Jessup Cellars Chatbot

20th July 2024

Ahmad Meda
Final Year Student at Heriot-Watt University
BSc Computer Science

Table of Contents

Introduction

Task

In today's fast-paced world, customers are looking for quick answers and seamless experiences when visiting websites. Jessup Cellars, a renowned wine business, wants to enhance their customer service by deploying a chatbot on their website. The goal is to provide visitors with immediate and accurate information about their products, services, and other relevant details without having to navigate through multiple web pages. The chatbot should be able to:

1. Answer questions using information extracted from a provided PDF document containing details about Jessup Cellars.
2. Ensure responses are generated from the provided corpus and not from external sources.
3. Handle out-of-context questions by directing users to contact the business directly.
4. Maintain a minimalistic and user-friendly interface.

Objectives

To address the problem statement, the following objectives were set for the project:
1. **PDF Text Extraction**: - Extract meaningful text from the provided PDF document containing information about Jessup Cellars.
2. **Text Embedding and Retrieval**: - Convert the extracted text into embeddings for efficient retrieval. - Implement a retrieval mechanism to find the most relevant text chunks based on user queries.
3. **Response Generation**: - Use Groq's LLM to generate detailed and accurate responses based on the retrieved text chunks.
4**. Web Interface:** - Develop a minimalistic web interface using Flask where users can interact with the chatbot. - Display the response time for each query to provide transparency to the users.
5. **Performance Optimization**: - Ensure the chatbot provides responses within an acceptable latency (2-3 seconds). - Optimize the use of Groq's API to manage costs effectively.

Implementation

The Jessup Cellars chatbot uses Flask as the web framework to create a minimalistic user interface where users can ask questions about Jessup Cellars. The information for answering these questions is extracted from a PDF document using PyPDF2 and sentence-transformers for text embedding. FAISS is used for managing and retrieving text chunks efficiently. The chatbot uses Groq's LLM to generate responses based on the retrieved text chunks from the PDF.

Frameworks & Libraries

1. **Flask**: - Used as the web framework to create routes and serve the HTML template. - Handles HTTP requests and responses.
2. **PyPDF2**: - Used to read and extract text from the PDF document containing information about Jessup Cellars.
3. **sentence-transformers**: - Used to convert text chunks into embeddings for efficient text retrieval. - Uses the `all-MiniLM-L6-v2` model for embeddings.
4. **torch:** - Required by `sentence-transformers` for tensor operations and similarity calculations.

5. **dotenv:** - Used to load environment variables from a `.env` file, specifically for loading the Groq API key.
6. **groq**: - Used to interact with Groq's LLM API to generate responses based on the retrieved text chunks.
7. **FAISS**: - Used for managing text chunks and performing efficient similarity searches.

Problems Faced

1. **PDF Text Extraction**: - Extracting meaningful chunks of text from the PDF was initially challenging. - **Solution**: Implemented a text-splitting function that divides the text into manageable chunks for embedding.
2. **Using Chromadb for managing text chunks:** Using Chromadb was quite challenging as I kept on getting errors, Then I realized that maybe it wasn't compatible with my code. So I switched to FAISS and the code ran without errors.
3. **Response Time Measurement**: Measuring the exact response time for the Groq API call requires careful placement of timing functions.
- **Solution**: Python's `time.time()` function was used to measure the time before and after the API call.

Future Scope

1. **Enhanced NLP Capabilities:** - Integrate more advanced NLP techniques to improve the understanding and generation of responses. - Utilize pre-trained models for more accurate entity recognition and context understanding.
2. **Multilingual Support**: - Expand the chatbot's capabilities to support multiple languages, making it accessible to a wider audience.
3. **Voice Interaction**: - Implement voice recognition and synthesis to allow users to interact with the chatbot using speech.
4. **User Feedback Mechanism:** - Add a feedback mechanism to allow users to rate the responses, helping to improve the system's accuracy over time.
5. **Data Analytics Dashboard:** - Develop a dashboard for monitoring user interactions, response times, and other metrics to gain insights into user behavior and chatbot performance.
6. **Improved UI/UX**: - Enhance the user interface with better styling and responsive design to improve the overall user experience.