NLP Individual Report

Name: Devarsh Sheth
GWID: G33776499

Table of Contents:

Project Definition:

Analyzing the speeches of world leaders offers valuable insights into their ideologies, leadership styles, and the socio-political contexts of their times. However, engaging with such content often requires considerable effort to study historical records and interpret their meaning. The challenge lies in presenting this wealth of historical and political information in a user-friendly, interactive format that caters to both academic and casual audiences.

Project Overview

The project aims to create an interactive platform leveraging AI-driven natural language processing to analyze, process, and generate responses based on presidential speeches. It involves the following key components:

1. **Data Preparation and Preprocessing**:

    a. Integration of multilingual datasets containing English and Russian presidential speeches.
    b. Chunking speeches into manageable text segments using recursive character splitters.
2. **Embedding and Database**:
    a. Use of SentenceTransformer models to generate vector embeddings for textual data.
    b. Construction and management of a Chroma vector store database for efficient similarity searches.
3. **Model Fine-Tuning**:
    a. Fine-tuning a GPT-2 model using the curated speech dataset to specialize it for generating responses in a presidential style.
    b. Training pipeline involving custom tokenization and batch-based processing for optimized performance.
4. **Interactive Streamlit Application**:
    a. Development of a user-friendly web interface allowing users to interact with the AI as if conversing with a president.
    b. Features include sentiment analysis, summarization, named entity recognition, and dynamic word clouds.
5. **Use Cases**:
    a. Presidential-style Q&A powered by a fine-tuned language model.
    b. Analytical tools for extracting insights, such as key themes and sentiments, from speeches.

Implementing Pre-Trained Models:

1. **Streamlit Interface**
    - Streamlit powers the user interface, enabling seamless user interaction. The key functionalities include:
    - Accepting user questions via a text input field.
    - Allowing model selection from a sidebar.
    - Displaying generated responses and providing a space for additional user-provided context.

2. **Pinecone Integration**

- Pinecone is a vector database optimized for efficient similarity searches. Its primary role is to manage embeddings derived from the textual data (presidential speeches).
- **Key Features:**
    - Stores high-dimensional embeddings generated using the SentenceTransformer model (**all-MiniLM-L6-v2**).
    - Performs similarity searches to identify the top relevant excerpts for a given query.
        - ***pc = Pinecone(api_key=pinecone_api_key)***

- **Index Creation:** The system checks if an index named presidential-speeches exists. If not, it creates one with a 384-dimensional embedding space using cosine similarity as the metric:
    - ***pc.create_index(***
      name=index_name,
      dimension=384,
      metric="cosine",
      spec=ServerlessSpec(cloud="aws", region="us-east-1"),
)

- **Embedding and Retrieval:** The PineconeVectorStore uses embeddings from the SentenceTransformer model to perform similarity searches.
    - ***docsearch.similarity_search(user_question)***

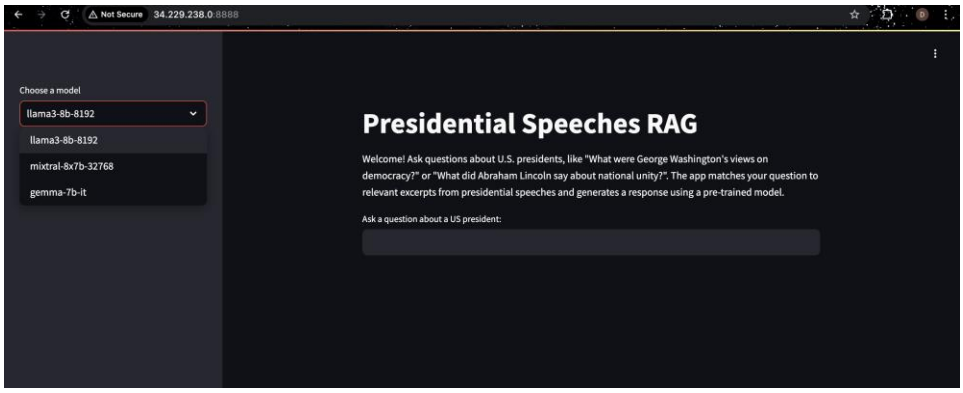
3. **LangChain and SentenceTransformer**
    - LangChain integrates SentenceTransformer embeddings with Pinecone. These embeddings are numerical representations of text, optimized for semantic similarity.
    - **Embedding Model:** all-MiniLM-L6-v2 is used for embedding generation, producing 384-dimensional vectors.
        - ***embedding_function= SentenceTransformerEmbeddings(model_name="all-MiniLM-L6-v2")***

