledgebase would be our documents and the user can interact with all those tax documents. Following steps were taken to build this architecture:

1. Uploaded documents are split into chunks and after that those are embedded into high dimensional vectors. These vectors are used in retrieving only the required documents based on the user query.
2. These document vectors are stored in a vector store (FAISS in our case) and are retrieved based on similarity to the user query.
3. After user prompts with a question, five other questions are generated by LLM (in our case Mistral-7B-Instruct). This is known as multi-query retrieval which helps us to retrieve all the documents with different perspectives of same question.
4. After retrieving all the documents based on the 6 questions (1 User Question + 5 generated Questions) we do a unique union off all these documents and get the unique documents.
5. These documents act as a context to final prompt for our LLM and from there LLM generate the answer based on the question and context provided.

**Limitations**

Due to Limited time was not able to do the following things:

* Efficient Hyperparameter Tuning for Temperature, Top\_p, Top\_K etc.
* Did not use Flask web app as it would have taken time to create HTML, CSS and JS files. So, went with efficient approach of using Streamlit.
* Tailored based on profiles, since it would require login as well security.