MITRA: Personalized Voice Assistant

A Project / Dissertation as a Course requirement for
BSc Computer Science (Hons)

KALUR LALITH ADITYA
194209

SRI SATHYA SAI INSTITUTE OF HIGHER LEARNING
(Deemed to be University)

Department of Mathematics and Computer Science
Muddenahalli Campus

April, 2022

Acknowledgements

I would like to dedicate this page for those who offered help and assisted me in making this project.

Throughout the entire process, I was assisted and motivated to work and putting in efforts.

Firstly, I would like to express my gratitude to **Dr. Sampath Lonka** who went along with us till the end providing insights and feedback and helping out in the process whenever we asked for it.

I thank the Deputy Director **B. Venkataramana** for planning the project systematically and providing us with lab and other resources.

I thank my partner, **Shri Bellapu Ravvala Sricharan (194206)** who provided ample resources and support.

I thank my **teachers** of this institute who taught me the concepts from my 1st semester onwards, which helped me a lot in this project.

I thank all my **classmates** who provided support.

I would like to specially thank **Shri Nagarjuna Ponati (194210)** for assisting our project by providing hardware resources whenever we asked.

Offering our efforts at Swami's Lotus Feet, we dedicate this project to Swami.

CERTIFICATE

This is to certify that this Project / Dissertation titled **Mitra-Personalized Voice Assistant** submitted by **KALUR LALITH ADITYA**, **194209**, Department of Mathematics and Computer Science, Muddenahalli Campus is a bonafide record of the original work done under my supervision as a Course requirement for the Degree of Bachelor of Science (Hons) in Computer Science

…………………………………

Dr. Sampath Lonka
Project Supervisor

Countersigned by

…………………………………

Head of the Department

Place: Muddenahalli
Date: April 25, 2022

DECLARATION

The Project titled **Mitra-Personalized Voice Assistant** was carried out by me under the supervision of Dr. Sampath Lonka, Department of Mathematics and Computer Science, Muddenahalli Campus as a course requirement for the Degree of Bachelor of Science (Hons) in Computer Science and has not formed the basis for the award of any degree, diploma or any other such title by this or any other University.

Lalith Aditya

....

Place: Muddenahalli KALUR LALITH ADITYA

194209

Date: 25th April, 2022

Muddenahalli Campus

Contents

Abstract

Mitra is a voice assistant programmed to make things easy for people. Mitra provides features where user can interactively get the information the user wants. Mitra also performs tasks which include speech summarization where user can give the speech and Mitra reads out the summary of the speech.

Mitra is a command driven interactive personalized voice assistant, which takes the command in the input and does the respective operation and tasks.

Personal Assistants with more features and interactive one’s can make jobs and activities easier and enjoyable.

Mitra is one such personal assistant which provides basic and interesting features which is appropriate and relevant for the user to seek.

Chapter 1 Introduction

1.1 Voice Assistant using Python

1.1.1 What Is a Voice Assistant?

The majority of our attention is focused on and sought-after technology that simplifies our lives. There are numerous agents capable of carrying out this task of resolving our daily activities. With a voice assistant, technology can be a friend in assisting humans in making their lives easier, more enjoyable, and more comfortable. Here are some basic definitions, similarities, and differences:

Intelligent Personal Assistant: This is software that assists people with routine tasks, typically through the use of natural language. Intelligent personal assistants are capable of conducting online searches in order to find the answer to a user's question. Text or voice can be used to initiate an action.

Automated Personal Assistant: This is synonymous with intelligent personal assistant.

Smart Assistant: This typically refers to the types of physical items that can perform a variety of functions through the use of smart speakers that listen for a wake word to activate and perform specific tasks. Smart assistants include Amazon's Echo, Google's Home, and Apple's Home Pod.

Virtual Digital Assistants: These are automated software applications or platforms that aid the user in comprehending natural language, whether written or spoken.

Chatbot: Text is the main way to get help from chatbot. Chabot’s can simulate conversations with human users. Many companies use them in customer service to answer basic questions and connect with real people when needed.

Voice Assistant: The important thing here is the voice. Speech Assistant is a digital assistant that uses speech recognition, speech synthesis, and natural language processing (NLP) to provide services through specific applications.

For more details, see [1].

1.1.2 The Uses of Voice Assistants

Numerous devices that we use on a daily basis make use of voice assistants. They're on our smartphones and embedded in our homes' smart speakers. Numerous mobile applications and operating systems make use of them. Additionally, certain technologies in automobiles, as well as those in retail, education, healthcare, and telecommunications, can be operated via voice commands.

1.2 Popular Voice Assistants

When Apple released the iPhone 4s on October 4, 2011, Siri became the first digital virtual assistant to be included as standard on a smartphone. Siri entered the world of smart speakers with the launch of the Home Pod in February 2018.

On the Android platform, Google Now (later renamed Google Assistant) was introduced. It is also compatible with Apple's iOS operating system, but with limited functionality. Then the smart speakers came along, and “Alexa” and “Hey Google” became a part of many household conversations. Alexa by

Amazon is part of the Echo and the Dot. Google Assistant is part of the Google Home.

Bixby is a feature of Samsung. IBM is home to Watson. Cortana is built into Microsoft's Windows 10, Xbox One, and Windows Phone operating systems, while Nina is built into Nuance's Nina. M was previously available on Facebook, but its use in the Facebook Messenger app ceased in January 2018.By default, most of the voice assistants have somewhat female-sounding voices, although the user can change them to other voices. Many people refer to Siri, Alexa, and Cortana as “she” and not “it.”

1.3 Different Types of Voice Assistants

There are several types of voice assistants, but here are the main players!

Amazon

Amazon first announced its smart speaker, the 'Amazon Echo,' four years ago. This technology was wildly popular, lauded for its ability to check the weather, monitor traffic, play music and podcasts, and much more with the simplest vocal command. This was the catalyst for the smart speaker trend!

Apple

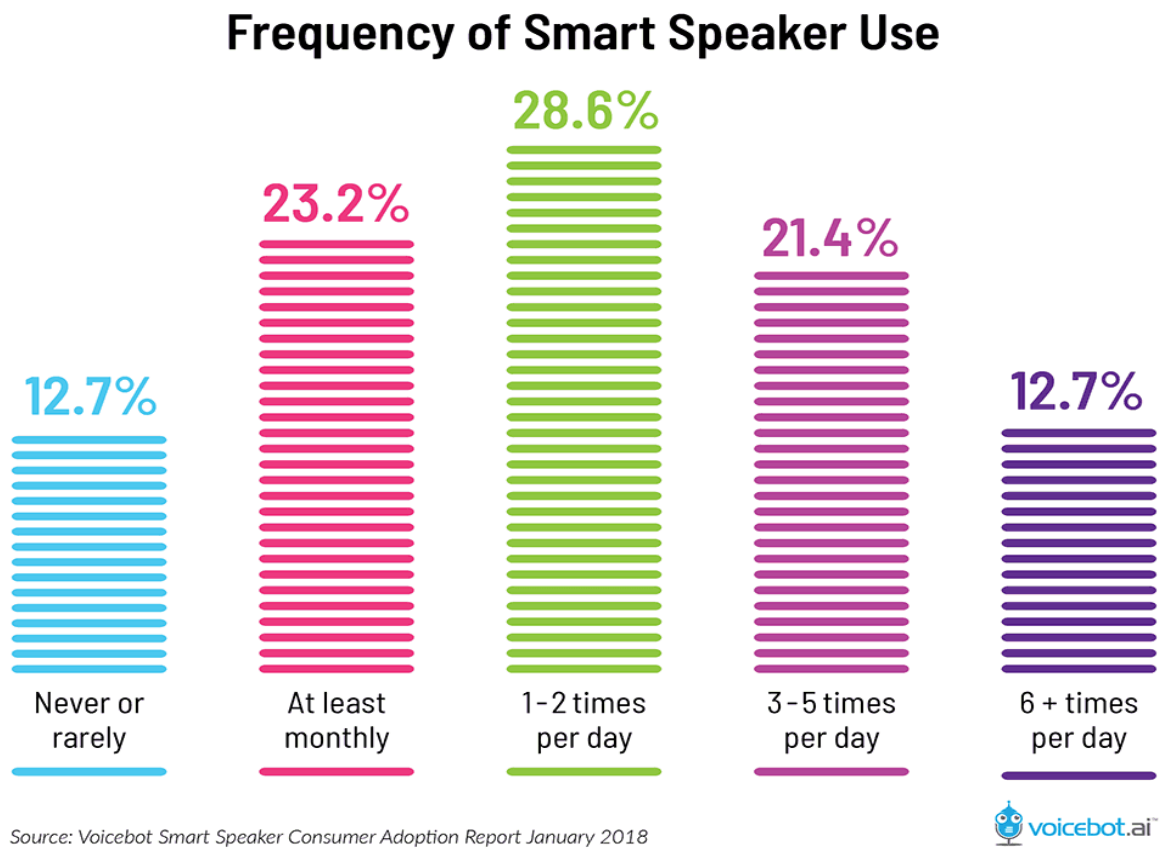
Apple entered the market first with Siri, long before the concept of voice recognition gained traction. While Siri began with iPhones, it has quickly expanded to include software and automotive products, as well as multiple mobile phone operating systems. Previously, Amazon and Google dominated the smart speaker market, but Apple recently made a splash with the launch of Home Pod, its own brand of smart home speaker.

Google Home

Google's 'Google Home' device has been a commercial and critical success, owing in large part to the vast amount of data already owned by Google. This includes search engine functionality and real-time data such as maps and traffic. Because the majority of people have a Google account or history, Google Home has successfully combined its fantastic voice recognition technology with personalized data.