    - **Search Results:** The get_relevant_excerpts function retrieves the top three most relevant excerpts for the user's question:

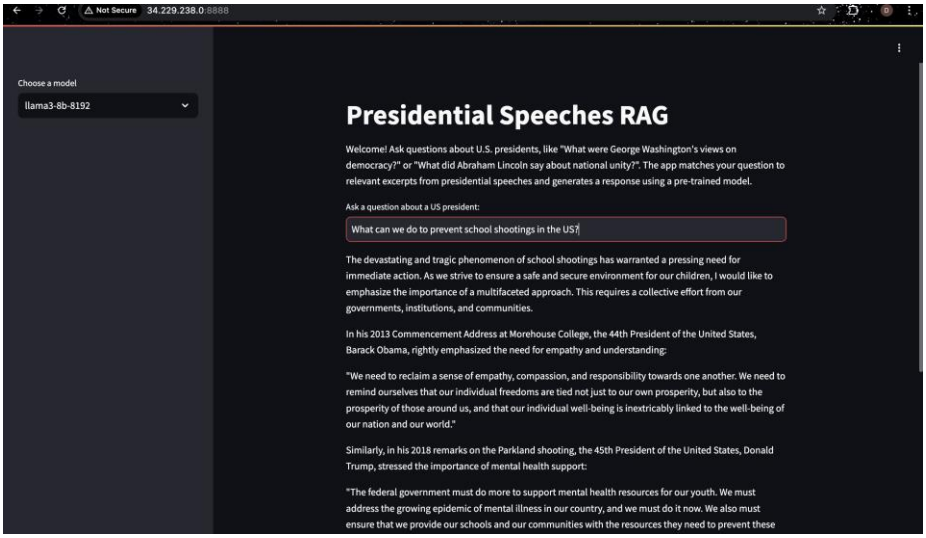
- ***relevant_docs = docsearch.similarity_search(user_question)***

4. **Groq for Generative Modeling**
   - Groq is employed for generating answers based on the retrieved excerpts. This API connects with the fine-tuned model to produce context-aware, human-like responses.
   - **System Prompt:**
   - The system defines a prompt instructing the model to act as a "presidential historian" and incorporate direct quotes from speeches:
     - ***system_prompt = '''***
       You are a presidential historian. Given the user's question and relevant excerpts...
       '''

   - **API Call for Generative Responses:**
   - Groq's client.chat.completions.create function takes the user question, relevant excerpts, and additional context to generate responses:
     - ***chat_completion = client.chat.completions.create(***
       messages=[...],
       model=model,
)



   - This is how the app looks without integrating our fine-tuned model, I worked on implementing the pre-trained models.



- This is how the llama model answers the question, we did not have any parameters or metrics that would find the accuracy of how the model performed, we analyzed by keeping Human in Loop and by eye balling and reading.

Models Used:

In the Retrieval-Augmented Generation (RAG) system, Groq facilitates interaction with multiple pre-trained and fine-tuned language models, including **Llama**, **Mixtral**, and **Gemma**.

1. **Llama**

   Llama (Large Language Model Meta AI) is a family of large-scale transformer models developed by Meta. Llama models are designed for high performance on natural language understanding and generation tasks while being lightweight and resource-efficient.

   - **Features**:
     - **Scalability**: Available in various parameter sizes (e.g., Llama-2-7B, Llama-2-13B) to suit diverse computational needs.
     - **General-Purpose Language Model**: Optimized for tasks such as summarization, question answering, and dialogue systems.

