VIRTUAL VOICE ASSISTANT IN PYTHON

Abstract :

Speech recognition and speech synthesis are two main fields in NLP which have a large scope in the real life. Communicating to our devices through voice has already started and it would soon be a primary input of the device rather than text, which is being used now. Most of the NLP problems and challenges are still open and are not solved yet. Coming to the voice assistant, even the world's best assistants available to people such as google assistant, cortona and siri have limited functionality. Most of them when asked any information would redirect to a webpage but this is not what a complete assistant does. It should instantly give an accurate and precise answer to the user, and if the information provided isn't satisfied by the user, then it should be redirected to a related webpage. So, in this project we are going to incorporate many functions and implement algorithms so that we create an efficient and powerful voice assistant.

Keywords – NLP, speech recognition, speech synthesis, regular expressions, web scrapping, threads, processes, webpage, HTML, requests, etc...

Introduction:

A voice assistant has 5 major steps.

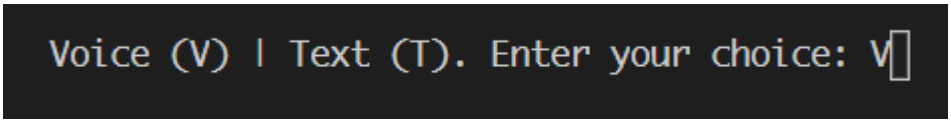
1. Take Input [either through voice or text]
2. If input is voice, convert audio to text.
3. Process the command (get the meaning of that command).
4. Redirect the command to related function
5. Get the related information for the answer and reply by either text or voice.

All of these are equally important for a voice assistant. Every assistant can take an input and give output. But here

what matters is how well the input or command is understood and how powerful enough is the algorithm so as to generate the desired answer or output. Precisely speaking, a virtual voice assistant should talk to us in the same way as humans and also should be able to answer any question asked in its domain. So let's see how we can efficiently build a simple and powerful voice assistant using python tools and libraries.

The Voice Assistant

1. Input of the assistant:
   This virtual assistant has 2 inputs; voice and text. User can choose between any of these and get the desired output.



2. Speech Recognition:
   If voice is chosen as an input to communicate to the assistant, with the default microphone of the device as source, we listen to the audio. Once any text is recognized in the audio, it will stop listening and the command is processed. If the source does not recognize any audio, it will raise an exception.

   So the speech recognition is achieved by using a python library called *speech_recognition.* This module contains multiple APIs for speech recognition from google, bing, etc.

```
# CONVERT AUDIO (SPEECH) TO TEXT
try:
    text=r.recognize_google(audio)
except:
    text='Sorry I did not get that'
```

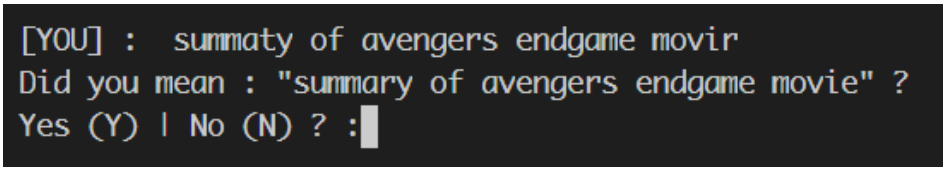
We have used google API for speech recognition in this project. It performs speech recognition on audio data using the Google Speech Recognition API. The Google Speech Recognition API key is specified by a key. If not specified, it uses a generic key that works out of the box.

3. Command Processing

After the audio is converted to text or after speech recognition is done; the main task is to figure out these things;

- Is the entered command valid or correct
- What is the meaning of the command?
- What type or class does the command belong to?
- From where should we get the data?
- What data is to be displayed or shown as output?

Auto Correction Feature: First, if the command is entered but has any spelling mistakes, there appears a prompt with corrected command. The user always has an option to proceed with either original command or with the corrected command.



This is achieved by using *spellchecker* library where the minimum edit distance of words are calculated and the closest word is chosen as the correct word.

```
corrected_text=""
for word in text.split(' '):
    corrected_text+=(spell.correction(word)+ " ")
```

Regular Expressions: One powerful tool for matching sentences to required expressions is regular expressions. Regular expressions can handle great amount of logics in a single line and match the command to a desired expression. So we basically do this;

```
if command match regex1:
    give output1;
elif command match regex2:
    give output2
```

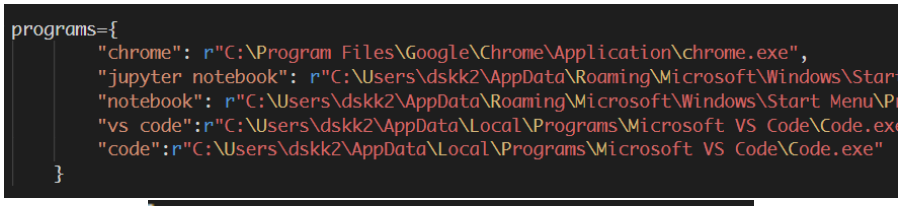
This is the most simple but also powerful way for processing the command.