By installing a smart hub that talks to your voice assistant, you can manage your Smart Home with your voice by controlling your compatible home automation devices. The ecosystem is expanding all the time, with more and more devices.

For more details, see [2].



1.4 The Drawbacks of Voice Assistants

Privacy:

Privacy is an issue, especially for smart speakers. The smart speaker is always listening while you wait for the word of awakening. On smartphones, the assistant is activated when you press a button or open the app. As soon as you wake up, you'll start recording an audio clip of what you say. These clips represent the files that are sent to the server to process the audio and create a response.

The real brain is not in the little speakers in our house. They are on another huge server somewhere. The speaker sends over an encrypted connection. The speaker does not record anything before the words of awakening.

"People confuse ‘always listen' to 'always record'," says Mutchler of Voicebot.ai. “The great thing about [smart speakers] is that they can eliminate background noise and emphasize wake words,” she continues. Only then will they start recording.

Accuracy:

Voice assistants don't always understand what we want. Sometimes we talk that way. Sometimes it's just because artificial intelligence hasn't learned how to do anything yet.

For more details, see [3].

Chapter 2 Mitra

2.1 What is Mitra?

Mitra is a command driven personalized voice assistant. Mitra provides you with features that include entertainment, news and current affairs cognizance, speech summarization, Wikipedia, music etc.

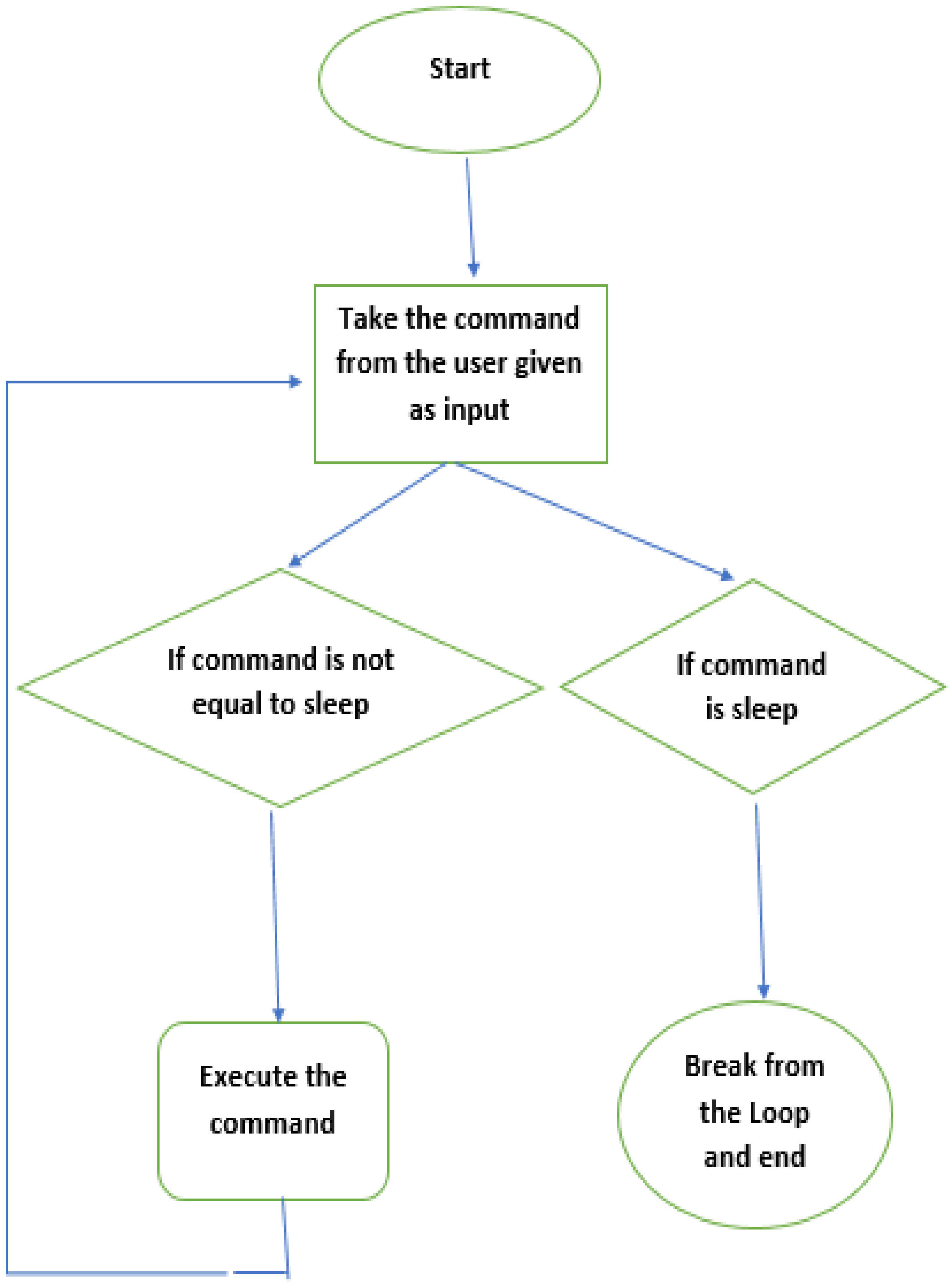
2.2 How does Mitra work?

This is a command driven program that runs in a loop until it is said to go to sleep by the person. In the process the person can get assistance of Mitra for his respective queries and requirements.

2.3 Features of Mitra

- Speech Recording
- Summarization of a text file
- Speech Summarization
- Conversion of a speech to a text file
- Telling time
- Finding information from Wikipedia
- Entertaining with a joke
- Speaking out National News
- Speaking out World News
- Playing Music
- Speaking about itself

2.4 Flowchart of Mitra


Start
Take the command from the user given as input
If command is not equal to sleep
If command is sleep
Execute the command
Break from the Loop and end

2.5 Packages and Language used in Mitra

Language Used: - **Python**

2.5.1 Speech Recognition

Speech recognition library with support for multiple engines and APIs, both online and offline.

2.5.2 pyttsx3

Pyttsx3 is a Python library for text-to-speech conversion. Unlike other libraries, it operates in the background and is compatible with both Python 2 and 3.

2.5.3 pywhatkit

Python includes a large number of built-in libraries that simplify our work. Among them, pywhatkit is a Python library for scheduling WhatsApp messages; it also has a number of other features.

The following are some of the characteristics of the pywhatkit module:

- Send WhatsApp messages.
- Play a YouTube video.
- Perform a Google Search.
- Get information on a particular topic.

The pywhatkit module can also be used for converting text into handwritten text images.

2.5.4 date time

This package implements the Zope-standard Date Time data type. Unless you require communication with Zope APIs, you are likely to be better off using Python's built-in date time module.

2.5.5 Wikipedia

Wikipedia is a Python library that makes it smooth to get entry to and parse records from Wikipedia.

2.5.6 pyjokes

On line jokes for programmers (jokes as a service)

2.5.7 requests

Requests is a simple, yet elegant, HTTP library.

2.5.8 json

JSON is a syntax for storing and changing data. JSON is text, written with JavaScript item notation. Python has an integrated package deal known as json, which may be used to paintings with JSON data.

2.5.9 sys

Python's sys module contains a variety of functions and variables for manipulating various aspects of the Python runtime environment. It enables interaction with the interpreter by providing access to variables and functions that have a strong relationship with the interpreter.

2.5.10 sound device

This Python module offers bindings for the Port Audio library and some comfort capabilities to play and document NumPy arrays containing audio signals. The sound tool module is to be had for Linux, macOS and Windows.

2.5.11 tkinter

Tkinter is the Python interface to Tk, that's the GUI toolkit for Tcl/Tk. Tcl (mentioned as tickle) is a scripting language regularly utilized in testing, prototyping, and GUI development. Tk is an open-source, cross-platform widget toolkit utilized by many unique programming languages to construct GUI programs.

2.5.12 queue

The queue module implements multi-producer, multi-patron queues. It is mainly beneficial in threaded programming while statistics should be exchanged correctly among a couple of threads. The queue elegance on this module implements all of the required locking semantics.

2.5.13 soundfile

SoundFile can study and write sound files. File studying/writing is supported via libsndfile, that is a free, cross-platform, open-source (LGPL) library for studying and writing many specific sampled sound document codecs that runs on many structures consisting of Windows, OS X, and Unix. It is accessed via CFFI, that is an overseas characteristic interface for Python calling C code. CFFI is supported for CPython 2.6+, 3.x and PyPy 2.0+. SoundFile represents audio records as NumPy arrays.

2.5.14 os

In Python, the OS module provides methods for interacting with the operating system. OS is a standard Python utility module. This module implements a portable interface for accessing operating system-specific functionality. Numerous functions for interacting with the file system are included in the *os* and *os. path* modules.

2.5.15 shutil

The shutil module performs high-level operations on files, such as copying, creating, and remote operations. It is classified as a standard utility module in Python. This module automates the copying and deleting of files and directories. This article will teach us about this module.

2.5.16 time

It provides functionality such as obtaining the current time, pausing the program's execution, and so on.

2.5.17 spotipy

Spotipy is a small Python library for accessing the Spotify Web API. Spotipy gives you complete access to the Spotify platform's music data.

2.5.18 web browser

The web browser module, according to Python's standard documentation, provides a high-level interface for displaying Web-based documents to users.

2.5.19 bert-extractive-summarizer

This tool performs extractive summarizations using the HuggingFace Pytorch transformers library. This is accomplished by first embedding the sentences and then running a clustering algorithm to determine which sentences are most closely associated with the cluster's centroids.