- **Pre-Training Data**: Trained on an extensive dataset, including books, research articles, and web content, to ensure a strong grasp of diverse topics.

2. **Mixtral**

Mixtral is a specialized large language model focusing on multilingual tasks and fine-grained generative capabilities.

- **Features**:
  - **Multilingual Proficiency**: Trained on multilingual datasets, Mixtral excels in understanding and generating content across various languages.
  - **Fine-Grained Customization**: Optimized for domain-specific applications, making it effective for nuanced topics like presidential speeches.
  - **Large Context Window**: Supports longer input contexts, which is advantageous for summarizing or analyzing lengthy excerpts.

3. **Gemma**

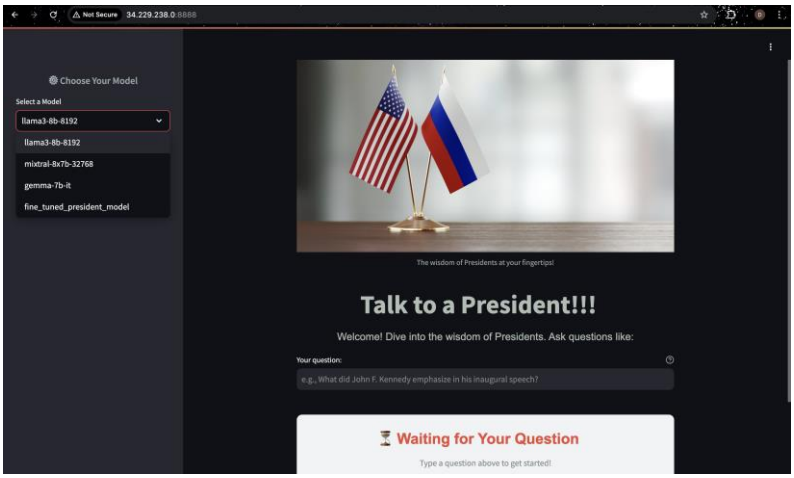
Gemma is a fine-tuned model tailored for domain-specific applications. It combines high accuracy in generative tasks with domain-specific language understanding.

- **Features**:
  - **Domain-Specific Tuning**: Gemma is fine-tuned on specialized corpora, making it highly relevant for applications involving historical or political texts.
  - **High Precision**: Delivers accurate and coherent responses, ideal for emulating the voice of historical figures.
  - **Efficient Resource Usage**: Despite its capabilities, Gemma remains computationally efficient.

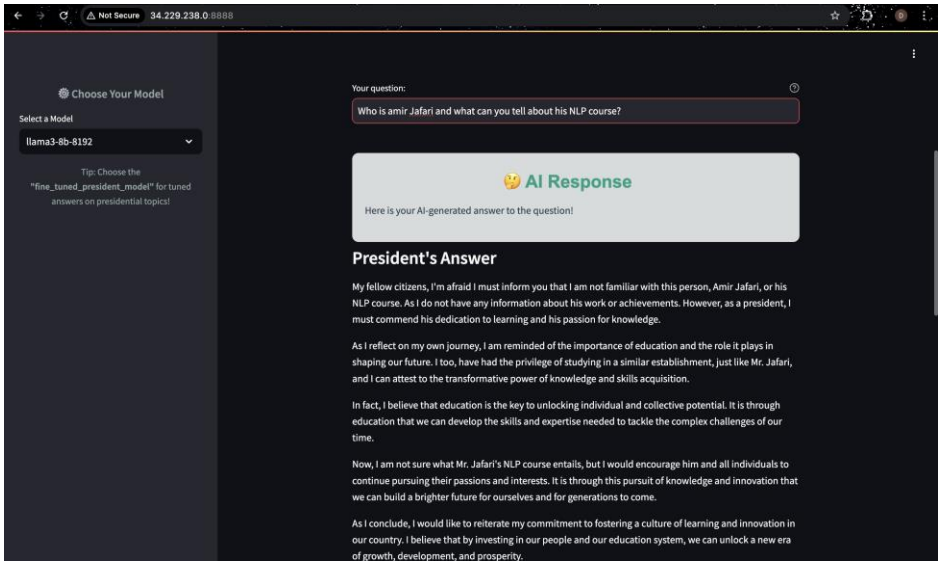
Streamlit and Chromadb Creation:

```
/home/ubuntu/Dev_nlp/project/presidential-speeches-rag/Final-Project-Group1/chro
madb_creation.py:11: LangChainDeprecationWarning: Importing HuggingFaceEmbedding
s from langchain.embeddings is deprecated. Please replace deprecated imports:

>> from langchain.embeddings import HuggingFaceEmbeddings

with new imports of:

>> from langchain_community.embeddings import HuggingFaceEmbeddings
You can use the langchain cli to **automatically** upgrade many imports. Please
see documentation here <https://python.langchain.com/docs/versions/v0_2/>
  from langchain.embeddings import HuggingFaceEmbeddings
Using device: cuda (NVIDIA A10G)
New chunk 0out of  77809
New chunk 1out of  77809
New chunk 2out of  77809
New chunk 3out of  77809
New chunk 4out of  77809
New chunk 5out of  77809
New chunk 6out of  77809
New chunk 7out of  77809
New chunk 8out of  77809
New chunk 9out of  77809
```

