**Documentation**

**Description:**

Created a End to End voice assistant Pipleine which takes the inputs in the form of audio file which is converted into text to feed it into transformemr model which generates the output that is again transformer into the audio fomat using the TTS model.

**Work flow:**

**Step-1: Pre-processing of the Audio file:**

First my audio is recorded in which the query is present in the audio file format. Then the pre-processing is done by loading the audio file using the **Library: Pydub** **(which is used to load the audio and for pre-processing task)** to convert the sampling/frame rate from 48KHz (which was of the original audio file) to the 16KHz which is asked to do so. Same with the audio channel initially the audio channel of it was 2 (stereo) and converted into 1 (mono) audio channel.

To identify and transcribe the text of this audio VAD (Voice Activity Detection) = 0.5 as per asked is used which will identify that which part of the audio have the speeched segment or which does have the background noise or no noise which will make the whole pipeline as less resource intensive and leads to better accuracy which is done by using the **Library: Pydub, whisper and numpy**

**Step – 2: Feeding text into LLM:**

The text which we got after transcribing the audio file is saved in a variable named as the input\_text. The transformer model form Hugging face from **Library: Transformers** which is used to use the GPT-2 Medium model which is best suited for these types of tasks.  
**working of LLM:**

* The poretrained model along with it’s tokenizer is loaded is first loaded or downloaeded. The purpose of the tokenizer is to convert the words or characters into the unique numerical numbers.
* The input\_text is feeded into the model with the help of the prompt.
* The output generated is stored for further passing into the TTS model.
* Additional step: By default this LLM model gives the prompt along with the output text. So for polished output it is passed to function – clean\_response which will remove the content of prompt and question

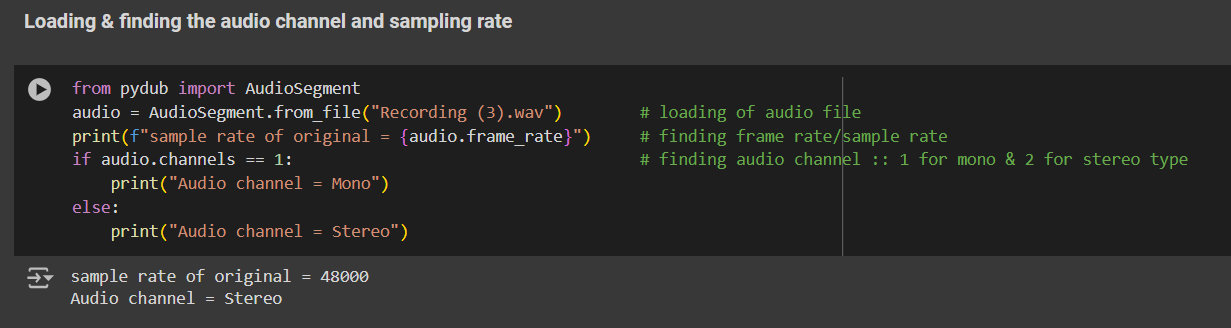
**Step – 3: TTS Model:**

The text that is generated by the LLM is passed into the Parler TTS model which is also a pre-trained model. It is initially loaded along with it’s specific tokenizer using the **Library: parler\_tts and transfomrers.** The text is feeded into it along with the description which defined the nature of voice: Male or Female voice, speed, pitch and other additional parameters as choice like: breathing=ng effect, or voice echos etc.

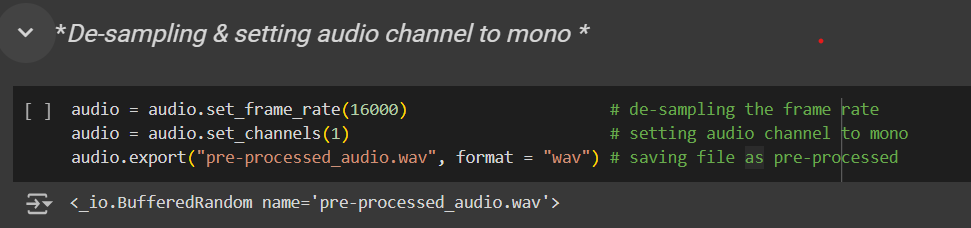
In the very first step the input text and the description is converted into tokens which is done by the tokenizer which is loaded initially after that feeded into model. Then audio is generated which is after stored in the audio file as output.

