Wine Chatbot Project Report

1. Overall Approach

The Wine Chatbot project aimed to create a multilingual, interactive chatbot that can answer wine-related queries. The key features include:

Natural Language Processing: Utilizing a pre-trained transformer model for understanding and answering questions.

Translation Services: Supporting multiple languages to cater to a diverse user base.

Text-to-Speech: Providing audio responses for better user engagement.

Web Interface: Building a user-friendly and responsive web interface for interactions.

The overall approach involved setting up a Flask backend to handle requests, using the Transformers library for question answering, integrating Google Translate for multilingual support, and Google Text-to-Speech for audio responses. The frontend was designed using HTML, CSS, and Bootstrap for a visually appealing and functional user experience.

2. Frameworks/Libraries/Tools Used

Backend:

Flask: Used as the web framework for creating the server and handling HTTP requests.

Transformers (Hugging Face): Employed for implementing the question-answering model using pipeline, AutoTokenizer, and AutoModelForQuestionAnswering.

Googletrans: Utilized for translating user input and bot responses to support multiple languages.

gTTS (Google Text-to-Speech): Used for generating audio responses from text.

FuzzyWuzzy: Applied for fuzzy string matching to find approximate matches to user questions.

json: Used for handling the corpus of sample questions and answers.

Frontend:

HTML/CSS: For structuring and styling the chatbot interface.

Bootstrap: Used for responsive design and styling components.

JavaScript: Included for handling dynamic interactions such as sending messages and updating the chat interface.

3. Problems Faced and Solutions

Problem: Handling Multilingual Support

Issue: The chatbot needed to support multiple languages, translating both user queries and responses.

Solution: Integrated the Google Translate API using the googletrans library, allowing the chatbot to understand and respond in different languages.

Problem: Text-to-Speech Integration

Issue: Providing audio responses in various languages.

Solution: Used the gTTS library to generate audio files from text responses. Managed the storage and retrieval of these audio files dynamically.

Problem: Approximate Matching of Questions

Issue: Users might not phrase their questions exactly as in the predefined corpus.

Solution: Implemented fuzzy string matching using the FuzzyWuzzy library to find the closest match to user queries in the corpus, improving the relevance of answers.

Problem: Managing Conversation History

Issue: Maintaining context for follow-up questions.

Solution: Stored the conversation history in a list, allowing the chatbot to reference previous interactions and provide contextually appropriate responses.

Problem: Responsive Web Design

Issue: Ensuring the web interface is user-friendly and works well on different devices.

Solution: Designed the frontend using Bootstrap for responsive design, ensuring the interface adjusts smoothly to various screen sizes.

4. Project Code and Files

1. **main.py:** Contains the backend logic for handling chat requests, translation, and text-to-speech.

2. **index.html**: The frontend interface for users to interact with the chatbot.

3. **Sample Question Answers.json:** A JSON file with sample questions and answers for the chatbot to use.

How to Run the Project

Clone the repository:

IN Bash: git clone https://github.com/your_username/your_repository.git

cd your_repository

Install the required dependencies:

Bash: pip install -r requirements.txt

Run the Flask application:

Bash**: python main.py**

Open your web browser and navigate to **http://localhost:5000** to use the chatbot.

Conclusion

The Wine Chatbot project successfully demonstrates the integration of natural language processing, translation services, and text-to-speech in a single application. By addressing key challenges and utilizing powerful libraries, the project provides an interactive and multilingual user experience.