Detailed Explanation

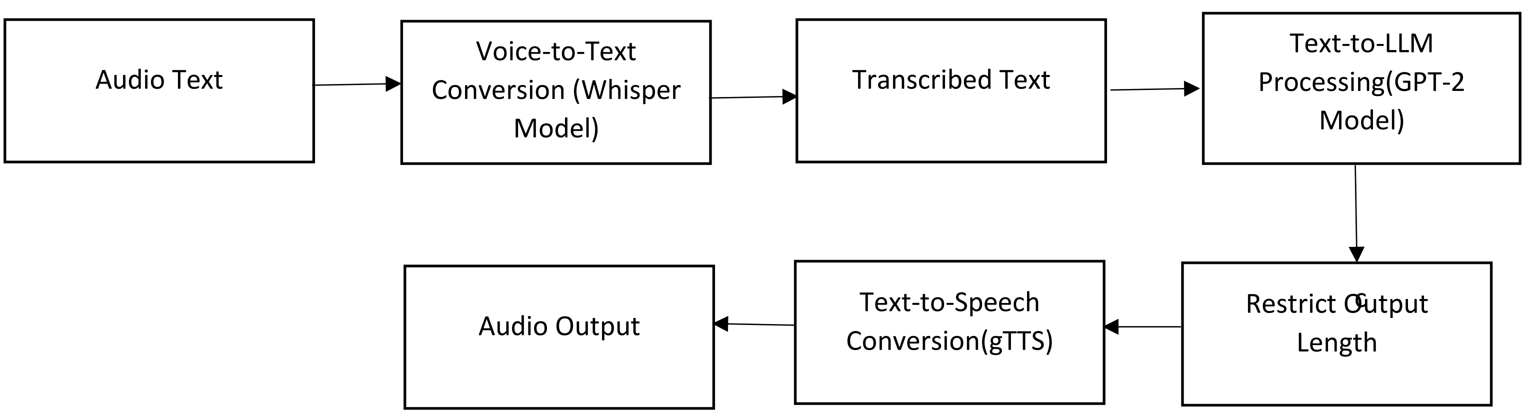
This project implements an end-to-end AI voice assistance pipeline, which processes audio input to generate meaningful text responses and converts those responses back into speech. The pipeline is broken down into the following steps:

1. **Voice-to-Text Conversion**:
   - **Library**: whisper (OpenAI)
   - **Model**: base
   - **Description**: The Whisper model is employed to transcribe the audio input into text. This model is selected for its robustness in handling various accents and background noise, making it ideal for voice-driven applications.
   - **Hyperparameters**:
     - **Language**: English (language="en")
     - **VAD Threshold**: Set to 0.5 for better voice activity detection, reducing background noise impact.
2. **Text Input into Language Model (LLM)**:
   - **Library**: transformers (Hugging Face)
   - **Model**: GPT-2
   - **Description**: The transcribed text is processed by a pre-trained language model, GPT-2, to generate a coherent and contextually relevant response. GPT-2 is chosen for its balance between performance and computational efficiency.
   - **Hyperparameters**:
     - **Max Length**: 50 tokens, to limit the response length and maintain relevance.
     - **Num Return Sequences**: 1, to ensure only one response is generated.
3. **Restricting Output Length**:
   - **Custom Function**: restrict_output_length
   - **Description**: This function limits the LLM's output to a maximum of two sentences, ensuring concise responses. It improves user experience by avoiding overly verbose answers.
4. **Text-to-Speech Conversion**:

- **Library**: gTTS (Google Text-to-Speech)
- **Description**: Converts the generated text response into speech, saved as an audio file. gTTS is selected for its simplicity and integration with Python, though alternatives like edge_tts can be used for more natural voice synthesis.
- **Output**: The audio file is saved in MP3 format for easy playback on various devices.