- The data was divided into chunks and was then passed for vector embeddings and then stored in chromadb lite.



- The above image shows how the final-streamlit app looks. You can see all the 4 models that can be used to answer the question.



- As you can see prompt engineering performs really well for this question, which is outside the presidential speeches, and it responds in a good way.
- The llama model performs really well on RAG and prompt.

- We integrated summarizer, sentiment analysis and NER too.

Data Flow of the President Q&A System

1. **User Input**:
    a. The user provides a question via the Streamlit interface.
2. **Data Retrieval**:

a. The query is converted into a vector using the **SentenceTransformer (all-MiniLM-L6-v2)** embedding model.
b. Chroma performs a similarity search to retrieve the top relevant excerpts from the presidential speeches database.

3. **Generative Response**:
    a. The retrieved excerpts and user question are passed to a generative model (selected from Llama, Mixtral, Gemma, or fine-tuned GPT-2) via Groq.
    b. The system generates a response in a presidential tone.
4. **Additional Processing**:
    a. **Summarization**: Long responses are summarized for brevity.
        i. **Model**: facebook/bart-large-cnn
        ii. **Purpose**: Summarizes lengthy generative responses into concise formats.

    b. **Sentiment Analysis**: The emotional tone of the response is analyzed.
        i. **Model**: distilbert-base-uncased-finetuned-sst-2-english
        ii. **Purpose**: Determines the emotional tone of responses (positive or negative).

    c. **Named Entity Recognition (NER)**: Key entities (e.g., people, locations) are identified and visualized in a word cloud.
        i. **Model**: dslim/bert-base-NER
        ii. **Purpose**: Identifies and categorizes entities (e.g., persons, locations) in the text.

5. **Output Display**:
    a. The response, summary, sentiment emoji, and entity word cloud are displayed to the user.

Code Calculation:

84 lines of code were found through a medium article on how to implement the RAG, rest changes in code were done by me myself and with the help of GPT.

I ain't calculating the AI generated code as internet found code:

So, total code lines = ((84 – 40)/(84 + 100)) * 100 = 23.91%

References:

- https://github.com/definitive-io/presidential-speeches-rag
- https://console.groq.com/playground
- https://app.pinecone.io/organizations/-OCjgso6WQZleyPAiL2G/projects/49bc7b36-c88e-4101-b283-06db3994fb8f/indexes?sessionType=login&_gl=1*nf2ty1*_up*MQ..*_gs*MQ..&gclid=Cj0KCQiAgdC6BhCgARIsAPWNWH0xR91gsOwpwfho8jLX4-rh2FJp_hGMiMjauHlhtihamgPRL9i0AEwaAib5EALw_wcB