Code snippets:

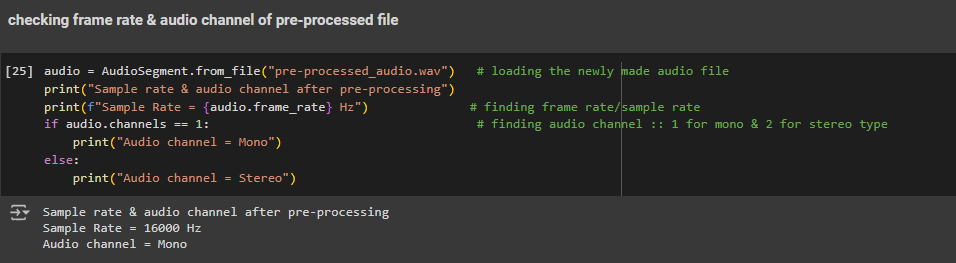
1. The audio file is loaded using the AudioSegment method from the pydub library for the pre-processing task. In this code snippet the audio file is oaded and the sampling rate and audio channel of it are found out.  
     
   which came to be 48000 and 2(stereo) which is not that is asked in the assignment. For the audio channel 1 means mono & 2 means stereo.



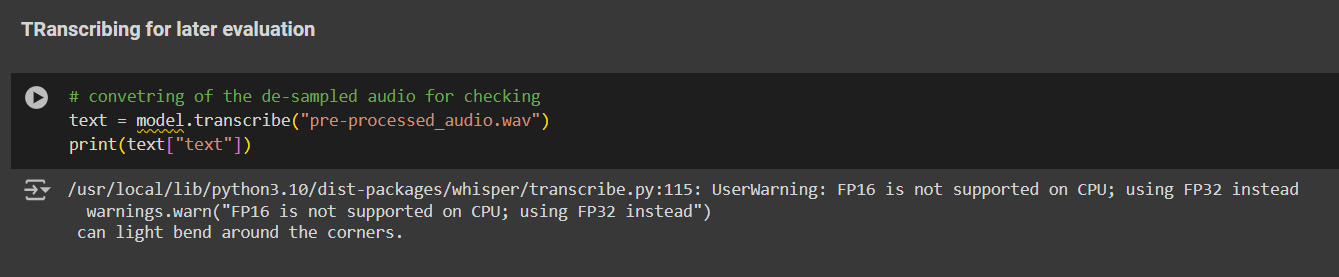
1. So to deal with it I have done the pre-processing to convert them into the desired settings by using the method as set\_frame\_rate() and set\_channels() I have de-sampled the audio file with the specific settings and saved as a new file so that we don’t have to pre-process it again and again. As seen in the below snippet.



1. Then to be sure the new audio file which is created I have checked it’s sampling rate and audio channel again and as we can see that not the sampling rate and audio channel are correct.