2.5.20 pydub

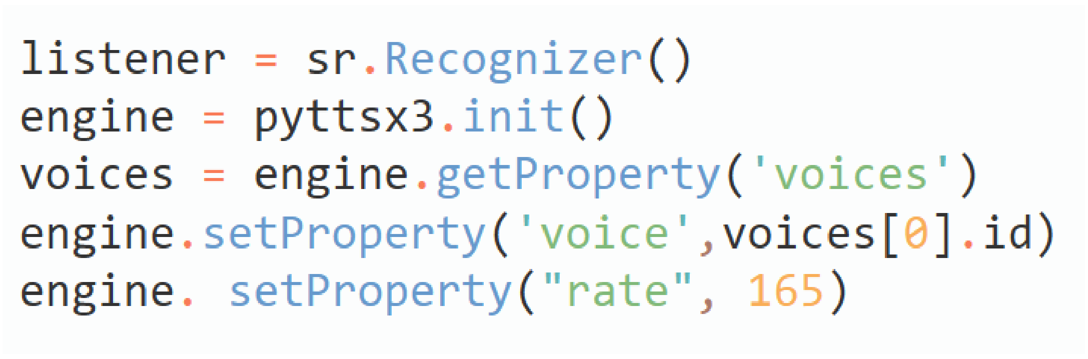
It is used to manipulate audio via a high-level interface that is simple to use.

2.5.21 bs4

This is a dummy package maintained by the Beautiful Soup developer to prevent name squatting. PyPI's Beautiful Soup Python package is officially named beautifulsoup4. This package ensures that when bs4 is installed, Beautiful Soup is created.

For more details, see [4].

Chapter 3 Working of Mitra



```
listener = sr.Recognizer()
engine = pyttsx3.init()
voices = engine.getProperty('voices')
engine.setProperty('voice',voices[0].id)
engine. setProperty("rate", 165)
```

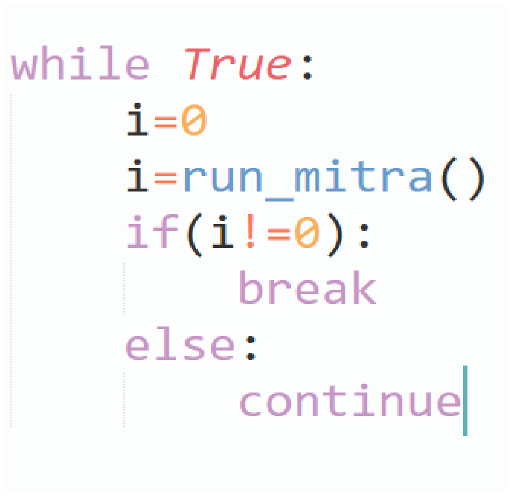
1) Importing Speech Recognition Package as sr and calling the Recognizer () function in it for capturing the input.

2) Initializing the python speech to text function.

3) Setting the personal voice assistant voice id as 0 i.e. a Male Voice.

4) Setting the Assistants rate at which it speaks.

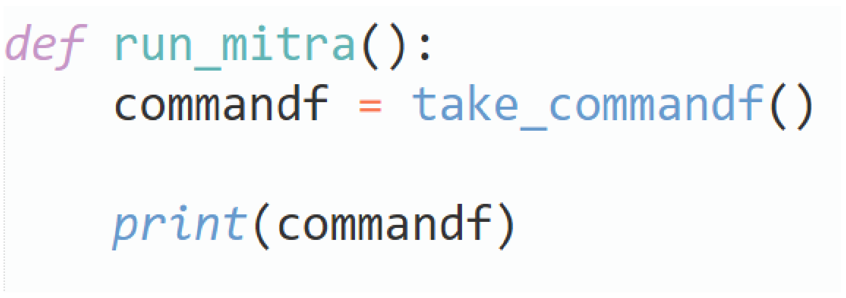
When we run Mitra:



```
while True:
    i=0
    i=run_mitra()
    if(i!=0):
        break
    else:
        continue
```

It goes in a continuous loop until the command sleep is given to it when the program stops.

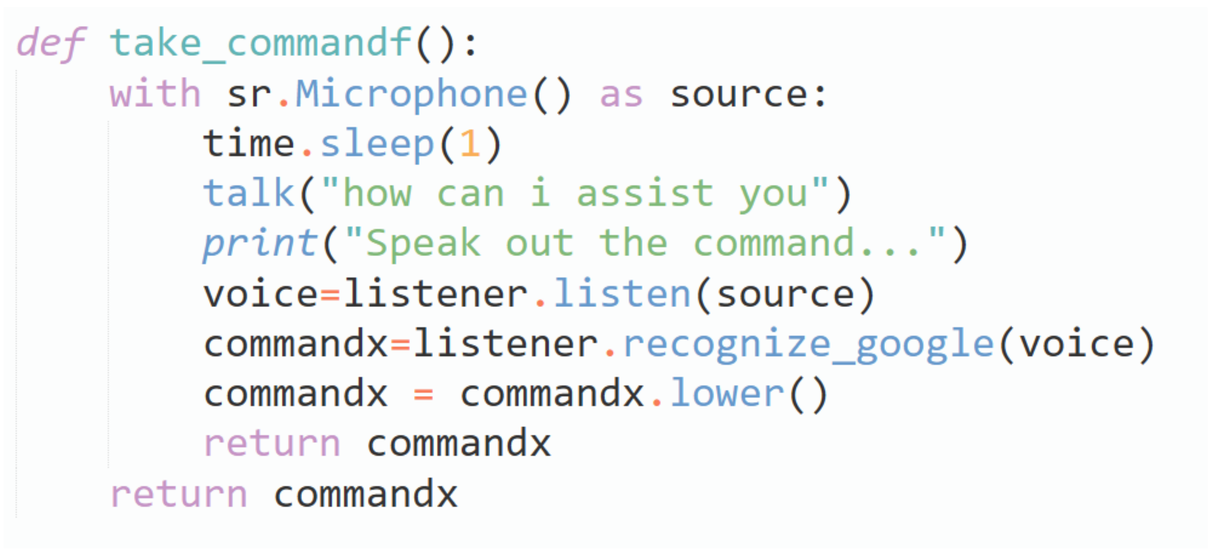
Basing on the command the user gives, Mitra functions Accordingly



```python
def run_mitra():
    commandf = take_commandf()

    print(commandf)
```

It calls the function take_commandf () function and collects the command in the form of a voice input and flows accordingly.

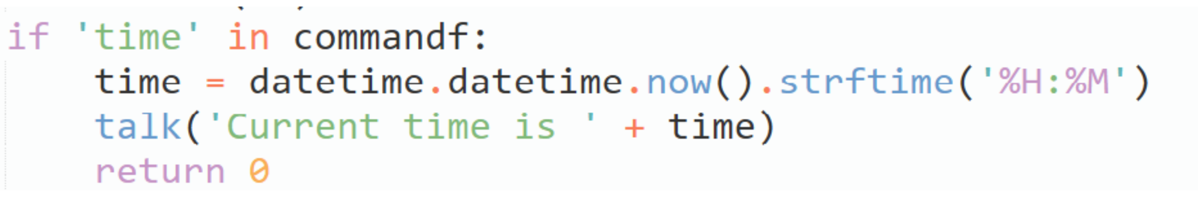


```python
def take_commandf():
    with sr.Microphone() as source:
        time.sleep(1)
        talk("how can i assist you")
        print("Speak out the command...")
        voice=listener.listen(source)
        commandx=listener.recognize_google(voice)
        commandx = commandx.lower()
        return commandx
    return commandx
```

After the command is collected it searches for the if statements in which there is a particular keyword in the command. If there is a keyword called time

Command Examples.

1. What is the <u>time</u> Mitra?
2. Can you tell the <u>time?</u>
3. Mitra what <u>time</u> is it right now
4. Hey Mitra can you tell me what is the <u>time</u> etc.



```python
if 'time' in commandf:
    time = datetime.datetime.now().strftime('%H:%M')
    talk('Current time is ' + time)
    return 0
```

It gets executed speaking out the time and returns the value 0 for the loop to continue. The if condition here uses the package date time and procures the current time and speaks out using talk function.

Talk Function…

```python
def talk(text):
    engine.say(text)
    engine.runAndWait()
```

Speaks out the input that is provided.

3.1 Working of Each Feature or Command of Mitra

3.1.1 Joke

```python
if 'joke' in commandf:
    talk(pyjokes.get_joke())
    return 0
```

If the command contains the keyword joke it will use the package pyjokes to fetch a joke a speaks out.

3.1.2 Find

```python
if 'find' in commandf:
    person = commandf.replace('find','')
    info = wikipedia.summary(person,1)
    talk(info)
    return 0
```

If the command contains find keyword. It will replace the word find and takes the word post find and searches in the Wikipedia module.

Example: Find Lata Mangeshkar is the command. Find gets replaced with Lata Mangeshkar and will be searched in Wikipedia. The information will be stored in a variable and will be spoken out.

3.1.3 time

```
if 'time' in commandf:
    time = datetime.datetime.now().strftime('%H:%M')
    talk('Current time is ' + time)
    return 0
```

If the command has the keyword time, the datetime package is used and current time is fetched and spoken out.

3.1.4 sleep

```
if 'sleep' in commandf:
    talk("Bye i miss you")
    return(-1)
```

If the command contains keyword sleep, then the loop breaks because this if condition returns -1.

```
while True:
    i=0
    i=run_mitra()
    if(i!=0):
        break
    else:
        continue
```

Integer i = - 1 and the loop breaks and the program will stop.

