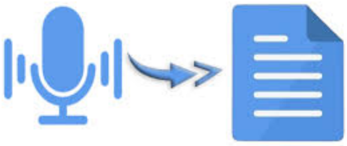
FIRST PART



AUDIO TO TEXT

- **Single channel audio**
- **Sample rate : 16 kHz**

STEP-1:Installing required model and libraries

```
!pip install openai-whisper
!pip install webrtcvad
!pip install pydub
!apt-get install ffmpeg
!pip install edge-tts
```

```
Requirement already satisfied: openai-whisper in /opt/conda/lib/python3.10/site-packages (20231117)
Requirement already satisfied: triton<3,>=2.0.0 in /opt/conda/lib/python3.10/site-packages (from openai-whisper) (2.3.1)
Requirement already satisfied: numba in /opt/conda/lib/python3.10/site-packages (from openai-whisper) (0.58.1)
Requirement already satisfied: numpy in /opt/conda/lib/python3.10/site-packages (from openai-whisper) (1.26.4)
Requirement already satisfied: torch in /opt/conda/lib/python3.10/site-packages (from openai-whisper) (2.4.0+cpu)
Requirement already satisfied: tqdm in /opt/conda/lib/python3.10/site-packages (from openai-whisper) (4.66.4)
Requirement already satisfied: more-itertools in /opt/conda/lib/python3.10/site-packages (from openai-whisper) (10.3.0)
Requirement already satisfied: tiktoken in /opt/conda/lib/python3.10/site-packages (from openai-whisper) (0.7.0)
Requirement already satisfied: filelock in /opt/conda/lib/python3.10/site-packages (from triton<3,>=2.0.0->openai-whispe
Requirement already satisfied: llvmlite<0.42,>=0.41.0dev0 in /opt/conda/lib/python3.10/site-packages (from numba->openai
Requirement already satisfied: regex>=2022.1.18 in /opt/conda/lib/python3.10/site-packages (from tiktoken->openai-whispe
Requirement already satisfied: requests>=2.26.0 in /opt/conda/lib/python3.10/site-packages (from tiktoken->openai-whispe
Requirement already satisfied: typing-extensions>=4.8.0 in /opt/conda/lib/python3.10/site-packages (from torch->openai-wl
Requirement already satisfied: sympy in /opt/conda/lib/python3.10/site-packages (from torch->openai-whisper) (1.12)
Requirement already satisfied: networkx in /opt/conda/lib/python3.10/site-packages (from torch->openai-whisper) (3.3)
Requirement already satisfied: jinja2 in /opt/conda/lib/python3.10/site-packages (from torch->openai-whisper) (3.1.4)
Requirement already satisfied: fsspec in /opt/conda/lib/python3.10/site-packages (from torch->openai-whisper) (2024.6.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /opt/conda/lib/python3.10/site-packages (from requests>=2.26.
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.10/site-packages (from requests>=2.26.0->tiktoken-
Requirement already satisfied: urllib3<3,>=1.21.1 in /opt/conda/lib/python3.10/site-packages (from requests>=2.26.0->tik
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/lib/python3.10/site-packages (from requests>=2.26.0->tik
Requirement already satisfied: MarkupSafe>=2.0 in /opt/conda/lib/python3.10/site-packages (from jinja2->torch->openai-wh
Requirement already satisfied: mpmath>=0.19 in /opt/conda/lib/python3.10/site-packages (from sympy->torch->openai-whispe
Requirement already satisfied: webrtcvad in /opt/conda/lib/python3.10/site-packages (2.0.10)
Requirement already satisfied: pydub in /opt/conda/lib/python3.10/site-packages (0.25.1)
Reading package lists... Done
Building dependency tree
Reading state information... Done
ffmpeg is already the newest version (7:4.2.7-0ubuntu0.1).
0 upgraded, 0 newly installed, 0 to remove and 30 not upgraded.
Requirement already satisfied: edge-tts in /opt/conda/lib/python3.10/site-packages (6.1.12)
Requirement already satisfied: aiohttp>=3.8.0 in /opt/conda/lib/python3.10/site-packages (from edge-tts) (3.9.5)
Requirement already satisfied: certifi>=2023.11.17 in /opt/conda/lib/python3.10/site-packages (from edge-tts) (2024.7.4)
Requirement already satisfied: aiosignal>=1.1.2 in /opt/conda/lib/python3.10/site-packages (from aiohttp>=3.8.0->edge-tt
Requirement already satisfied: attrs>=17.3.0 in /opt/conda/lib/python3.10/site-packages (from aiohttp>=3.8.0->edge-tts)
Requirement already satisfied: frozenlist>=1.1.1 in /opt/conda/lib/python3.10/site-packages (from aiohttp>=3.8.0->edge-t
Requirement already satisfied: multidict<7.0,>=4.5 in /opt/conda/lib/python3.10/site-packages (from aiohttp>=3.8.0->edge
Requirement already satisfied: yarl<2.0,>=1.0 in /opt/conda/lib/python3.10/site-packages (from aiohttp>=3.8.0->edge-tts)
Requirement already satisfied: async-timeout<5.0,>=4.0 in /opt/conda/lib/python3.10/site-packages (from aiohttp>=3.8.0->
Requirement already satisfied: idna>=2.0 in /opt/conda/lib/python3.10/site-packages (from yarl<2.0,>=1.0->aiohttp>=3.8.0
```