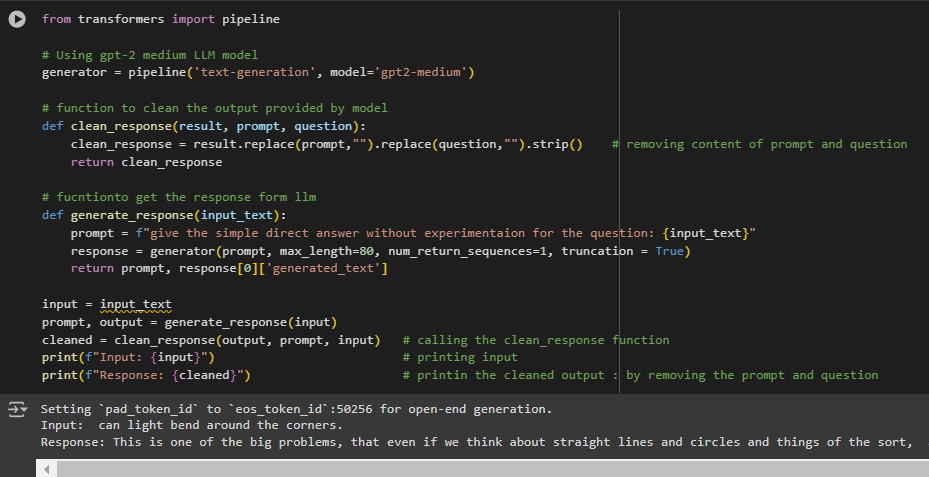
1. Transcribing the audio file so that the true transcription of it can be used as last for evaluation.



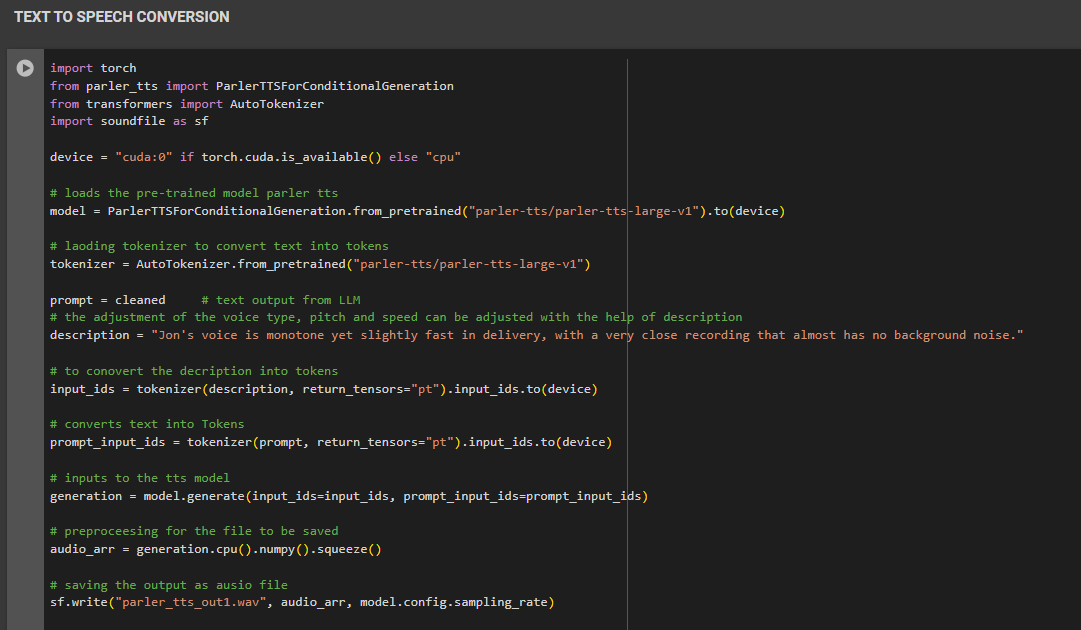
1. After it the VAD threshold is to be taken as 0.5.  
   VAD (Voice activity detection) it means to detect the speech portion form the audio and neglect all the portion of it which have either the background noise or is silent. It will make the process as less computationally expensive and more accurate as some background noises can also be recognized as the speeched part.



1. The input\_text is taken as the input form the audio file and feeded into the LLM model which in my case is gpt2-medium. By the default nature of this Model. It always gives the prompt and the question with appending with the response. So for the polished form I have processed the output of this model by removing the content of the prompt and the question from the response by using the user-defined function named clean\_response().