3.2 Web Scrapping in Python

Package used is BeautifulSoup

Beautiful Soup offers easy strategies for navigating, searching, and enhancing a parse tree in HTML, XML files. It transforms a complicated HTML file right into a tree of Python objects. It additionally routinely converts the file to Unicode, so that you don't need to reflect on consideration on encodings. This device now no longer best enables you scrape however additionally to easy the data. Beautiful Soup helps the HTML parser protected in Python's widespread library.

3.2.1 How does web scraping work

Scraping an internet web page method inquiring for unique information from a goal webpage. When you scrape a web page, the code you write sends your request to the server web website hosting the vacation spot web page. The code then downloads the web page, most effective extracting the factors of the web page described first of all with inside the crawling job.

For example, let's say we're seeking to goal information in H3 or H2 identify tags. We might write code for a scraper that appears particularly for that information. The scraper will work in 3 stages:

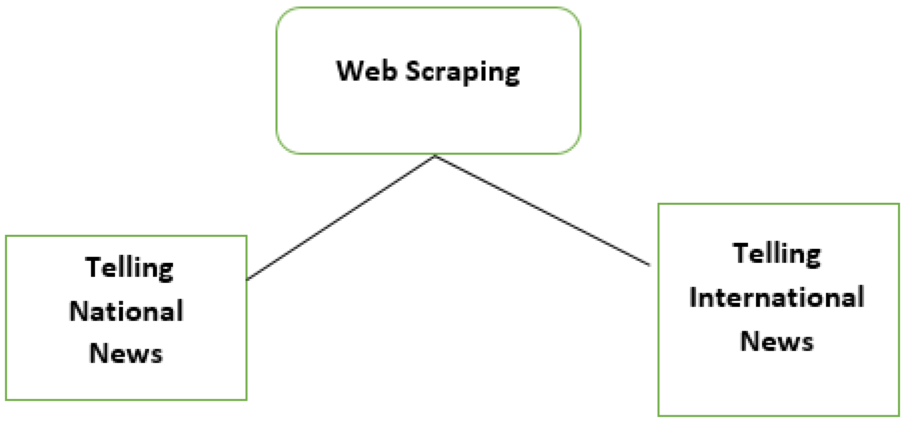
Step 1: Send a request to the server to download the site's content.

Step 2: Filter the page's HTML to look for the desired H3 or H2 tags.

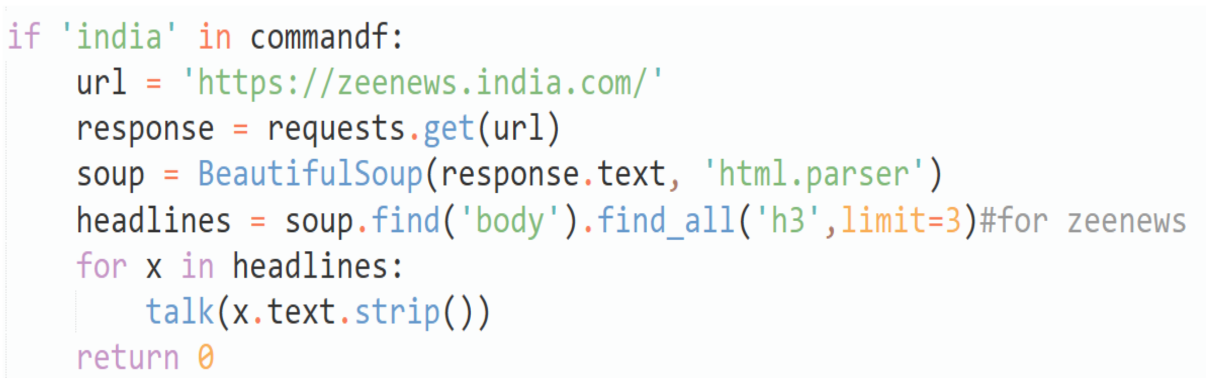
Step 3: Copying the text inside the target tags, producing the output in the format previously specified in the code.

For more details, see [5].

It is possible to carry out web scraping tasks in many programming languages with different libraries, but using Python with the Beautiful Soup library is one of the most popular and effective methods.

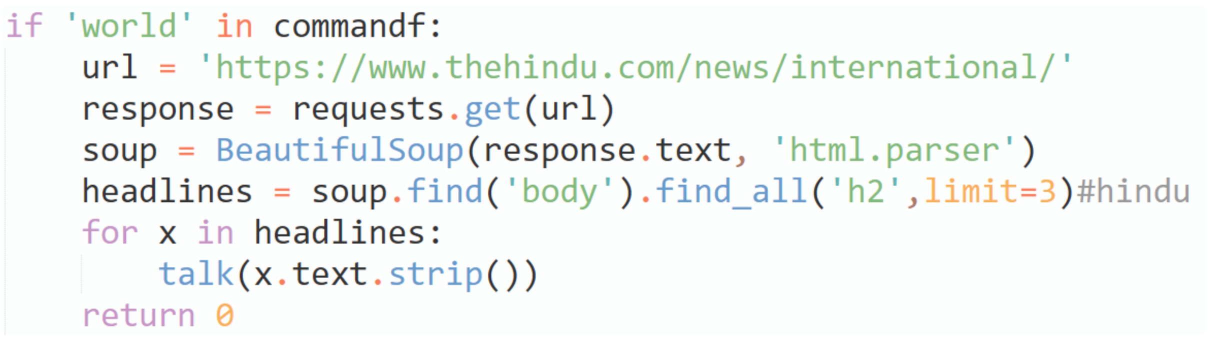


Two statements in if condition i.e., National News and International News use web scraping method in Python using BeautifulSoup Module.



```
if 'india' in commandf:
    url = 'https://zeenews.india.com/'
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')
    headlines = soup.find('body').find_all('h3',limit=3)#for zeenews
    for x in headlines:
        talk(x.text.strip())
    return 0
```

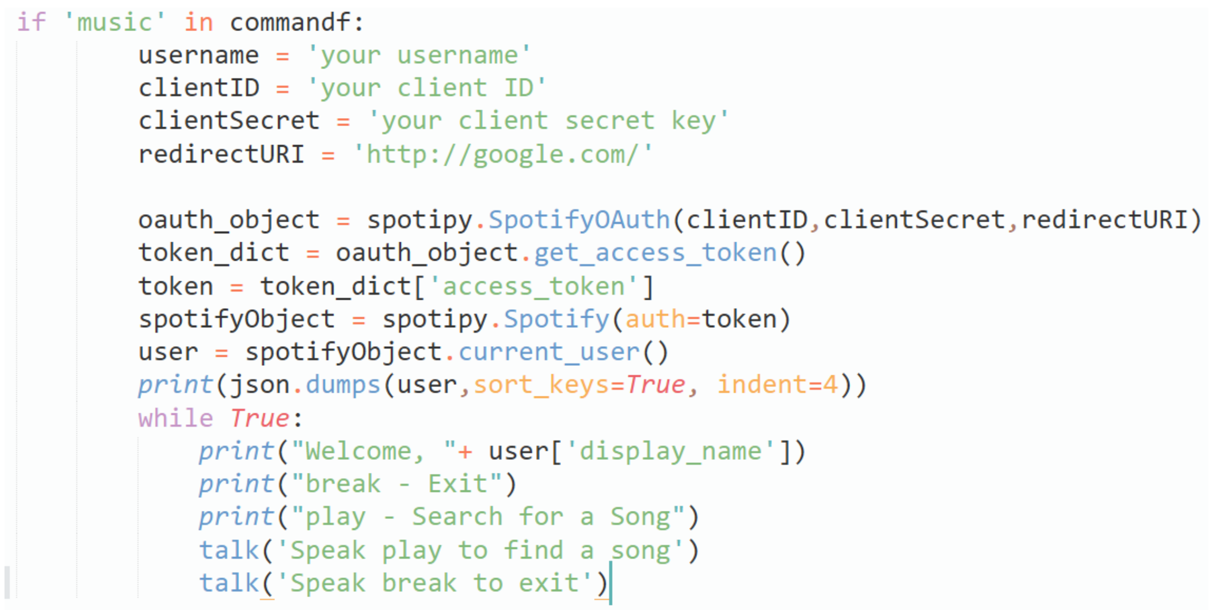
If the keyword in the command is India, then the URL is provided that is of zee news from which the headlines of header h3 will be selected and the limit is set to 3 lines. So, 3 will be selected and spoken.



```python
if 'world' in commandf:
    url = 'https://www.thehindu.com/news/international/'
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')
    headlines = soup.find('body').find_all('h2',limit=3)#hindu
    for x in headlines:
        talk(x.text.strip())
    return 0
```

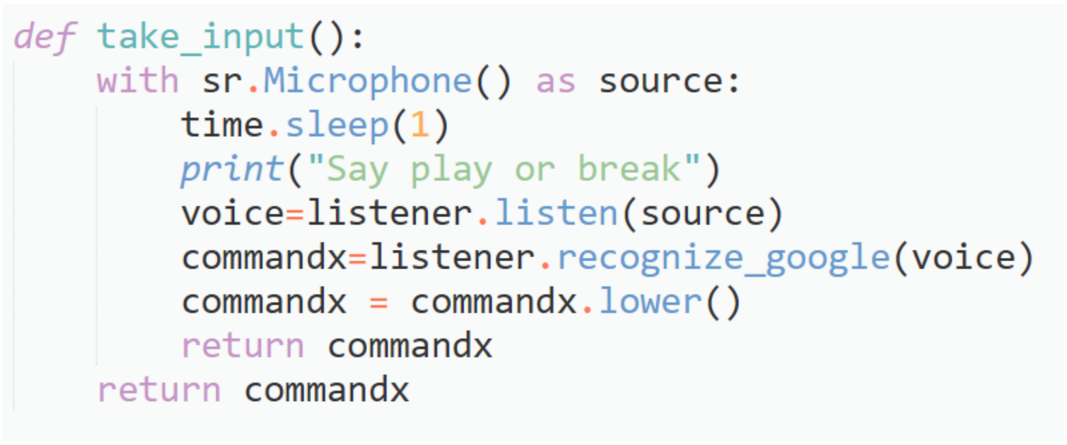
If the keyword in the command is world, then the URL is provided that is of the Hindu international from which the headlines of header h2 will be selected and the limit is set to 3 lines. So, 3 will be selected and spoken.