STEP-2:Importing the installed liraries and model

```
import whisper
import numpy as np
import wave
import webrtcvad
from pydub import AudioSegment
```

STEP-3:loading the model with base architecture and initialising the VAD(virtual audio detection)

```
model = whisper.load_model("base")
```

```
/opt/conda/lib/python3.10/site-packages/whisper/__init__.py:146: FutureWarning: You are using `torch.load` with `weights_
  checkpoint = torch.load(fp, map_location=device)
```

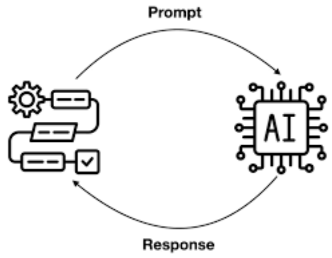
STEP-4:

- **Creating a functon for resampling the loaded audio using pydub and meeting the requirements (mono channel , sample rate)**
- **Creating a function for transcribing the resulted audio into text tuning VAD sensitivity**

```
def read_audio(file_path):

    audio = AudioSegment.from_file(file_path)
    audio = audio.set_channels(1).set_frame_rate(16000)  ## Hardcoing the sample rate to be 16kHz and mono-channeled
    audio_data = np.array(audio.get_array_of_samples(), dtype=np.int16)

    return audio_data, 16000


def transcribe_with_vad(audio_file):
    # Load audio
    audio, sample_rate = read_audio(audio_file)

    vad = webrtcvad.Vad()
    vad.set_mode(1)  ## adjusting the agressivnes of vad

    frame_duration = 30
    frame_size = int(sample_rate * frame_duration / 1000)
    frame_byte_size = frame_size * 2

    segments = []

    for start in range(0, len(audio), frame_size): ## saving audio_data after required padding for same sample size chunks a
        stop = min(start + frame_size, len(audio))
        frame = audio[start:stop]

        if len(frame) < frame_size:

            frame = np.pad(frame, (0, frame_size - len(frame)), 'constant')
        elif len(frame) > frame_size:

            frame = frame[:frame_size]


        if vad.is_speech(frame.tobytes(), sample_rate):
            segments.append(frame)

    if segments:
        detected_audio = np.concatenate(segments)


        detected_audio = detected_audio.astype(np.float32) / 32768.0


        result = model.transcribe(detected_audio, language="en")
        return result['text']
    else:
        return "No speech detected."


output_text = transcribe_with_vad("/kaggle/input/audio-file/download") ##output
print("Transcription:", output_text)
```

```
/opt/conda/lib/python3.10/site-packages/whisper/transcribe.py:115: UserWarning: FP16 is not supported on CPU; using FP32
  warnings.warn("FP16 is not supported on CPU; using FP32 instead")
Transcription:  But what if somebody decides to break it? Be careful that you keep adequate coverage, but look for place:
```

SECOND PART



TEXT TO TEXT

- **1-2 sentence of output**

STEP-1:Importing required libraries

```
import os
from huggingface_hub import hf_hub_download
```

STEP-2:Initialising required fields to variables for accessing model

```
api_key ="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"
model_id = "lmsys/fastchat-t5-3b-v1.0"
filenames = [
    "added_tokens.json",
    "config.json",
    "generation_config.json",
    "pytorch_model.bin",
    "special_tokens_map.json",
    "spiece.model",
    "tokenizer_config.json"
]
```

STEP-3:Downloading the required files for initialising model

```
for filename in filenames:
    download_model_path = hf_hub_download(
        repo_id = model_id,
        filename = filename,
        token = api_key
    )
```

