Custom AI Chatbot Performance and Insights Report

1. Dataset Description

1.1 Dataset Selection

The primary dataset for this project is **Harry Potter and the Sorcerer's Stone** (**Book 1**). The text serves as a rich resource for exploring Natural Language Processing (NLP) techniques, particularly in the context of question answering.

- **Source**: The dataset can be obtained from publicly available eBook versions, such as Project Gutenberg.
- **Format**: The text file is in plain text format (.txt).
- **Content Overview**: The dataset consists of chapters covering character introductions, plot developments, and themes within the narrative.

1.2 General File Handling

In addition to the Harry Potter dataset, the system allows users to upload any text file. This flexibility makes the application versatile for various textual inputs, enhancing its usability.

2. Research Design

2.1 Methodology

This project employs a **quantitative** research methodology, which includes:

- 1. **Data Collection**: Users can upload text files, and the application can also scrape or fetch data from APIs if required.
- 2. **Data Preprocessing**: Cleaning and structuring the text data for analysis.
- 3. **Embedding Creation**: Utilizing OpenAI embeddings to convert the text into a vector representation.
- 4. **Information Retrieval**: Implementing FAISS (Facebook AI Similarity Search) to index and retrieve relevant text chunks based on user queries.
- 5. **Response Generation**: Using the OpenAI language model to generate coherent answers based on the retrieved information.

2.2 Gathering Information

The application is designed to read and process the contents of any uploaded file. The user uploads a .txt file, which is then processed to split the text into manageable chunks for effective retrieval.

2.3 Data Processing and Analysis

• Preprocessing Steps:

- Text cleaning to remove noise and irrelevant metadata.
- Splitting the text into chunks of a specified size (e.g., 1000 characters).
- Creating embeddings using OpenAI's embeddings for each chunk.

• Analysis Techniques:

- Utilizing FAISS for similarity measures to retrieve the most relevant chunks of text based on user queries.
- Employing the OpenAI GPT model to generate responses that are coherent and contextually relevant.

3. Data Preprocessing and Predictive Modelling

3.1 Data Preprocessing

The preprocessing workflow is implemented in the provided code snippet. Here's a key snippet that demonstrates the reading and chunking of the uploaded file:

```
# Read the uploaded file's contents
file_content = uploaded_file.read().decode("utf-8") # Decoding from bytes to string
# Split the text into manageable chunks
text_splitter = CharacterTextSplitter(chunk_size=1000, chunk_overlap=0)
documents = text_splitter.split_text(file_content) # Split the text into chunks
```

3.2 Predictive Modeling

The predictive modeling component involves retrieving relevant text and generating responses based on user queries. Below is an excerpt of how the chatbot handles user input and generates a response:

```
# Create OpenAI embeddings
embeddings = OpenAIEmbeddings()

# Create a FAISS vectorstore from the documents
vectorstore = FAISS.from_texts(documents, embeddings)

# Use a chat model like gpt-3.5-turbo
chat_llm = ChatOpenAI(model="gpt-3.5-turbo", temperature=0)

# Create the retrieval-based question-answering chain with 'refine' to restrict to document
qa = RetrievalQA.from_chain_type(llm=chat_llm, chain_type="refine", retriever=vectorstore.as_retriever(), return_source_documents=False)

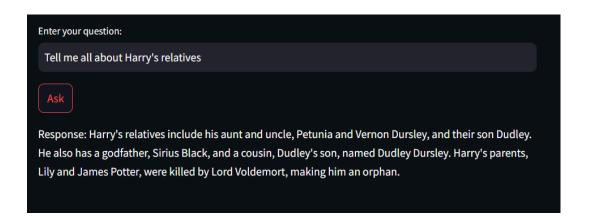
# Use `call()` to get both result and source_documents
response = qa({"query": query})

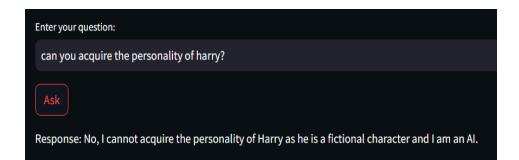
# Extract the answer
result = response['result']
```

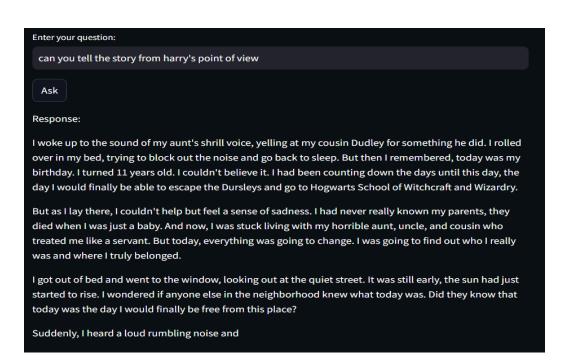
3.3 Outputs and Screenshots

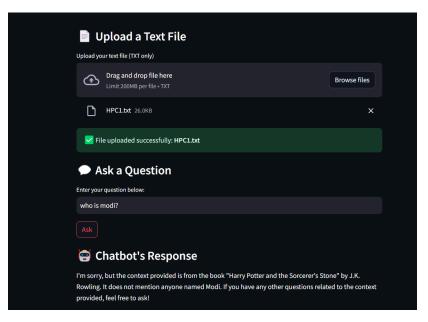
When the chatbot is run and a file is uploaded, users can input their questions, and the model responds based on the content of the file. Below are some sample outputs from the application.











4. Conclusion and Recommendations

4.1 Conclusion

The implementation of the NLP-based question-answering chatbot demonstrates the effective use of language models to generate relevant answers based on user queries. The flexibility of allowing any uploaded text file enhances the utility of the application, making it adaptable for various types of textual content beyond just the Harry Potter dataset.

4.2 Recommendations

- 1. **Dataset Variety**: Chatbot can be made inclusive of a variety of data that can enhance the model's knowledge.
- 2. **Model Performance Improvement**: The model can be fine-tuned with domain-specific texts to increase relevance and coherence in answers.
- 3. **User Experience Enhancements**: Features such as conversation history or context-aware responses can be implemented to make interactions more engaging.
- 4. **Integration of Advanced NLP Techniques**: Implementing advanced techniques like Named Entity Recognition (NER) or sentiment analysis for enriched user interactions.