3.3 Spotipy



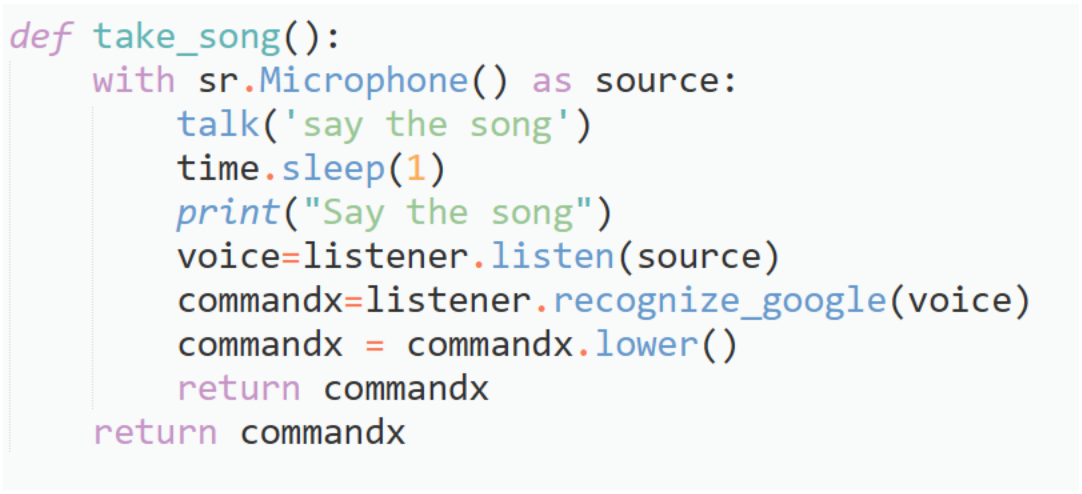
```python
if 'music' in commandf:
        username = 'your username'
        clientID = 'your client ID'
        clientSecret = 'your client secret key'
        redirectURI = 'http://google.com/'

        oauth_object = spotipy.SpotifyOAuth(clientID,clientSecret,redirectURI)
        token_dict = oauth_object.get_access_token()
        token = token_dict['access_token']
        spotifyObject = spotipy.Spotify(auth=token)
        user = spotifyObject.current_user()
        print(json.dumps(user,sort_keys=True, indent=4))
        while True:
            print("Welcome, "+ user['display_name'])
            print("break - Exit")
            print("play - Search for a Song")
            talk('Speak play to find a song')
            talk('Speak break to exit')
```



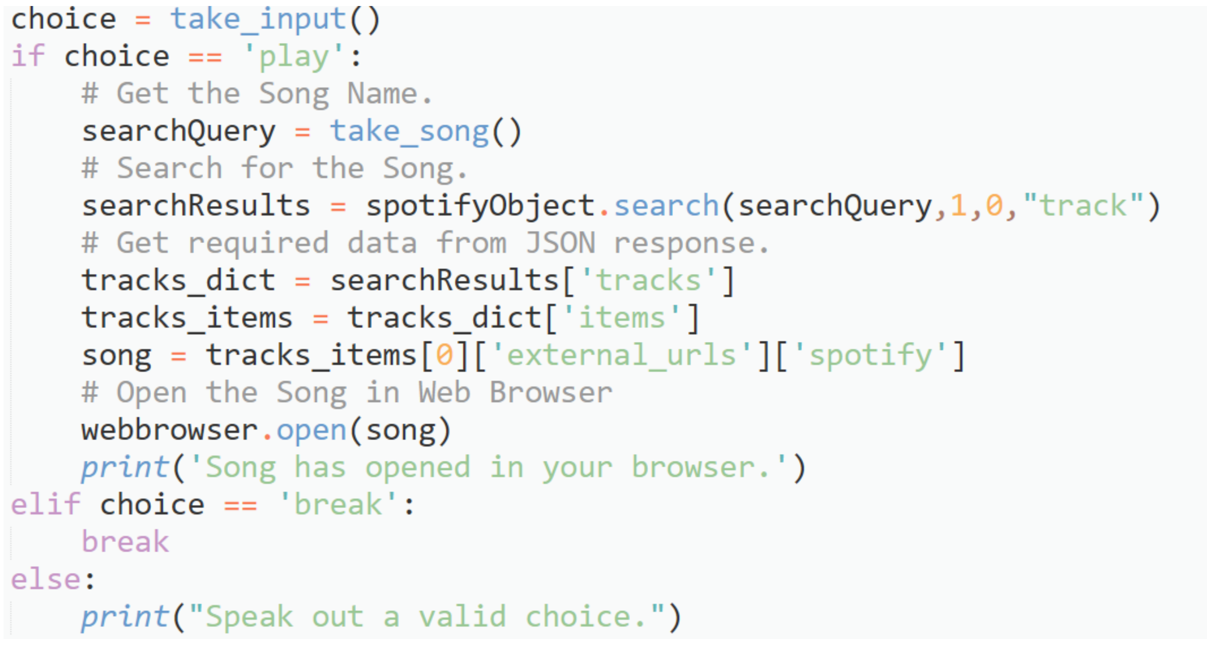
```python
def take_input():
    with sr.Microphone() as source:
        time.sleep(1)
        print("Say play or break")
        voice=listener.listen(source)
        commandx=listener.recognize_google(voice)
        commandx = commandx.lower()
        return commandx
    return commandx
```

def take input () function takes input whether to play or exit using googles Speech Recognition Package.



```python
def take_song():
    with sr.Microphone() as source:
        talk('say the song')
        time.sleep(1)
        print("Say the song")
        voice=listener.listen(source)
        commandx=listener.recognize_google(voice)
        commandx = commandx.lower()
        return commandx
    return commandx
```

def take_song () function takes input of what song the user wants to hear.



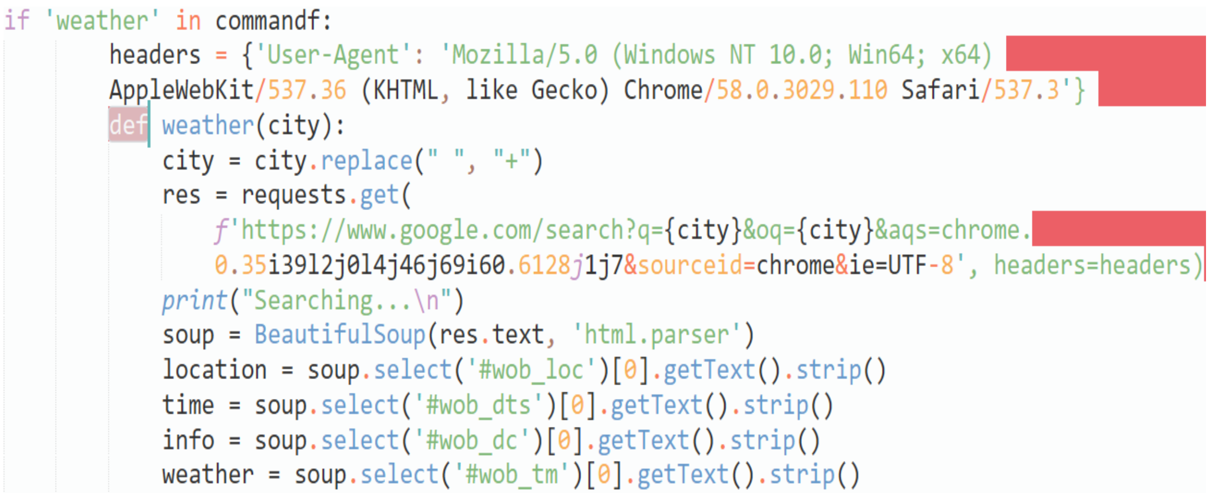
```
choice = take_input()
if choice == 'play':
    # Get the Song Name.
    searchQuery = take_song()
    # Search for the Song.
    searchResults = spotifyObject.search(searchQuery,1,0,"track")
    # Get required data from JSON response.
    tracks_dict = searchResults['tracks']
    tracks_items = tracks_dict['items']
    song = tracks_items[0]['external_urls']['spotify']
    # Open the Song in Web Browser
    webbrowser.open(song)
    print('Song has opened in your browser.')
elif choice == 'break':
    break
else:
    print("Speak out a valid choice.")
```

Based on the input and the command of the user the Spotipy opens the website which is already signed in into the username that of the account in Spotify.

