CUSTOMER SERVICE CHATBOT - MARR LABS

Project Overview

The Customer Service Chatbot is a prototype developed to assist executives at an Internet Service Company in managing their tasks and inquiries efficiently. This project emphasizes functionality over aesthetics, focusing on solving user problems through Natural Language Processing (NLP) and task automation. The chatbot is designed to enhance productivity by streamlining common workplace activities such as scheduling, information retrieval, and task management.

Live Demo: https://customer-service-chatbot-marr-labs.streamlit.app/

Objectives 🎯

The primary objectives of this project are:

1. Understanding User Queries: Utilize NLP capabilities to accurately interpret and categorize user inquiries into predefined categories such as scheduling, reminders, information retrieval, task management, and general assistance.
2. Implementing Core Functionalities: Develop features that enable the chatbot to perform tasks such as setting reminders, scheduling events, providing information, managing to-do lists, offering recommendations, and facilitating basic communication tasks.
3. Integration with Existing Platforms: Seamlessly integrate with tools like calendars, email, and task management applications to enhance user experience.
4. Human Escalation Mechanism: Establish a clear process for the chatbot to escalate issues to human agents when necessary, ensuring that users receive timely assistance.

Approach to the Problem

Understanding User Queries

I implemented a Natural Language Processing (NLP) system to address the requirement for understanding user queries. This system is capable of:

- Query Parsing: Utilizing NLP algorithms to parse user queries and classify them into relevant categories. For example, queries related to scheduling would be directed to the scheduling module, while questions about the weather would trigger the information retrieval module.

- Categorization: The queries are categorized into predefined labels such as:
- Scheduling
- Reminders
- Information Retrieval
- Task Management
- General Assistance

Functionality Development

To meet the functional requirements, we designed and implemented several key features:

1. User Query Understanding:

- Developed an NLP model that can recognize intent and extract entities from user input, allowing the chatbot to respond appropriately based on the query type.

2. Information Retrieval 📚:

Implemented modules to fetch:

- Weather Updates: Users can request current weather conditions based on their location using a weather API.
- News Updates: The chatbot fetches the latest headlines and summaries from various news sources.
- General Knowledge: The chatbot responds to trivia or fact-based inquiries using predefined knowledge databases.

3. Task Management ✅:

The chatbot includes robust task management capabilities, allowing users to:

- Manage to-do lists: Add, remove, or update tasks.
- Set reminders for specific tasks or events.

4. Scheduling Events 📅:

- Users can schedule events directly through the chatbot by providing details such as the event title, date, and time. While we aimed to integrate with calendar APIs, limitations in accessing the Google Developer Console prevented full implementation. Users can manually input events for now.

5. Recommendations 🍽️:

The chatbot can offer personalized recommendations based on user preferences for:

- Restaurants: Suggestions based on location and cuisine.
- Entertainment: Recommendations for movies or shows.

6. Email Communication ✉️:

- Users can send emails directly through the chatbot. They specify the recipient, subject, and body of the email, which the chatbot sends using an SMTP service.

7. Holiday Information 🎉:

- Users can check public holidays and important dates to facilitate scheduling.

8. Escalation to Human Agents 🚨:

If the chatbot cannot resolve a user's query, it can escalate the issue to a human agent. This process includes:

- Transfer of Context: The chatbot summarizes the user's query and context for the human agent.
- Seamless Transition: Users do not need to repeat themselves when transitioning from the chatbot to a human agent.

Financial Planning 💼

To further enhance the chatbot's capabilities, we integrated a Financial Planning section, which includes:

- Personalized Financial Plan: Users can input their risk tolerance, financial goals, investment preferences, goal amounts, and current savings to generate a customized financial plan.
- Tax Optimization Strategy: Recommendations on how to optimize taxes based on current savings and investment preferences.
- Financial News Summary: Users can input a URL of a financial news article to receive a concise summary.
- Sentiment Intensity Analyzer: Analyze the sentiment of a provided financial news URL, breaking down positive, neutral, negative, and compound sentiments.

Large Language Model (LLM) Integration with GPT-3.5 Turbo ⚡

The chatbot utilizes OpenAI's GPT-3.5 Turbo to enhance conversational capabilities:

- Natural Language Understanding: Advanced NLP techniques allow accurate interpretation of user intents.
- Response Generation: Produces human-like responses for natural interactions.
- Context Awareness: Retains context across conversation turns for relevant responses.
- Scalability: Handles numerous user queries simultaneously, suitable for high-demand environments.

Technical Stack 🖥️

- Frontend: Developed using Streamlit for rapid web application creation.
- Backend: Implemented in Python, leveraging libraries for NLP and API interactions.
- APIs Used:
  - OpenAI API: For NLP capabilities.
  - Weather API: To retrieve current weather information.
  - News API: For fetching the latest news articles.
  - Yelp API: For restaurant recommendations.
  - SMTP Service: For sending emails.
  - Calendarific API: To retrieve information about public holidays.

Key Performance Indicators (KPIs) 📊

To evaluate the chatbot's effectiveness, we propose tracking the following KPIs:

1. Total Interactions: Total number of interactions logged.

2. Active Users: Number of unique users interacting with the chatbot.

3. Average Satisfaction Score: Average user satisfaction score collected from feedback.

4. Average First Response Time: Average time taken for the chatbot to respond.

5. Resolution Rate: Percentage of user queries successfully resolved by the chatbot.

Installation and Setup ⚙️

To run the chatbot locally, follow these steps:

1. Clone the Repository:

   ```bash
   git clone https://github.com/Ashishlathkar77/Customer-Service-Chatbot.git
   cd Customer-Service-Chatbot
   ```

2. Install Required Packages:

   Ensure you have Python installed, then run:

   ```bash
   pip install -r requirements.txt
   ```

3. Run the Application:

   Execute the following command to start the chatbot locally:

   ```bash
   streamlit run app.py
   ```

Conclusion

The Customer Service Chatbot project for Marr Labs was a focused effort to develop a functional prototype within a limited timeframe. Despite challenges, such as API integration limitations, the core functionalities were successfully implemented. The chatbot serves as a valuable tool for executives, streamlining their tasks and inquiries, and stands to improve further with additional features and refinements.