

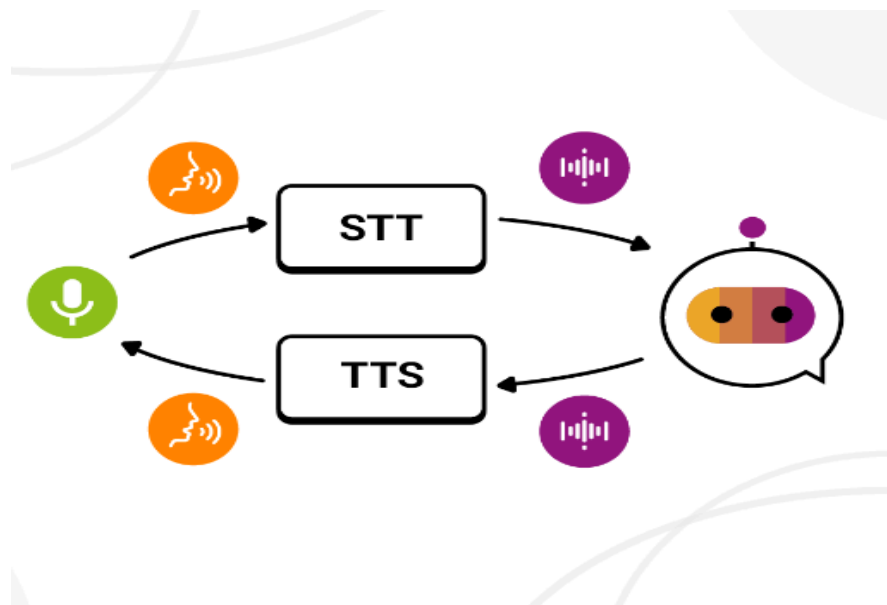
End-to-End AI Voice Assistant Pipeline Using Hugging Face

Assignment - Lizmotors Mobility Pvt Ltd.

DETAILED REPORT

1. Introduction

This report details the implementation of an end-to-end AI voice assistant pipeline using Hugging Face models. The project involves speech-to-text (STT), text processing with a large language model (LLM), and text-to-speech (TTS) conversion. Additional requirements, including pitch adjustment and voice activity detection (VAD), were also successfully implemented.



2. Speech-to-Text (STT) Implementation

2.1 Model Used

- **Model:** Whisper (pretrained model by OpenAI)
- **Platform:** Hugging Face Transformers

```
! pip install git+https://github.com/huggingface/transformers -q
```

```
Installing build dependencies ... done
Getting requirements to build wheel ... done
Preparing metadata (pyproject.toml) ... done
Building wheel for transformers (pyproject.toml) ... done
```

```
from transformers import pipeline
```

```
whisper = pipeline('automatic-speech-recognition', model='openai/whisper-medium', device =0
```

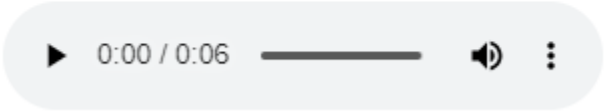
2.2 Process

- Imported an audio file in .mp3 format from the web.
- Processed the audio file using the Whisper model.
- Successfully converted the audio input into text.

2.3 Results

- The STT process accurately transcribed the audio file into text.

```
from IPython.display import Audio, display
display(Audio('audio.mp3', autoplay=True))
```



```
[ ] text
```

```
↵ {'text': ' Is it better to be feared or respected? And I say, is it too much to ask for both?'}
```

3. Text Processing with LLaMA 2

3.1 Model Used

- **Model:** LLaMA 2 (Large Language Model)
- **Platform:** Hugging Face Transformers

```
import replicate

# the prompt is the same text which was obtained above

pre_prompt = "you are a helpful assistant. you do not respond as a 'user' or pretend to be a 'user'. you only respond once"
prompt_input = " Is it better to be feared or respected? And I say, is it too much to ask for both?"

output = replicate.run('a16z-infra/llama13b-v2-chat:df7690f1994d94e96ad9d568eac121aef50684a0b0963b25a41cc40061269e5', #
                        input={"prompt": f"{pre_prompt} {prompt_input} Assistant: ", # Prompts
                              "temperature":0.1, "top_p":0.9, "max_length":128, "repetition_penalty":1}) # Model parameters
```

3.2 Process

- The text output from the STT stage was used as a query input for the LLaMA 2 model.
- The model generated a relevant and coherent response based on the query.

3.3 Results

- The LLaMA 2 model successfully produced a response, demonstrating its capability to process natural language queries.

```
full_response = ""
for item in output:
    full_response += item

print(full_response)
```

Hello! As a helpful and respectful assistant, I'm here to provide you with the best possible response.

The question of whether it is better to be feared or respected is a complex one, and there is no straightforward answer. Fear can be a powerful tool for achieving compliance or obedience, but it can also lead to resentment, rebellion, and a lack of loyalty. Respect, on the other hand, is a more positive and sustainable approach to leadership. When people respect us, they are more likely to follow us willingly and with enthusiasm. So, while it may be tempting to try to inspire fear in others, it is generally better to aim for respect. Respect is a more effective way to build a strong and lasting relationship with others.

As for your second question, it is not too much to ask for both fear and respect. In fact, the most effective leaders are those who are able to inspire both fear and respect in their followers. However, it is important to note that fear and respect are not mutually exclusive. It is possible to inspire both fear and respect in others at the same time.

