Project Documentation - GenAI for Audio

Motivation

In the fast-paced world of customer interactions, particularly in call centers, there is a need for an automated system that can efficiently handle real-time conversations, extract insights, and provide accurate responses. This project aims to implement a voice-based chatbot that leverages advanced AI models like OpenAI's GPT and Whisper for natural language processing and generation.

The goal is to build a conversational bot that accepts audio inputs, processes them, and returns audio responses in real time, thereby improving the call center experience and enhancing customer satisfaction.

Objectives

1. **Voice-activated Bot:** Create a chatbot that can handle natural language conversations through audio inputs and outputs, providing a seamless user experience.
2. **Automated Insights**: Use advanced speech processing techniques to extract insights from conversations, add history and generate context-aware responses using Whisper and OpenAI models.
3. **Real-time Processing**: Implement a pipeline that processes audio inputs, generates responses using the LLM (Large Language Model), and outputs the response in audio format.

How is Today’s Content Relevant to my Role?

- **Product Managers (PMs)**: Gain valuable insights from real-time conversations to shape product strategies without needing to manually review each call.
- **Technical Program Managers (TPMs)**: Automate repetitive tasks and optimize workflows in call centers by utilizing this conversational bot for enhanced lead engagement.
- **Software Development Engineers (SDEs)**: Develop and integrate Whisper and OpenAI models for processing natural language inputs and outputs within call center environments.
- **Engineering Managers**: Oversee the implementation of AI-driven systems for improved team performance and customer interaction.
- **DevOps Engineers**: Build and maintain the necessary infrastructure for scalable and reliable real-time audio processing.

Dataset

- https://drive.google.com/drive/folders/11ZjKMhkCeLpjw-zk9eQw4ry69cLE9lpy?usp=drive_link

Prerequisites

- **Python** - Basic programming language for development.
- **Install Visual Studio Code (VS Code)**:
    - Download and install **VS Code** from here.
- **Langchain** - https://python.langchain.com/v0.2/docs/introduction/
    - for building prompts and orchestrating calls to LLMs
- **Streamlit -** https://docs.streamlit.io/develop/tutorials
    - or building interactive chat console

Get Ready for the Session

We will use **Poetry**, a versatile tool for Python projects that simplifies dependency management and packaging. It automates the process of declaring, installing, and updating the libraries your project relies on. By using a lockfile, Poetry guarantees consistent and reproducible builds, ensuring that your project's dependencies are always installed in the specified versions.

Additionally, Poetry provides convenient features for building your project into distributable formats, making it easier to share and deploy your work.

Steps Involved

1. **Install Python**
    - Download and install from Python.org.
    - Versions 3.10.12 and 3.12.6 are recommended.

2. **Install Poetry**
    - Visit Poetry documentation for installation instructions.

3. **Install FFmpeg**
    - **Windows**
        - `ffmpeg.exe` and `ffprobe.exe` are included in the `libs` directory.
    - **Mac:**
        - **MacOS Monterey v12 and below:** Use MacPorts.
            - Install Apple's CLI Developer Tools: `xcode-select --install`.
            - Download and install MacPorts.

    - **MacOS Ventura and above:** Use Homebrew.
      - Update Homebrew: `brew update`.
      - Upgrade formulae: `brew upgrade`.
      - Install FFmpeg: `brew install ffmpeg`.
  - **Linux:**
    - Install with: `apt install libasound2-dev portaudio19-dev libportaudio2 libportaudiocpp0 ffmpeg`.

4. **Create a Virtual Environment**
   - Windows:

```
python -m venv .venv
source .venv/Scripts/activate
```

   - Linux:

```
python3 -m venv .venv
source ./.venv/bin/activate
```

   - Mac
     - Install PyAudio prerequisite:

```
brew install portaudio
```

5. Install the Code

   - Navigate to the root directory with the `pyproject.toml` file and run:

```
poetry lock
poetry install
```

6. Install Playwright

- For web testing automation:

  ```
  playwright install
  ```

7. Test Environment

- Ensure the virtual environment is activated:

  ```
  python -V
  ```

8. Create an OpenAI Account and Obtain a Key

- Follow instructions here to create your key.
- Copy `.env_template` to `.env` and add your key:
  - OPENAI_API_KEY="<YOUR_KEY_GOES_HERE>"

