Business AI Agent

Project Overview

The Business AI Agent is a conversational AI system designed to automate and enhance the customer feedback experience. Its primary purpose is to process both audio and text-based feedback, analyze the content using a powerful Llama 3 large language model, retrieve relevant product information if needed, and respond with synthesized speech. The agent supports seamless switching between voice and text input, making it user-friendly and accessible. Key functionalities include audio recording and transcription, intelligent text analysis, mobile product search integration, and dynamic text-to-speech response generation. This end-to-end feedback loop helps businesses better understand customer sentiments and provide prompt, intelligent responses without manual intervention.

Project Structure

The project is organized into multiple modules for clarity and modularity. Each file plays a distinct role in the system's workflow.

src/main.py
This is the main entry point of the application. It manages command-line arguments, chooses between audio or text input modes, and coordinates the end-to-end feedback cycle from recording to response.

src/config.py
Contains configuration parameters including audio settings, model names, and API keys. Modifying this file allows easy customization of how the agent behaves.

prompts/llm_analyzer.py
Handles interaction with the Llama 3 model. It sends the customer query to the model, optionally calls tools like product search, and processes the model's response.

search/data_searcher.py
Implements the logic for mobile product search. It parses the query, reads from a dataset, and returns relevant product results to the LLM for use in its response.

speech_to_text/audio_recorder.py
Manages microphone input and records audio. It handles the technical details of capturing audio and saving it to a temporary file.

speech_to_text/stt_transcriber.py
Transcribes the recorded audio into text using the Google Web Speech API. This transcription is passed along for analysis by the language model.

text_to_speech/tts_speaker.py
Converts the text output from the LLM into speech using a text-to-speech engine and plays the response aloud to the user.

Key Functions Explained

run_automation (in src/main.py)
This function controls the entire flow of the application. It determines the input mode, triggers audio recording or prompts for text, handles the transcription or text input, sends the input for LLM analysis, and finally speaks the response.

record_audio (in speech_to_text/audio_recorder.py)
This function captures audio from the user's microphone for a predefined duration and saves it as a WAV file. It ensures that audio input is available for transcription.

transcribe_audio (in speech_to_text/stt_transcriber.py)
This function takes the recorded audio and transcribes it into plain text using the Google Web Speech API. The output is the textual version of the customer's voice feedback.

analyze_with_llama3 (in prompts/llm_analyzer.py)
This function sends the customer query to the Llama 3 model hosted on the Together AI platform. It receives intelligent insights, summaries, and recommendations from the model based on the input.

search_mobiles (in search/data_searcher.py)
This function is used by the LLM to search a local dataset of mobile phones. It filters results based on parameters like price, storage, and brand to find phones that match customer needs.

text_to_speech_and_play (in text_to_speech/tts_speaker.py)
This function converts the model's text response into speech and plays it for the user. It allows the assistant to speak back using natural-sounding voices.

call_together_api (in prompts/llm_analyzer.py)
This function interacts with the Together AI API, sending the prompt and tools to the Llama 3 model and receiving structured results. It ensures secure and consistent communication with the LLM service.

Dependencies

The following Python libraries are required for this project:

sounddevice
soundfile
numpy
speech_recognition
pyttsx3
requests
pandas
python-dotenv

Setup Instructions

1. Clone the repository
   git clone https://github.com/Sohammhatre10/business_ai_agent.git
   cd business_ai_agent

2. Create and activate a virtual environment
   python -m venv .venv

On Windows: ./.venv/Scripts/activate
On macOS/Linux/bash: source .venv/Scripts/activate

3. Install dependencies
pip install -r requirements.txt

4. Set the TOGETHER_API_KEY environment variable
Create a .env file in the root directory
TOGETHER_API_KEY="your_actual_together_api_key"

Usage Instructions

To run the application, use the main script in one of the supported modes.

1. Running in record mode (default)
python -m src.main.py --mode record
This will begin recording audio, process it through the pipeline, and respond audibly. Press Ctrl+C to stop.

2. Running in query mode
python -m src.main.py --mode query
This will prompt the user to type in feedback or a query. After analyzing and responding, it will prompt again. Type exit or quit to stop.

3. Listing available TTS voices
Inside tts_speaker.py, modify the script to print out available voice options using pyttsx3's getProperty method.

4. Using male or female voices
Edit text_to_speech_and_play to select a voice ID corresponding to either male or female based on preference.

Conclusion

This Business AI Agent streamlines the process of receiving, analyzing, and responding to customer feedback in real-time. By combining speech recognition, large language models, product search, and text-to-speech synthesis, it provides a hands-free, intelligent interaction experience. With flexible modes, customizable configurations, and robust integration with Llama 3 via Together AI, the system demonstrates how AI can meaningfully improve customer engagement workflows.

Submitted to Apprisity Technologies by Soham Motiram Mhatre