STEP-4:Importing AutoTokenizer for tokenizing the output text,model_skeleton,Pipeline as entry point

```
from transformers import AutoTokenizer , AutoModelForCausalLM
from transformers import pipeline,AutoModelForSeq2SeqLM


tokenizer = AutoTokenizer.from_pretrained(model_id,legacy=False)
model = AutoModelForSeq2SeqLM.from_pretrained(model_id)

pipeline = pipeline("text2text-generation",model=model,device=-1,tokenizer=tokenizer
                   ,max_length = 300) #Adjusting token length for satisfying requirements(output_lergth)
model_output = pipeline(output_text)


model_output = model_output[0]['generated_text']
print(model_output)
```

```
You kneep a dyir or wax or gas needed oon thi type of song.
```

THIRD PART

TEXT TO AUDIO

- **Variable Pitch**
- **Variable Voice gender**
- **Variable rate**

STEP-1 : Observe the list of voice/dialects for setting custom switch of voice parameter

```
import edge_tts
async def list_voices():
    voices = await edge_tts.list_voices()
    for voice in voices:
        print(f"Name: {voice['Name']}, Gender: {voice['Gender']}, Locale: {voice['Locale']}")

await(list_voices())
```

```
Name: Microsoft Server Speech Text to Speech Voice (af-ZA, AdriNeural), Gender: Female, Locale: af-ZA
Name: Microsoft Server Speech Text to Speech Voice (af-ZA, WillemNeural), Gender: Male, Locale: af-ZA
Name: Microsoft Server Speech Text to Speech Voice (sq-AL, AnilaNeural), Gender: Female, Locale: sq-AL
Name: Microsoft Server Speech Text to Speech Voice (sq-AL, IlirNeural), Gender: Male, Locale: sq-AL
Name: Microsoft Server Speech Text to Speech Voice (am-ET, AmehaNeural), Gender: Male, Locale: am-ET
Name: Microsoft Server Speech Text to Speech Voice (am-ET, MekdesNeural), Gender: Female, Locale: am-ET
Name: Microsoft Server Speech Text to Speech Voice (ar-DZ, AminaNeural), Gender: Female, Locale: ar-DZ
Name: Microsoft Server Speech Text to Speech Voice (ar-DZ, IsmaelNeural), Gender: Male, Locale: ar-DZ
Name: Microsoft Server Speech Text to Speech Voice (ar-BH, AliNeural), Gender: Male, Locale: ar-BH
Name: Microsoft Server Speech Text to Speech Voice (ar-BH, LailaNeural), Gender: Female, Locale: ar-BH
Name: Microsoft Server Speech Text to Speech Voice (ar-EG, SalmaNeural), Gender: Female, Locale: ar-EG
Name: Microsoft Server Speech Text to Speech Voice (ar-EG, ShakirNeural), Gender: Male, Locale: ar-EG
Name: Microsoft Server Speech Text to Speech Voice (ar-IQ, BasselNeural), Gender: Male, Locale: ar-IQ
Name: Microsoft Server Speech Text to Speech Voice (ar-IQ, RanaNeural), Gender: Female, Locale: ar-IQ
Name: Microsoft Server Speech Text to Speech Voice (ar-JO, SanaNeural), Gender: Female, Locale: ar-JO
Name: Microsoft Server Speech Text to Speech Voice (ar-JO, TaimNeural), Gender: Male, Locale: ar-JO
Name: Microsoft Server Speech Text to Speech Voice (ar-KW, FahedNeural), Gender: Male, Locale: ar-KW
Name: Microsoft Server Speech Text to Speech Voice (ar-KW, NouraNeural), Gender: Female, Locale: ar-KW
Name: Microsoft Server Speech Text to Speech Voice (ar-LB, LaylaNeural), Gender: Female, Locale: ar-LB
Name: Microsoft Server Speech Text to Speech Voice (ar-LB, RamiNeural), Gender: Male, Locale: ar-LB
Name: Microsoft Server Speech Text to Speech Voice (ar-LY, ImanNeural), Gender: Female, Locale: ar-LY
Name: Microsoft Server Speech Text to Speech Voice (ar-LY, OmarNeural), Gender: Male, Locale: ar-LY
Name: Microsoft Server Speech Text to Speech Voice (ar-MA, JamalNeural), Gender: Male, Locale: ar-MA
