COMPUTER HUB

Approach: -

I have implemented a Retrieval-Augmented Generation (RAG) system using a quantized LLaMA 2B model with LlamaIndex. First, I loaded the document and indexed it, converting the text into embeddings. Then, I used the quantized LLaMA 2-7B chat model to generate outputs based on these embeddings. Finally, I hosted the entire setup on Gradio to create an interactive application.

TOOLS:-

1. Llama index:- to implement RAG and get the quantized llama model
2. Sentence transformers:- To convert data to embeddings
3. Llama 2-7b:- Quantized model so that it can be run in google colab GPU
4. Gradio:- To host the site

PROBLEMS:-

1. My first approach was to use **langchain** but I could not find a way to utilize a LLM(not openai since there are no free credits left) to work on GPU T4 in colab. When I found an LLM, I could not use its quantized version.
   So, I shifted to llama index and utilized this model instead.
2. Even the quantized model is heavy and runs only once on colab before all the free credits expire so I tried to run the same cod eon Kaggle. But is crashes their at gradio level.
   I shifted back to colab
3. I can only run this model once before the free credits are expired so I had to create new google accounts in order to run them.
4. Due to this reason, I could not find the optimal number of tokens, context window and overlap to optimize the model for speed, verbosity and accuracy.

FEATURES

1. I could have used GGUF llama models. They are extremely light weight and can run on CPU and don't require GPU.
2. Counting the tokens and limiting them via langchain. Could not find this token in llama index.
3. Storing the embeddings in a database like chroma or pinecone. Fetching based on similarity index. This will improve the speed and accuracy.