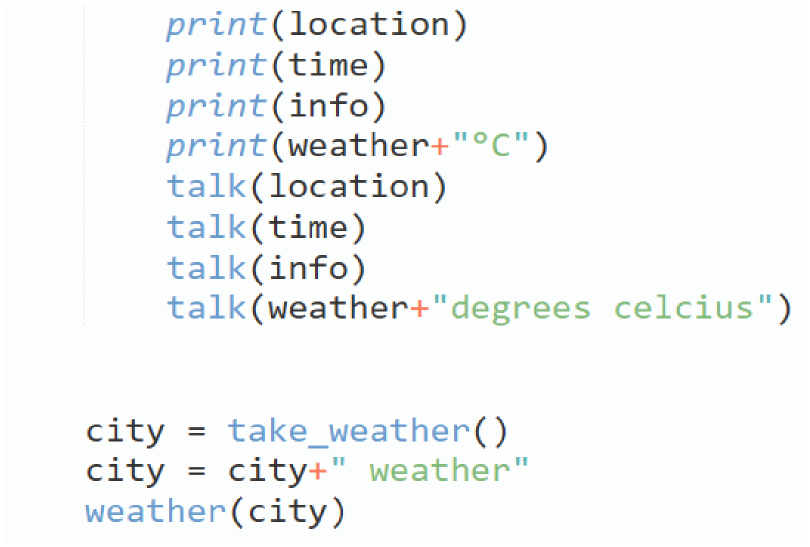
This uses the package Spotipy.

For more details, see [6].

3.4 Weather

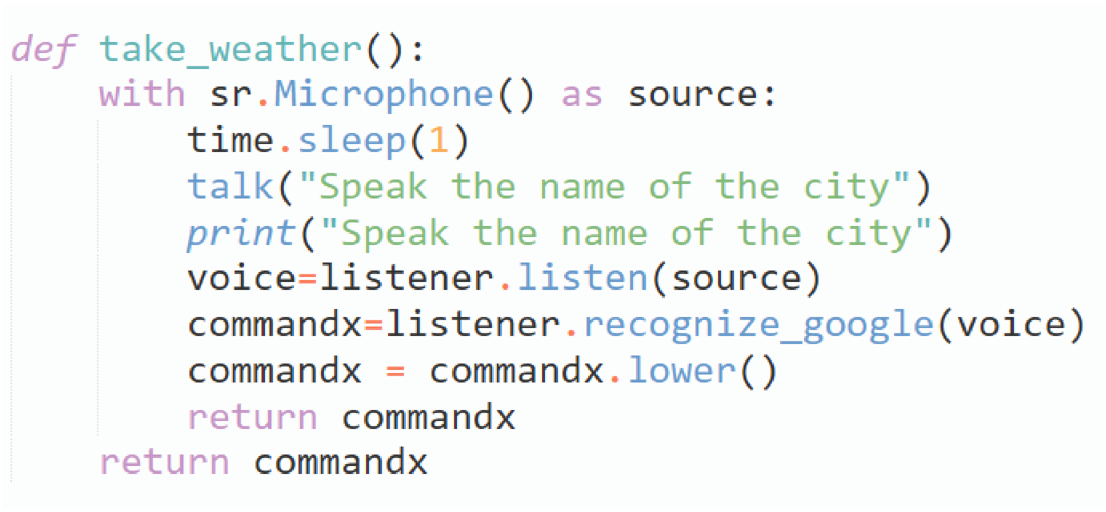


```
if 'weather' in commandf:
        headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)
        AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.3'}
        def weather(city):
            city = city.replace(" ", "+")
            res = requests.get(
                f'https://www.google.com/search?q={city}&oq={city}&aqs=chrome.
                0.35i39l2j0l4j46j69i60.6128j1j7&sourceid=chrome&ie=UTF-8', headers=headers)
            print("Searching...\n")
            soup = BeautifulSoup(res.text, 'html.parser')
            location = soup.select('#wob_loc')[0].getText().strip()
            time = soup.select('#wob_dts')[0].getText().strip()
            info = soup.select('#wob_dc')[0].getText().strip()
            weather = soup.select('#wob_tm')[0].getText().strip()
```



```
    print(location)
    print(time)
    print(info)
    print(weather+"°C")
    talk(location)
    talk(time)
    talk(info)
    talk(weather+"degrees celcius")


city = take_weather()
city = city+" weather"
weather(city)
```



```
def take_weather():
    with sr.Microphone() as source:
        time.sleep(1)
        talk("Speak the name of the city")
        print("Speak the name of the city")
        voice=listener.listen(source)
        commandx=listener.recognize_google(voice)
        commandx = commandx.lower()
        return commandx
    return commandx
```

User gives the input of the city in the above picture.

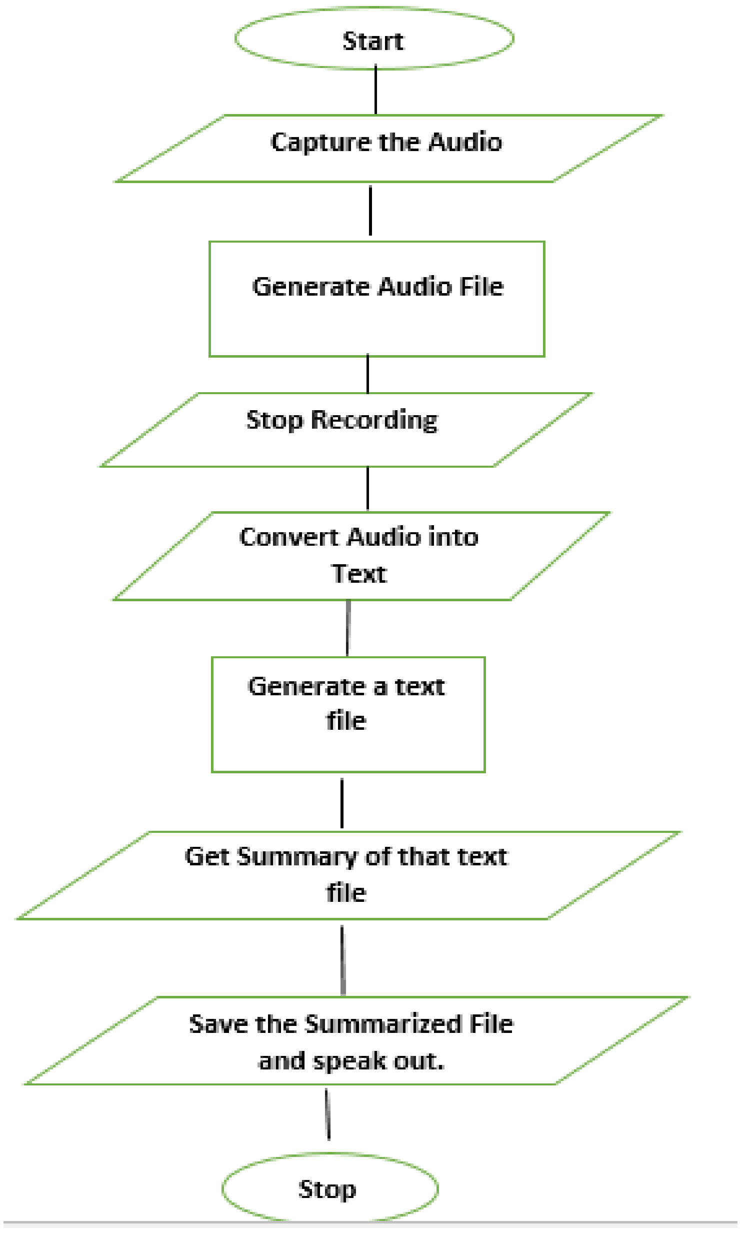
In this weather command Mitra takes the input of the city and speaks out the weather of the city with the temperature, time, location and city information.

Technique used is web scraping.

User gives the input of the city of which he wants to get the weather.

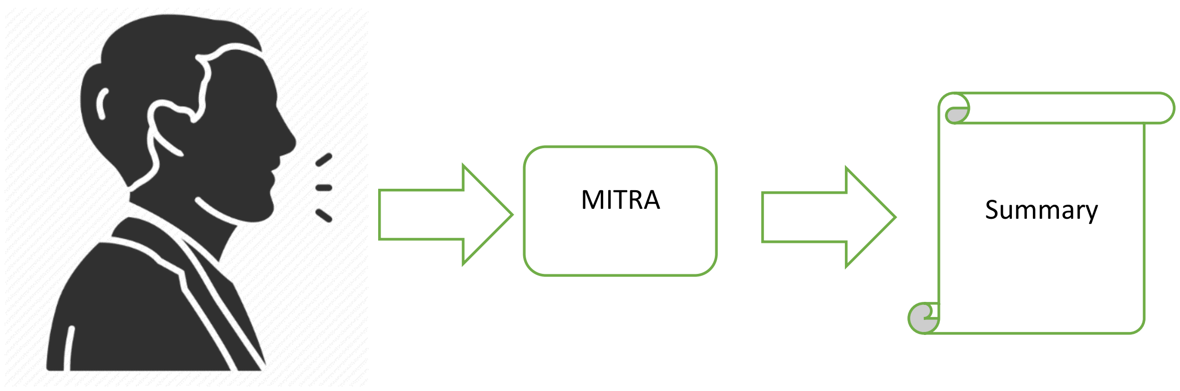
Chapter 4 Speech Summarization

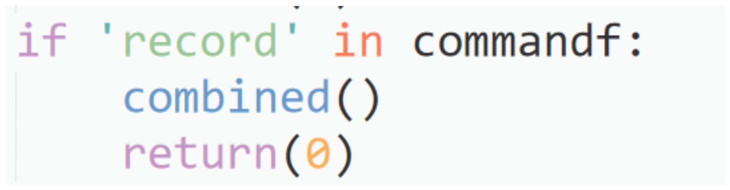
Speech Summarization is one of the most useful and interesting aspect of Mitra.