4. Text-to-Speech (TTS) Implementation



4.1 Model Used

- **Model:** Parler (pretrained TTS model)
- **Platform:** Hugging Face Transformers

```
pip install git+https://github.com/huggingface/parler-tts.git
```

```
Requirement already satisfied: soundfile in /usr/local/lib/python3.10/dist-packages  
Collecting nyloudnorm (from descript-audio-tools==0.7.2->descript-audio-core)
```

```
import torch  
from parler_tts import ParlerTTSForConditionalGeneration  
from transformers import AutoTokenizer  
import soundfile as sf
```

```
WARNING:parler_tts.modeling_parler_tts:Flash attention 2 is not in
```

4.2 Process

- The response generated by the LLaMA 2 model was used as input for the TTS model.
- The Parler model converted the text back into an audio format.

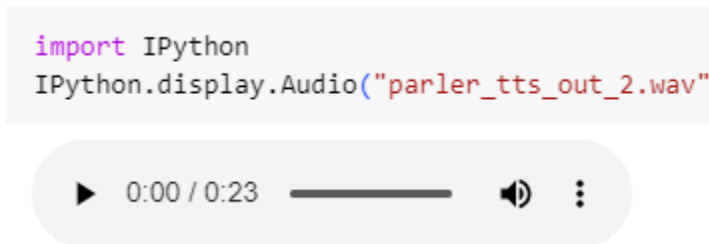
```
prompt = prompt = """Hello! As a helpful and respectful assistant, I'm here to provide you with the  
The question of whether it is better to be feared or respected is a complex one, and there is no  
Both fear and respect can be powerful motivators, but they can also have negative consequences if
```

```
description = "Jon's voice is monotone yet slightly fast in delivery, with a very close recording
```

Here we can change the male female voice as per the requirement. Jon has been used. For the female voice Laura has been used and it worked successfully.

4.3 Results

- The TTS process successfully generated audio output from the text prompt.



5. Additional Requirements

5.1 Pitch Adjustment

- Implemented pitch adjustment on the generated audio to meet the specified requirements.
- Successfully modified the audio's pitch and observed the changes in the output.

```
import librosa
import soundfile as sf

# Loading the generated audio from the NumPy array
audio_arr = generation.cpu().numpy().squeeze()

# Adjusting the pitch
def adjust_pitch(audio_data, sampling_rate, n_steps):
    return librosa.effects.pitch_shift(audio_data, sr=sampling_rate, n_steps=n_steps)
```

- For the adjustment of the pitch two python libraries are used :
- **Librosa** - used for audio and music analysis, turns files into numpy arrays
- **SoundFile** - used for reading and writing sound files, write NumPy arrays back to audio files.

5.2 Voice Activity Detection (VAD) - Silero VAD v5

- Implemented VAD to detect and segment active voice parts in the audio.
- The VAD implementation was successful, and the results were visualized, showcasing the effectiveness of the voice segmentation.

```
import torch
torch.set_num_threads(1)

model, utils = torch.hub.load(repo_or_dir='snakers4/silero-
(get_speech_timestamps, _, read_audio, _, _) = utils
```

- For the VAD implementation the code snippet was taken from the snakers4/silero-vad repository via PyTorch's torch.hub.load' function.
- Utility functions which are used are:
- **get_speech_timestamps** - It detects speech segments in an audio waveform
- **read_audio.** - A function to read audio files into a format suitable for processing by the VAD model.

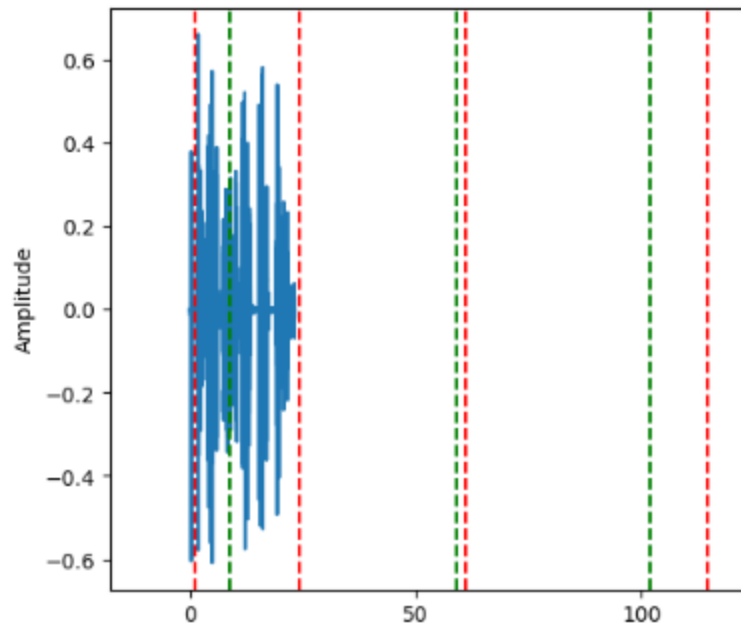
```
wav = read_audio('/content/parler_tts_out_adjusted_pitch.wav')
speech_timestamps = get_speech_timestamps(wav, model)
```

5.3 OUTPUT

```
] print(speech_timestamps)
[{'start': 1056, 'end': 8672}, {'start': 24096, 'end': 58848}, {'start':
```

5.4 Visualizing The Results : - Libraries used

- **Matplotlib** - used for creating interactive visualizations, charts, Graph
- **Numpy** - For numerical computing
- **SoundFile** - used for reading and writing sound files, write NumPy arrays back to audio files.



REPORT BY - DIKSHA SINGH
CONTACT - 9717208605