Experiment 5 - Cognitive Computing Application on Audio data using Python

Roll No.	37
Name	Mikil Lalwani
Class	D20 B
Subject	Data Science Lab
LO Mapped	L2: Explore use cases of Cognitive Computing

Aim: To implement Cognitive computing for audio data using python .

Python Library Function Used:

Speech recognition is a machine's ability to listen to spoken words and identify them. You can then use speech recognition in Python to convert the spoken words into text, make a query or give a reply. You can even program some devices to respond to these spoken words. You can do speech recognition in python with the help of computer programs that take in input from the microphone, process it, and convert it into a suitable form.

Speech recognition seems highly futuristic, but it is present all around you. Automated phone calls allow you to speak out your query or the query you wish to be assisted on; your virtual assistants like Siri or Alexa also use speech recognition to talk to you seamlessly.

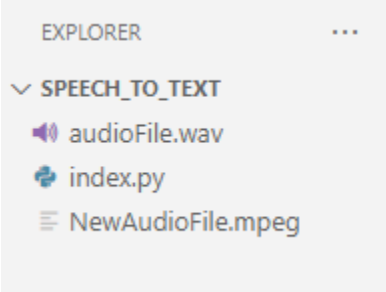
How Does Speech Recognition work?
Speech recognition in Python works with algorithms that perform linguistic and acoustic modeling. Acoustic modeling is used to recognize phenones/phonetics in our speech to get the more significant part of speech, as words and sentences.

Speech recognition starts by taking the sound energy produced by the person speaking and converting it into electrical energy with the help of a microphone. It then converts this electrical energy from analog to digital, and finally to text.
It breaks the audio data down into sounds, and it analyzes the sounds using algorithms to find the most probable word that fits that audio. All of this is done using Natural Language Processing and Neural Networks. Hidden Markov models can be used to find temporal patterns in speech and improve accuracy.

AssemblyAI web api
AssemblyAI provides AI models to transcribe and analyse audio and speech data through our production-ready, scalable web API. Our models are customizable and enable features such as content moderation, sentiment analysis, PII redaction, key phrase identification, and speaker diarization.

Code and Observation:

Folder structure:



The audio files are **audioFIle.wav** and **NewAudioFIle.mpeg**

index.py

```
# AssemblyAI api

import requests
import json
import time

base_url = "https://api.assemblyai.com/v2"

# Enter the api key provided by AssemblyAI in the headers dictionary below.
headers = {
   "authorization": "<your_api_key_from_AssemblyAI>"
}
with open("./NewAudioFile.mpeg", "rb") as f:
  response = requests.post(base_url + "/upload",
                           headers=headers,
                           data=f)

upload_url = response.json()["upload_url"]

data = {
   "audio_url": upload_url
}

url = base_url + "/transcript"
response = requests.post(url, json=data, headers=headers)

transcript_id = response.json()['id']
```

```
print("Converting speech to text.....")

polling_endpoint = f"https://api.assemblyai.com/v2/transcript/{transcript_id}"

while True:
  transcription_result = requests.get(polling_endpoint, headers=headers).json()
  if transcription_result['status'] == 'completed':
    print(transcription_result['text'])
    break
  elif transcription_result['status'] == 'error':
`     raise RuntimeError(f"Transcription failed: {transcription_result['error']}")
  else:
    time.sleep(3)
```

Output:

```
PS C:\Users\admin\Desktop\speech_to_text> python .\index.py
Converting speech to text.....
Text data: Eleven temperature and humidity sensor features a temperature and humidity sensor complex with a calibrated digital signal.
PS C:\Users\admin\Desktop\speech_to_text>
```

Conclusion:

Used AssemblyAI's web api for converting speech data into text format.