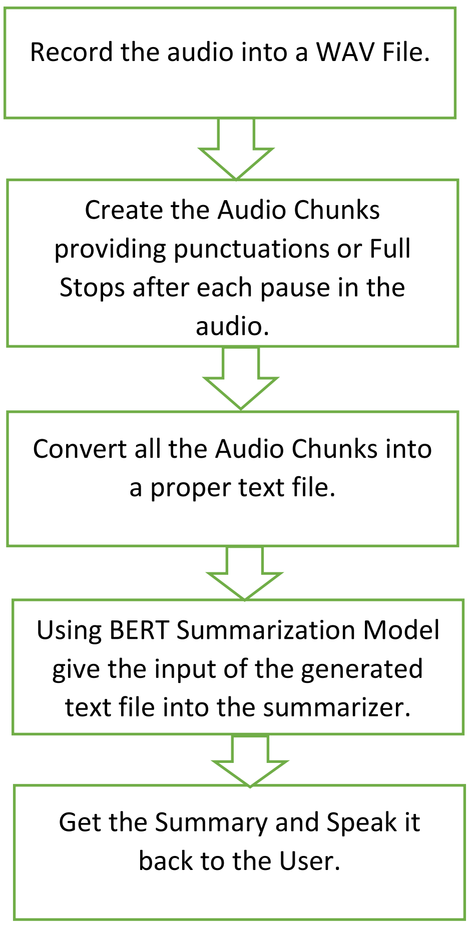
4.1 Motive

Motive for the user is to get speech summarization and Mitra will give out the summary of the speech.



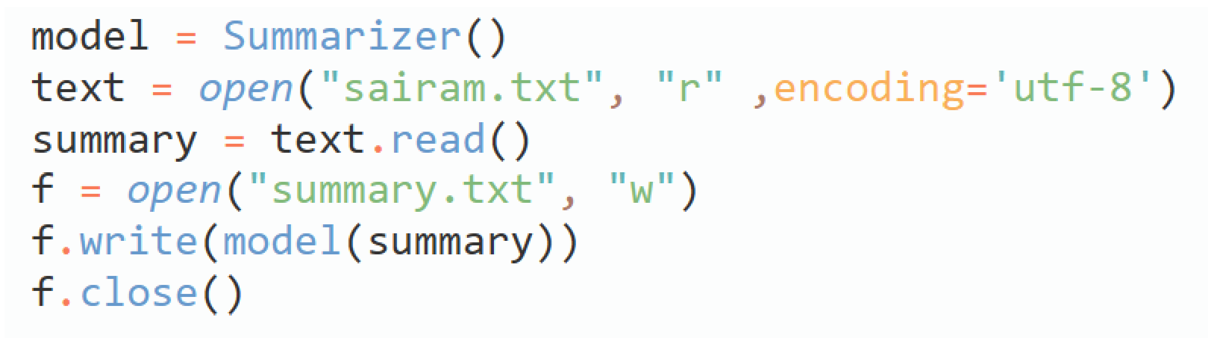

```
if 'record' in commandf:
    combined()
    return(0)
```

If there is a record keyword in the command, then the function named combined will be called and it follows the following control flow.


Record the audio into a WAV File.
Create the Audio Chunks providing punctuations or Full Stops after each pause in the audio.
Convert all the Audio Chunks into a proper text file.
Using BERT Summarization Model give the input of the generated text file into the summarizer.
Get the Summary and Speak it back to the User.

4.2 Summarization of the Text File

Package used: bert-extractive-summarizer

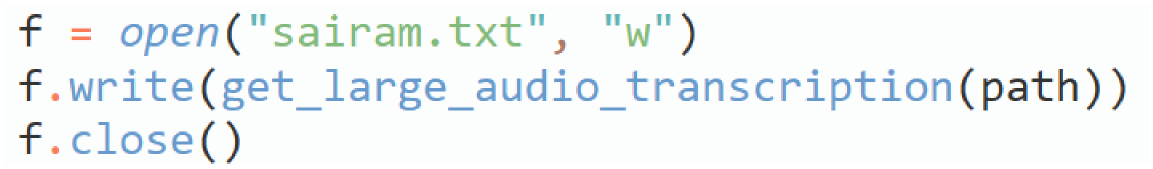


```
model = Summarizer()
text = open("sairam.txt", "r" ,encoding='utf-8')
summary = text.read()
f = open("summary.txt", "w")
f.write(model(summary))
f.close()
```

sairam.txt is the file in which the audio chunks transcription is saved. It is opened and taken in as an input into the Summarizer.

The variable summary is then passed into the model () function and summary is stored in the summary.txt file.

Model () function is from the Summarizer Package. Which is a pre trained model.



```
f = open("sairam.txt", "w")
f.write(get_large_audio_transcription(path))
f.close()
```

The sairam.txt file is written with the audio chunks transcription.

4.3 Types of Summarizations

4.3.1 Abstractive Summarization

Abstractive Summarization is a Natural Language Processing (NLP) task that entails generating a concise summary of a source text. Abstractive Summarization may generate new relevant phrases, which can be viewed as paraphrasing.

Currently, the most effective method for abstractive summarization is to use transformer models that have been fine-tuned specifically for a summarization dataset.

Transformers includes thousands of pre-trained models for tasks involving text, vision, and audio.

Transformer Models

These models can be used on:

Text, for tasks such as text classification, information extraction, question answering, summarization, translation, and text generation, in over 100 different languages.

Image classification, object detection, and segmentation are all tasks that require images.

Audio, which can be used for tasks such as speech recognition and audio classification.

Transformers is supported by the three most widely used deep learning libraries — JAX, Pytorch, and Tensor Flow — and integrates them seamlessly. It's simple to train your models on one and then load them for inference on the other.

For more details, see [7].

What is Hugging Face?

Hugging Face includes a large number of pre-trained models that have been trained on billions of text corpora and a variety of natural language processing tasks. BERT is one such pre-trained model that you may be familiar with. Additionally, there is a lighter version of BERT called DSITIL BERT.

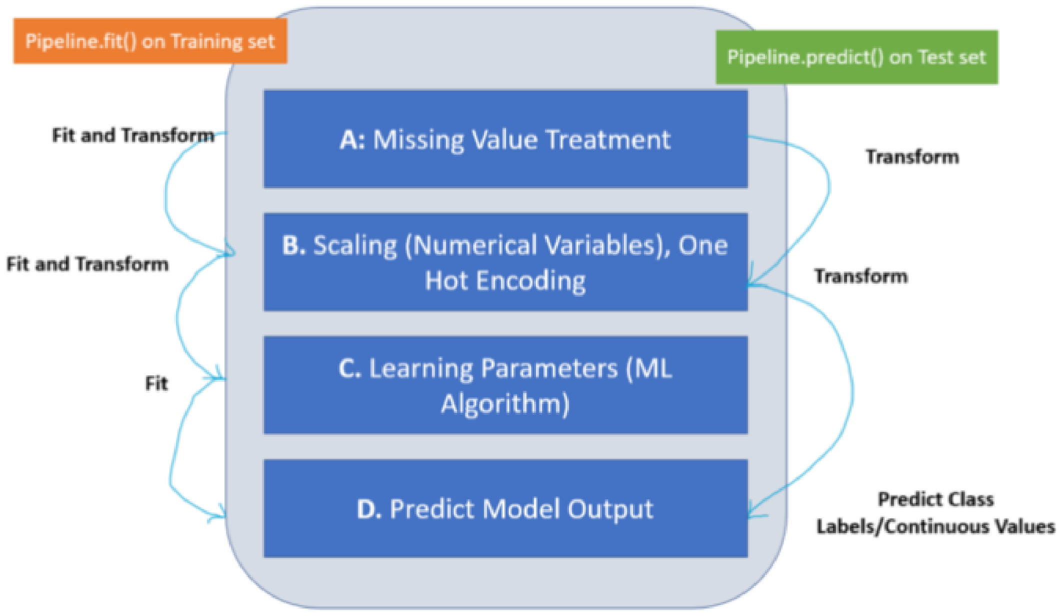
All of these Pretrained models are based on transformer architecture; some models, such as BERT, use only the encoder portion of the transformer, while others use only the decoder or encoder-decoder portion of the transformer (both).

For more details, see [8].

Pipeline

Pipeline performs all pre- and post-processing steps on the text data you provide. It performs pre-processing steps such as text conversion to numerical values and post-processing steps such as text sentiment generation, which may vary depending on the task of a pre-trained model.

Hugging face enables the use of two highly effective summarization models: BART (bart-large-cnn) and t5 (t5–small, t5–base, t5–large, t5–3b, t5–11b).



Pipeline.fit() on Training set
Pipeline.predict() on Test set
Fit and Transform
A: Missing Value Treatment
Transform
Fit and Transform
B. Scaling (Numerical Variables), One Hot Encoding
Transform
Fit
C. Learning Parameters (ML Algorithm)
D. Predict Model Output
Predict Class Labels/Continuous Values

Image source: Pipeline and Custom Transformer with a Hands-On Case Study in Python | by Angel Das | Towards Data Science

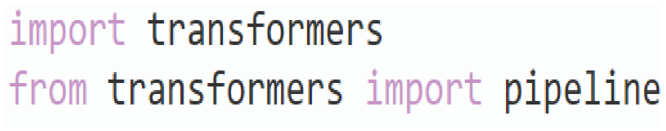
<u>How do Pipelines work?</u>

Pipelines comprise of a progression of fit and change tasks that differ among preparing and test information. Typically, fit and change together or fit work is utilized on the preparation set; change alone is utilized on the test set. The fit and change activity initially distinguish significant data about the information dissemination and afterward changes the information in view of this learned data. The fit () learns the Median Value of the relating property by concentrating on the information dispersion, though the change () replaces the missing worth with the Median gained from the information. An, e.g., includes, suppose, we are utilizing a Simple Imputer (), a python bundle for missing worth treatment, to supplant missing qualities with Median qualities.