1. After getting the output form the LLM model this output is fed into the TTS (text to speech) model knwoln as parler\_tts. The working of it is in the way as: It first loads the tokenizer and model. Then the description is passed in which the setting of the voice if passed in the form of instructions. Then the text and the description is converted into the Tokens by their tokenizer. Then the speech audio file is created.



**Documentation and the diagrams that supported my implementation**

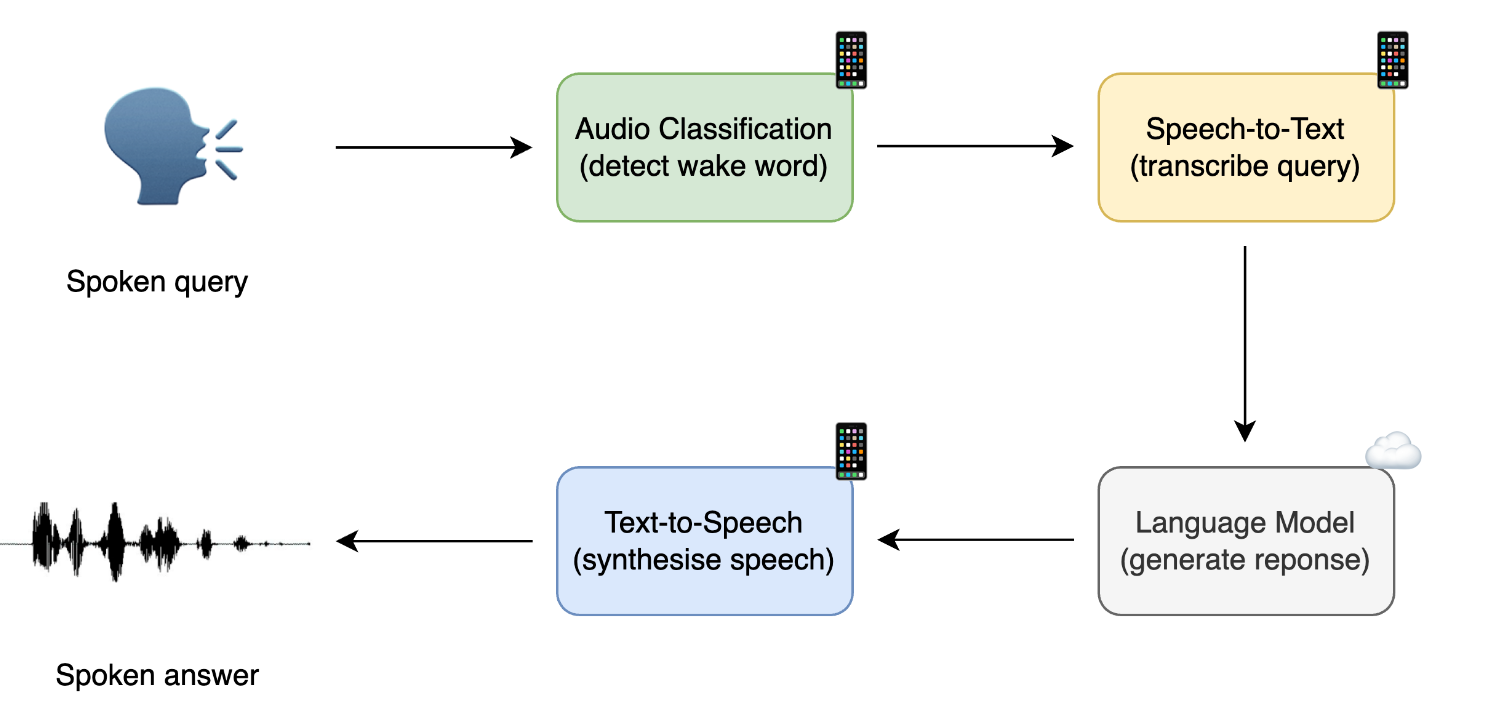


Diagram showing the high level flow

Whisper model research paper: https://arxiv.org/pdf/2212.04356

TTS model: https://huggingface.co/parler-tts/parler-tts-large-v1