Diagram

The following flowchart visually represents the data flow through the pipeline:



Code

AI Voice ASSISTANCE.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Files: sample_data, WhatsApp Audio 2024-08-26 at 3..., output.mp3

Disk 73.67 GB available

[4]
```python
from transformers import pipeline
import asyncio
import webrtcvad
import wave
from gtts import gTTS
import whisper
from edge_tts import Communicate
```

[14]
```python
def voice_to_text(audio_file_path):
    model = whisper.load_model("base")
    transcription = model.transcribe(audio_file_path)
    text = transcription["text"]
    return text
```

[6]
```python
# Step 2: Text Input into LLM
def text_to_llm_response(text):
    llm = pipeline("text-generation", model="gpt2")
    response = llm(text, max_length=50, num_return_sequences=1)[0]['generated_text']
    return response
```

[46]
```python
# Text-to-Speech Conversion with Tunable Parameters
async def text_to_speech_with_params(text, output_path, voice="en-US-JennyNeural",rate="+10%"):
    communicate = Communicate(text=text, voice=voice, rate=rate)
    await communicate.save(output_path)
```

[8]
```python
# Apply VAD to Detect Speech
def apply_vad(audio_file_path):
```

AI Voice ASSISTANCE.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Files: sample_data, WhatsApp Audio 2024-08-26 at 3..., output.mp3

[46]
```python
# Text-to-Speech Conversion with Tunable Parameters
async def text_to_speech_with_params(text, output_path, voice="en-US-JennyNeural",rate="+10%"):
    communicate = Communicate(text=text, voice=voice, rate=rate)
    await communicate.save(output_path)
```


[8]
```python
# Apply VAD to Detect Speech
def apply_vad(audio_file_path):
    vad = webrtcvad.Vad()
    vad.set_mode(1)  # Aggressiveness mode
    with wave.open(audio_file_path, 'rb') as wf:
        frames = wf.readframes(wf.getnframes())
        sample_rate = wf.getframerate()
        frame_duration = 20  # ms
        for i in range(0, len(frames), int(sample_rate * frame_duration / 1000)):
            frame = frames[i:i + int(sample_rate * frame_duration / 1000)]
            if vad.is_speech(frame, sample_rate):
                yield frame
```

[9]
```python
# Additional Features: Restrict Output Length
def restrict_output_length(text, max_sentences=2):
    sentences = text.split('. ')
    return '. '.join(sentences[:max_sentences]) + '.'
```

```python
# Example usage
text = voice_to_text('/content/temp_audio (1).wav')
print(text)

response = text_to_llm_response(text)
response = restrict_output_length(response)
```

Disk 73.67 GB available

AI Voice ASSISTANCE.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

Files

..
sample_data
WhatsApp Audio 2024-08-26 at 3...
output.mp3

Disk 73.67 GB available

+ Code + Text

RAM
Disk

Gemini

```
# Example usage
text = voice_to_text('/content/temp_audio (1).wav')
print(text)

response = text_to_llm_response(text)
response = restrict_output_length(response)
print(response)

import nest_asyncio
import asyncio

# Patch the running event loop
nest_asyncio.apply()

# Use await if you're in a Jupyter notebook or similar environment
await text_to_speech_with_params(response, "output.mp3", voice='en-GB-RyanNeural', rate="+10%")
```

```
/usr/local/lib/python3.10/dist-packages/whisper/__init__.py:146: FutureWarning: You are using `torch.load` with `weights_only=False` (the
  checkpoint = torch.load(fp, map_location=device)
/usr/local/lib/python3.10/dist-packages/whisper/transcribe.py:115: UserWarning: FP16 is not supported on CPU; using FP32 instead
  warnings.warn("FP16 is not supported on CPU; using FP32 instead")
 Hi, I am Harshel How are you? What are you doing? Ok, bye
Truncation was not explicitly activated but `max_length` is provided a specific value, please use `truncation=True` to explicitly truncate
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
 Hi, I am Harshel How are you? What are you doing? Ok, bye.

I'm going to do a follow up to this story that I had previously posted as well as how I think it would look in that scenario.
```

[ ] Start coding or generate with AI.