Test information, then again, utilizes just the change activity. This is chiefly done to stay away from any information spill while building the model. The thought is to perform pre-handling on the test information in view of data gained from the preparation set than cleaning the whole information overall. Any missing data in the approaching information stream should be taken care of before it tends to be taken care of into the expectation model.

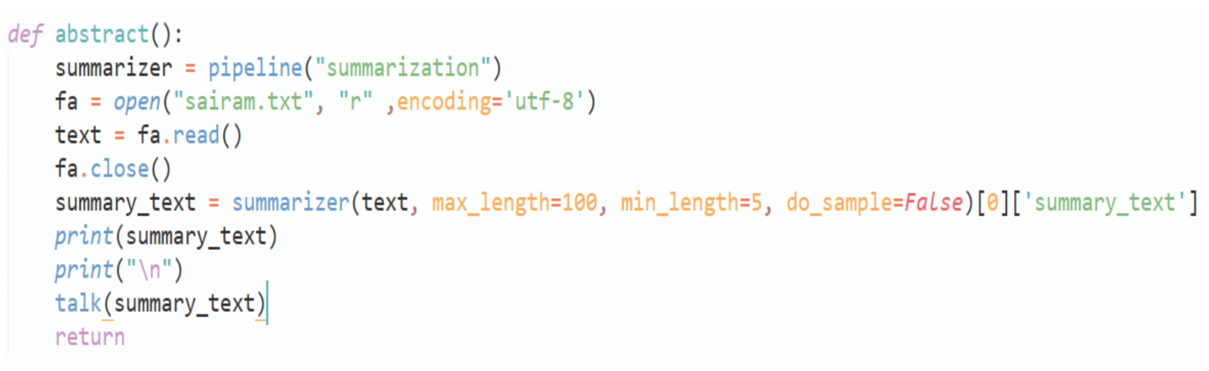
For more details, see [9].

4.3.2 Abstract Summarization using the default parameters via Hugging face's inbuilt pipeline module



```
import transformers
from transformers import pipeline
```

From the package transformers importing the pipeline module.



```
def abstract():
    summarizer = pipeline("summarization")
    fa = open("sairam.txt", "r" ,encoding='utf-8')
    text = fa.read()
    fa.close()
    summary_text = summarizer(text, max_length=100, min_length=5, do_sample=False)[0]['summary_text']
    print(summary_text)
    print("\n")
    talk(summary_text)
    return
```

Sending the input of the file in which the audio chunks got saved.

Summarizing the input text and saving the summarized part into variable summary_text and speaking out back to the user.

4.3.3 Extractive Summarization

This method summarises the text by extracting the most significant subset of the original text's sentences. As implied by the name, it extracts the most critical information from the text. Because this method is incapable of generating text on its own, the output will always contain some portion of the original text.

For more details, see [10].

4.4 Extractive Summarization with BERT

Summarization has long been a source of contention in the field of Natural Language Processing. To create a condensed version of a document while retaining its most critical information, we need a model capable of accurately extracting key points while avoiding redundant information.

Recent works in NLP such as Transformer models and language model pretraining have advanced the state-of-the-art in summarization.

BERTSUM is a simple variant of BERT, for extractive summarization from Text Summarization with Pretrained Encoders

The BERT summarizer has 2 parts: a BERT encoder and a summarization classifier.

BERT Encoder

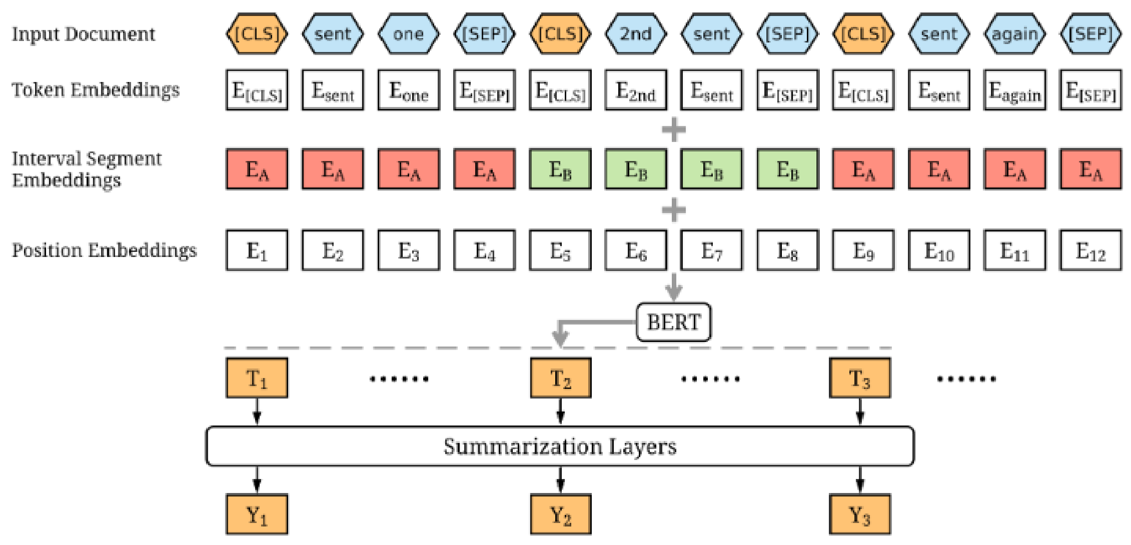


Image source: https://iq.opengenus.org/bert-for-text-summarization/

The BERT encoder is the pretrained BERT-base encoder used in the masked language modelling task. Extractive summarization is a problem of binary classification at the sentence level. Each sentence should be labelled yi 0,1yi 0,1 to indicate whether it should be included in the final summary.

For more details, see [11].

Summarization Classifier

After obtaining the vector representation of each sentence, we can use a simple feed forward layer to compute a score for each sentence. The author conducted experiments with a simple linear classifier, a Recurrent Neural

Network, and a three-layer Transformer model. The Transformer classifier produces the best results, demonstrating the critical role of inter-sentence interactions via the self-attention mechanism in selecting the most significant sentences.

For more details, see [12].

Chapter 5 Scope and Conclusion

There are many Voice Assistants in the market which perform day to day activities of human beings very interactively, Mitra is also focused in making tasks easier for humans providing them with new features and assisting them.

What more can we do?

Multithreading

In this program the command input is taken by Mitra after the previous task is completed. If there is multi-threading, then one thread can execute the program based on the input command given by the user and one thread can always be listening to the input. If there is a specific input to be given to the always listening thread, then the interruption of Mitra can happen. This feature is really good and can be implemented.

If this can be implemented, we can interrupt Mitra even when it’s in the middle of the tasks.

Making Mitra like an Interactive Voice Response System(IVRS) is more feasible and useful. More and more features can be added and through command-based input methods many other tasks can be included and executed by Mitra.

In the current market there are no voice assistants with speech summarization features. Mitra can easily help in summarizing a foyer talk, class lecture, elocution speech etc.

Providing similar features like speech summarization and many others would increase the competition in the market among other voice assistants.

5.1 Use Cases

In the institute, for getting summary of the foyer talks on Wednesday and Friday and any sort of speech or a communication.

This product can be launched with appropriate AI and much sophisticated methods and can really perform well in the market like other voice assistants.

Mitra is a command driven personalized voice assistant which can assist you as friend and do mundane tasks, this can be upgraded and much more features and tools can come into play.

Bibliography

1. The comprehensive guide to voice assistants:

   (https://www.slanglabs.in/voice-assistants#:~:text=can%20divide%20them.%3A-,General%20Purpose%20Voice%20Assistants,speakers%20and%20other%20smart%20devices).

2. Most popular digital helpers:

   (https://www.diysmarthomesolutions.com/voice-assistant-guide-digital-helper/)

3. How voice assistants play role in daily life:

   (https://www.smartsheet.com/voice-assistants-artificial-intelligence)

   https://www.slanglabs.in/voice-assistants

4. Python packages:

   (https://pypi.org/)

5. Geeks For Geeks Web Scraping:

   (https://www.geeksforgeeks.org/implementing-web-scraping-python-beautiful-soup/)

6. About Spotify API using python:

   (https://towardsdatascience.com/extracting-song-data-from-the-spotify-api-using-python-b1e79388d50)

   (https://pypi.org/project/spotipy/)

7. About Transformers Package in python and text summarization:

   (https://www.thepythoncode.com/article/text-summarization-using-huggingface-transformers-python)

   (https://turbolab.in/types-of-text-summarization-extractive-and-abstractive-summarization-basics/)

8. Hugging Face:

   (https://huggingface.co/docs/transformers/index)

9. About Abstractive Summarization:

(https://towardsdatascience.com/abstractive-summarization-using-pytorch-f5063e67510)

10. About Extractive Summarization:

(https://towardsdatascience.com/extractive-summarization-using-bert-966e912f4142)

11. About BERT models for text summarization:

(https://iq.opengenus.org/bert-for-text-summarization/)

12. Pipeline module in Transformers for abstract summarization:

(https://medium.com/analytics-vidhya/hugging-face-transformers-how-to-use-pipelines-10775aa3db7e)

For Audio chunks and audio to text conversion refer to B.R Sricharan https://github.com/charan-debug