JESSUP CELLARS CHATBOT

CHATBOT DEVELOPMENT PROJECT REPORT

OVERALL APPROACH

Objective:

The goal of this project was to develop a Chabot for Jessup Cellars that provides users with information about the winery, including their offerings, amenities, and other relevant details. The Chabot should handle natural language queries, maintain conversation context, and provide relevant answers based on historical interaction.

Approach:

1. **Model Selection**:
    - **Sentence Transformers**: Used for generating sentence embeddings to represent questions and answers in a high-dimensional space.
    - **SpaCy**: Used for entity recognition and coreference resolution, ensuring the Chabot can handle pronouns and references effectively.
2. **Data Handling**:
    - QA pairs were manually created from a text corpus with the assistance of ChatGPT. This approach involved generating relevant questions and corresponding answers based on the information available in the text corpus.
    - Computed embeddings for each question to facilitate similarity comparison.
3. **Conversation Management**:
    - Implemented a history management system to track user interactions and maintain context.

    - Managed follow-up queries by recognizing trigger phrases and retrieving relevant information based on conversation history.
4. **User Interface**:
    - **Streamlit**: Utilized for creating a user-friendly and visually appealing chat interface that displays conversation history and user interactions.

Frameworks/Libraries/Tools Used

Core Libraries

- **Sentence Transformers** (`Sentence Transformer`): Used to compute sentence embeddings for the Chabot's questions and user queries.
- **SpaCy** (`spacy`): Leveraged for NLP tasks such as entity recognition and coreference resolution.
- **NumPy** (`NumPy`): Employed for numerical operations, particularly for managing embeddings and computing similarities.
- **Scikit-learn** (`cosine similarity`): Utilized to measure the similarity between query embeddings and corpus embeddings.

Development Environment

- **Python**: The primary programming language used for the development of the Chabot.

User Interface

- **Streamlit** (`Streamlit`): Used to build the web-based chat interface, enabling interaction with the Chabot through a streamlined and customizable UI.

Data Handling

QA Pair Creation

- **Method**: QA pairs were manually created from a text corpus with the assistance of ChatGPT. This approach involved generating relevant questions and corresponding answers based on the information available in the text corpus.
- **Purpose**: The manually created QA pairs served as the basis for training and querying the Chabot, ensuring that it could provide accurate responses based on the pre-defined data.

Embedding Computation

- **Extraction**: Extracted questions and answers from the manually created QA pairs.
- **Computation**: Used Sentence Transformers to compute embeddings for each question. These embeddings facilitate similarity comparison between user queries and the questions in the corpus, allowing the Chabot to find the most relevant responses.

4. Problems Faced and Solutions

Installation Issues

- **Problem**: Faced difficulties installing various coreference resolution libraries, including `neuralcoref`, `coreferee`, `allennlp`, `stanfordnlp`, and `stanza`, in the Google Colab environment and Visual studio code(VSC)
- **Solution**: After encountering issues with these libraries, the decision was made to use SpaCy for Named Entity Recognition (NER) and basic coreference resolution. SpaCy was chosen due to its compatibility with the environment and its robust NLP capabilities, which allowed for effective handling of entity recognition and resolution in user queries.

Handling Coreference Resolution

- **Problem**: Difficulty in managing pronouns and references in user queries. The coreference resolution model did not fully address all pronoun resolution cases accurately, and `coreferee` was not used as originally planned.
- **Solution**: Despite trying multiple coreference resolution tools, the accuracy in managing coreference was still not fully resolved. SpaCy was employed for NER, but further improvement is needed for effective coreference resolution. Ongoing efforts to refine and enhance coreference handling are required.

Auto-Scrolling Issues

- **Problem**: JavaScript auto-scrolling functionality in the Streamlit UI did not work as intended.
- **Solution**: Attempted to integrate JavaScript for auto-scrolling, but faced implementation challenges. Alternative methods or UI improvements are being explored to ensure the chat area scrolls correctly and provides a seamless user experience.

Performance Optimization

- **Problem**: The Chabot’s performance was suboptimal, with Streamlit experiencing delays in providing bot responses.
- **Solution**: Performance optimization is still in progress. The current model's response time is longer than desired, and efforts are being made to improve the efficiency of the model and the responsiveness of the Streamlit interface.

5. Future Scope

User Experience Improvements

- **Personalization**: Implement features that allow the Chabot to remember user preferences and previous interactions for a more personalized experience.
- **Contextual Awareness**: Enhance the Chabot's ability to handle and remember context across interactions for more accurate and relevant responses.

Integration with Other Systems

- **CRM Integration**: Connect the Chabot with Customer Relationship Management systems to provide seamless user interactions and data management.
- **Analytics**: Incorporate analytics tools to track user interactions and gain insights into Chabot performance and user behaviour.

Scalability

- **Model Upgrades**: Explore more advanced models or fine-tune existing ones to improve performance and accuracy.
- **Infrastructure**: Consider scalable solutions for handling increased user traffic and ensuring consistent performance.

Web App Redesign

- **Plan**: Replace the current Streamlit interface with a minimalist web app developed using HTML, CSS, JavaScript, and Flask.
- **Purpose**: To create a more flexible and customizable user interface that aligns with specific design preferences and performance requirements. This change aims to offer a more streamlined and efficient interaction experience for users.