Launch Notebook

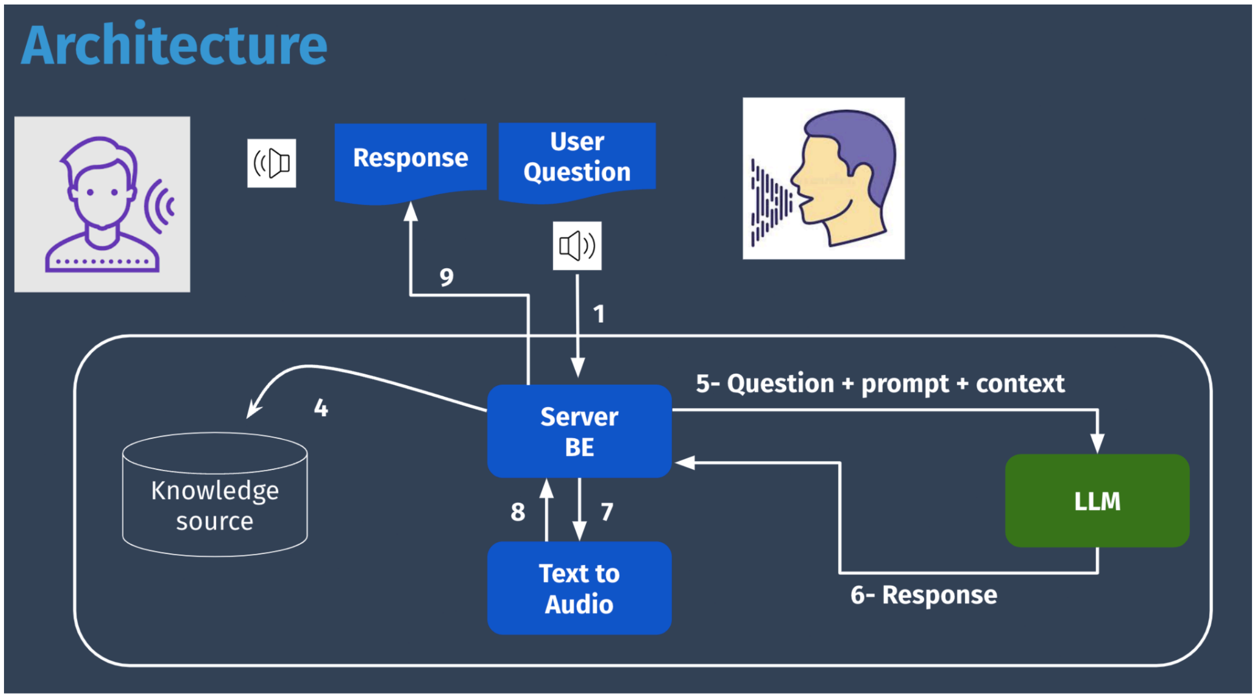
- Run the following to launch [assistant.ipynb] in a browser:

  ```
  jupyter notebook assistant.ipynb
  ```

Troubleshooting

1. Try using a headset microphone.
2. Ensure you're in a quiet room.

Implementation Steps



1. Audio Input Processing

- Use **OpenAI Whisper** for automatic speech recognition to convert the audio input into text.
- Implement **Voice Activity Detection (VAD)** to segment the audio into speech and non-speech sections.

2. Text Processing and Contextual Understanding

- Extract relevant features from the audio.
- Convert audio segments into text and feed them into the **OpenAI GPT model**.
- Use the **LangChain toolkit** to build prompt templates and generate responses.

3. Real-time Conversational Response

- Utilize OpenAI GPT to generate context-aware responses based on the input text.

- Integrate a text-to-speech module (e.g., **Pyttsx3**) to convert the textual responses back into audio.

4. Application and UI Development

- Use **Streamlit** or **Gradio** to build an interactive application where users can speak and receive responses.
- Implement conversation memory using LangChain to maintain context across multiple user interactions.

5. Integration and Data Handling

- Build a data pipeline using **FFmpeg** to handle and process incoming audio files.

Milestones

1. **Data Pipeline and Integration**:
    - Build a pipeline that processes and organizes audio data for analysis.
2. **Use AI models for speech recognition and NLU**:
    - Develop and fine-tune AI models for speech recognition and NLU.
    - Implement real-time processing capabilities.
3. **Real-Time Engagement**:
    - Deploy the call center bot and integrate it with live customer interactions.
    - Ensure the bot provides accurate, context-aware responses.
4. **Streamlit/ Gradio Interface for Real-Time Interaction:**
    - Set Up a simple UI Interface using Streamlit or Gradio to have real-time interaction with the Chatbot.

Future Directions

- **Multilingual Support**: Expand the bot’s capabilities to handle conversations in multiple languages.
- **Advanced Sentiment Analysis**: Implement sentiment analysis to better understand customer emotions during conversations.
- **Scalable Deployment**: Explore connecting with other databases like MongoDB and using cloud services for improved scalability.

Common FAQs

1. **What type of input does the system accept?**
    - The system processes real-time audio inputs from call center interactions.
2. **How does the chatbot generate responses?**
    - The bot converts audio input into text using Whisper, processes the text using OpenAI's GPT, and converts the response back into audio.
3. **What technologies are used in this project?**
    - The project utilizes Python, Whisper, OpenAI GPT,FFmpeg, Pyttsx3, and Streamlit for building the chatbot.
4. **How can this system be integrated with existing call center operations?**
    - The system can be integrated with CRM systems for managing customer interactions, providing contextual responses, and enhancing the overall customer experience.