Name: Microsoft Server Speech Text to Speech Voice (ar-MA, MounaNeural), Gender: Female, Locale: ar-MA
Name: Microsoft Server Speech Text to Speech Voice (ar-OM, AbdullahNeural), Gender: Male, Locale: ar-OM
Name: Microsoft Server Speech Text to Speech Voice (ar-OM, AyshaNeural), Gender: Female, Locale: ar-OM
Name: Microsoft Server Speech Text to Speech Voice (ar-QA, AmalNeural), Gender: Female, Locale: ar-QA
Name: Microsoft Server Speech Text to Speech Voice (ar-QA, MoazNeural), Gender: Male, Locale: ar-QA
Name: Microsoft Server Speech Text to Speech Voice (ar-SA, HamedNeural), Gender: Male, Locale: ar-SA
Name: Microsoft Server Speech Text to Speech Voice (ar-SA, ZariyahNeural), Gender: Female, Locale: ar-SA
Name: Microsoft Server Speech Text to Speech Voice (ar-SY, AmanyNeural), Gender: Female, Locale: ar-SY
Name: Microsoft Server Speech Text to Speech Voice (ar-SY, LaithNeural), Gender: Male, Locale: ar-SY
Name: Microsoft Server Speech Text to Speech Voice (ar-TN, HediNeural), Gender: Male, Locale: ar-TN
Name: Microsoft Server Speech Text to Speech Voice (ar-TN, ReemNeural), Gender: Female, Locale: ar-TN
Name: Microsoft Server Speech Text to Speech Voice (ar-AE, FatimaNeural), Gender: Female, Locale: ar-AE
Name: Microsoft Server Speech Text to Speech Voice (ar-AE, HamdanNeural), Gender: Male, Locale: ar-AE
Name: Microsoft Server Speech Text to Speech Voice (ar-YE, MaryamNeural), Gender: Female, Locale: ar-YE
Name: Microsoft Server Speech Text to Speech Voice (ar-YE, SalehNeural), Gender: Male, Locale: ar-YE
Name: Microsoft Server Speech Text to Speech Voice (az-AZ, BabekNeural), Gender: Male, Locale: az-AZ
Name: Microsoft Server Speech Text to Speech Voice (az-AZ, BanuNeural), Gender: Female, Locale: az-AZ
Name: Microsoft Server Speech Text to Speech Voice (bn-BD, NabanitaNeural), Gender: Female, Locale: bn-BD
Name: Microsoft Server Speech Text to Speech Voice (bn-BD, PradeepNeural), Gender: Male, Locale: bn-BD
Name: Microsoft Server Speech Text to Speech Voice (bn-IN, BashkarNeural), Gender: Male, Locale: bn-IN
Name: Microsoft Server Speech Text to Speech Voice (bn-IN, TanishaaNeural), Gender: Female, Locale: bn-IN
Name: Microsoft Server Speech Text to Speech Voice (bs-BA, GoranNeural), Gender: Male, Locale: bs-BA
Name: Microsoft Server Speech Text to Speech Voice (bs-BA, VesnaNeural), Gender: Female, Locale: bs-BA
Name: Microsoft Server Speech Text to Speech Voice (bg-BG, BorislavNeural), Gender: Male, Locale: bg-BG
Name: Microsoft Server Speech Text to Speech Voice (bg-BG, KalinaNeural), Gender: Female, Locale: bg-BG
```

```
Name: Microsoft Server Speech Text to Speech Voice (my-MM, NilarNeural), Gender: Female, Locale: my-MM
Name: Microsoft Server Speech Text to Speech Voice (my-MM, ThihaNeural), Gender: Male, Locale: my-MM
Name: Microsoft Server Speech Text to Speech Voice (ca-ES, EnricNeural), Gender: Male, Locale: ca-ES
Name: Microsoft Server Speech Text to Speech Voice (ca-ES, JoanaNeural), Gender: Female, Locale: ca-ES
Name: Microsoft Server Speech Text to Speech Voice (zh-HK, HiuGaaiNeural), Gender: Female, Locale: zh-HK
Name: Microsoft Server Speech Text to Speech Voice (zh-HK, HiuMaanNeural), Gender: Female, Locale: zh-HK
Name: Microsoft Server Speech Text to Speech Voice (zh-HK, WanLungNeural), Gender: Male, Locale: zh-HK
Name: Microsoft Server Speech Text to Speech Voice (zh-CN, XiaoxiaoNeural), Gender: Female, Locale: zh-CN
Name: Microsoft Server Speech Text to Speech Voice (zh-CN, XiaoyiNeural), Gender: Female, Locale: zh-CN
```

