**End-to-End AI Voice Assistant Documentation**

**Overview**

This project implements an end-to-end AI voice assistant capable of converting spoken audio into text, generating a response using a Large Language Model (LLM), and converting that text back into speech. The workflow is divided into three main steps: Voice-to-Text Conversion, Text Input into an LLM, and Text-to-Speech Conversion.

**1. Voice-to-Text Conversion using Whisper**

* **Libraries Used:**
  + **wave**: Used to read and process WAV audio files.
  + **webrtcvad**: Implements Voice Activity Detection (VAD) to filter out non-speech portions of the audio.
  + **pydub**: For audio file conversion and manipulation, allowing the transformation of formats like M4A or MP3 to WAV.
  + **faster-whisper**: A more efficient implementation of the Whisper model for transcription.
* **Implementation:**
  + **Audio Conversion**: The function convert\_audio\_to\_wav() converts M4A or MP3 files to a WAV format, setting the sample rate to 32 kHz and using a mono channel to ensure compatibility with Whisper.
  + **Voice Activity Detection (VAD)**: The vad\_filter() function filters out non-speech frames using a WebRTC VAD instance. This helps in reducing the processing load and improving transcription accuracy.
  + **Transcription**: The transcribe\_audio() function loads the filtered audio and uses the Whisper model to transcribe it into text.
* **Advantages:**
  + **Efficiency**: faster-whisper is faster and more resource-efficient than the original Whisper model, making it suitable for real-time applications.
  + **Accuracy**: VAD helps in reducing noise and non-speech parts, improving transcription quality.

**2. Text Input into LLM**

* **Libraries Used:**
  + **torch**: Provides support for tensor computation and GPU acceleration.
  + **transformers**: Used to load and interact with pre-trained language models like LLaMA.
* **Implementation:**
  + - **Loading the Model**: The LLaMA model is loaded using AutoTokenizer and AutoModelForCausalLM from Hugging Face's transformers library.
    - **Tokenization and Inference**: The transcribed text is tokenized and passed to the LLaMA model to generate a response. The generation process uses parameters like top\_k, top\_p, and temperature to control the randomness and relevance of the output.
    - **Response Processing**: The generated response is cleaned to remove redundant information and provide a concise output.
* **Advantages:**
  + **Customizability**: LLaMA allows fine-tuning of generation parameters to control the style and creativity of the output.
  + **Scalability**: The model can be run on GPUs for faster inference, making it suitable for real-time applications.

**3. Text-to-Speech Conversion**

* **Libraries Used:**
  + **parler-tts**: A library for generating speech from text using the Parler TTS model.
  + **soundfile**: For handling audio file I/O operations.
* **Implementation:**
  + - **Limiting Sentences**: The limit\_sentences() function limits the number of sentences in the generated response to ensure concise output.
    - **VAD Application**: The apply\_vad() function applies a VAD threshold to remove low-energy segments, ensuring clarity in the generated speech.
    - **Text-to-Speech Conversion**: The text\_to\_speech() function uses the Parler TTS model to convert the LLM-generated text into speech. Parameters like pitch, speed, and gender are adjustable to customize the output.
* **Advantages:**
  + **Flexibility**: The Parler TTS model allows customization of speech characteristics like pitch, gender, and speed, making it adaptable to different user preferences.
  + **Clarity**: The use of VAD ensures that only the relevant speech segments are synthesized, improving clarity.

**Models Used**

**1. Whisper (faster-whisper)**

A model for converting speech to text, optimized for speed and efficiency compared to the original Whisper model.

**2. LLaMA (open\_llama\_3b)**

A language model that generates contextually relevant responses based on the input text. It's designed for tasks requiring natural language understanding and generation.

**3. Parler TTS**

A text-to-speech model that generates high-quality speech from text input, with customizable parameters for pitch, speed, and gender.

**Conclusion**

This AI voice assistant pipeline integrates cutting-edge models for voice transcription, language understanding, and speech synthesis, providing an efficient and flexible solution for voice-based applications. The use of VAD, model optimization, and customizable parameters ensures high-quality output tailored to various use cases.

**Code and Demo Video**

* [Code Repository](https://github.com/coderishabh11/AI-Voice-Assistant-Pipeline/tree/main)
* [Demo Video](https://drive.google.com/drive/folders/1ThX67QT1_N8zMjDBOcZqvBT5nrvV8WDs?usp=sharing)