STEP-2 :

- **Importing and using edge_tts with variable voice(gender),rate or speed,pitch**
- **Saving it at any format (mp3,wav)**

```
import edge_tts
import asyncio

text = model_output

voice_female = "en-SG-LunaNeural"
voice_male = "en-SG-WayneNeural"
rate = "+10%"
pitch = "+2Hz"

async def main():
    communicator = edge_tts.Communicate(text=text, voice=voice_female, rate=rate, pitch=pitch)
    output_file = "/kaggle/working/output_female_finally.wav"
    await communicator.save(output_file)
    print(f"Audio saved to {output_file}")

await(main())
```

```
Audio saved to /kaggle/working/output_female_finally.wav
```

FINAL STEP

JOINING ALL THREE PARTS INTO A FINAL PIPELINE

```
from transformers import AutoTokenizer , AutoModelForCausalLM
from transformers import pipeline,AutoModelForSeq2SeqLM
import edge_tts
class PIPELINE_Q:
    def __init__(self, audio_path, sample_rate, channels, aggressiveness, token_length, gender, rate, pitch):
        self.model = whisper.load_model("base")
        self.sample_rate = sample_rate
        self.audio_path = audio_path
        self.channels = channels
        self.threshold = aggressiveness
        self.token_length = token_length
        self.gender = gender
        self.rate = rate
        self.pitch = pitch

    def read_audio(self):
        audio = AudioSegment.from_file(self.audio_path)
        audio = audio.set_channels(self.channels).set_frame_rate(self.sample_rate)
        audio_data = np.array(audio.get_array_of_samples(), dtype=np.int16)
        return audio_data, self.sample_rate

    def transcribe_with_vad(self):
        audio, sample_rate = self.read_audio()

        vad = webrtcvad.Vad()
        vad.set_mode(self.threshold)

        frame_duration = 30
        frame_size = int(sample_rate * frame_duration / 1000)
        frame_byte_size = frame_size * 2

        segments = []

        for start in range(0, len(audio), frame_size):
            stop = min(start + frame_size, len(audio))
            frame = audio[start:stop]

            if len(frame) < frame_size:
                frame = np.pad(frame, (0, frame_size - len(frame)), 'constant')
            elif len(frame) > frame_size:
                frame = frame[:frame_size]

            if vad.is_speech(frame.tobytes(), sample_rate):
                segments.append(frame)

        if segments:
            detected_audio = np.concatenate(segments)
            detected_audio = detected_audio.astype(np.float32) / 32768.0

            result = self.model.transcribe(detected_audio, language="en")
            return result['text']
        else:
            return "No speech detected."

    def text_to_text(self):
        result = self.transcribe_with_vad()
        model_id = "lmsys/fastchat-t5-3b-v1.0"  # Replace with the actual model ID
        tokenizer = AutoTokenizer.from_pretrained(model_id,legacy=False)
        model = AutoModelForSeq2SeqLM.from_pretrained(model_id)


        pipeline_1 = pipeline("text2text-generation",model=model,device=-1,tokenizer=tokenizer
                   ,max_length = 300) #Adjusting token length for satisfying requirements(output_lergth)

        output = pipeline_1(result, max_length=self.token_length)
        return output[0]['generated_text']


    async def text_to_audio(self, text):
        voice_female = "en-SG-LunaNeural"
        voice_male = "en-SG-WayneNeural"

        voice = voice_male if self.gender == "male" else voice_female
        output_file = "/kaggle/working/output_finally.wav"

        communicator = edge_tts.Communicate(text=text, voice=voice, rate=self.rate, pitch=self.pitch)
        await communicator.save(output_file)
        print(f"Audio saved to {output_file}")

    async def run_pipeline(self):
        transcribed_text = self.transcribe_with_vad()
        generated_text = self.text_to_text()
```

```
        await self.text_to_audio(generated_text)


pipeline_true_final = PIPELINE_Q(audio_path="/kaggle/input/audio-file/download", sample_rate=16000, channels=1, aggressivene
                    token_length=300, gender="male", rate="+10%", pitch="+2Hz")

await pipeline_true_final.run_pipeline()
```

```
/opt/conda/lib/python3.10/site-packages/whisper/__init__.py:146: FutureWarning: You are using `torch.load` with `weights_
  checkpoint = torch.load(fp, map_location=device)
```