4. Features of Voice Assistant

This voice assistant is incorporated with many features functions.

- **Local Features :**
  - Play songs: Choose songs from local directory and play. Song can be stopped anytime. The music will be played in an asynchronous thread and this doesn't cause any interruption to the current execution of the program. The music played can be stopped by asking the assistant to stop the music.

  - Open any program in computer: The path of the programs can be added to the *programs* dictionary in util.py file to open any of the program.

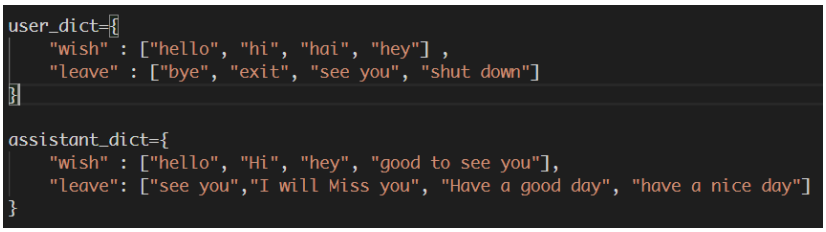



```
os.startfile(programs[prg])
```

Here *prg* is the key of the dictionary *programs.*

  - Shutdown device

- **Intelligent replies:**
  - Wishes or greets with a random message to the user if command is related to wishing.
  - If the user is leaving or exiting from assistant; it displays a random exit message.



```
user_dict={
    "wish" : ["hello", "hi", "hai", "hey"] ,
    "leave" : ["bye", "exit", "see you", "shut down"]
}

assistant_dict={
    "wish" : ["hello", "Hi", "hey", "good to see you"],
    "leave": ["see you","I will Miss you", "Have a good day", "have a nice day"]
}
```

If user enters anything from *user_dict*, a random message from *assistant_dict is outputted*

  - If user is bored or wants to hear a joke, the assistant tells a joke instantly.

- **Web Scrapping**

  For most of the time, we need real time data from internet such as price of a product, stock price or rating of a movie. We can achieve this by scraping desired websites and create an API that requests the page to give the source and then we extract the exact information we want. Below are some APIs created that can extract information from a specific website.

  - Price of any product from Amazon or Flipkart.
  - Summary (Rating, gen. movie info, plot) of any movie from IMDb.
  - Real time stock price from Yahoo finance
  - Answers from Stack Over Flow website.

  Many other such APIs can be created for more functionality.

  This is achieved by a python library BeautifulSoup. It helps to extract information from clumsy html source code of the page.

- **Math Answers**

  If the command or input contains maths such as math keywords, symbols, numbers, etc. it is classified as a math question. These questions are redirected to WolphramAlpha Api which can answer basic additions

to complex mathematical problems such as integration and differenciation.

- **Open relevant webpages and search web**
  If user asks to open any program or webpage, it first searches in the local program files mentioned in the code, if none of them matches the most relevant webpage is opened by choosing the first link of the google search using the library *googlesearch.*

  If the given command has no direct answer to output, it can redirect us to a google search page with the command already searched.

5. **Reply from Assistant**

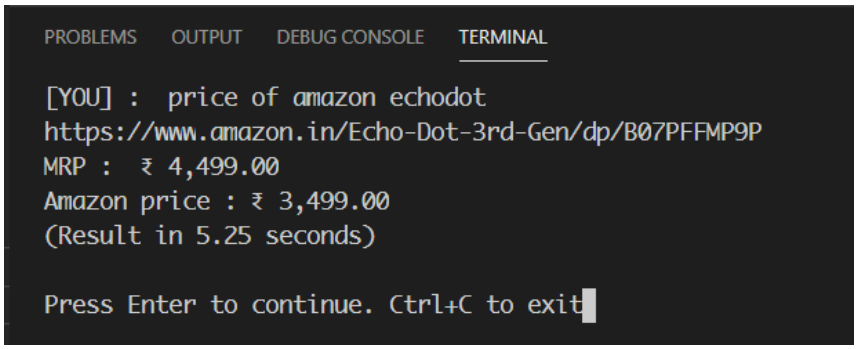
Primary mode of reply by this virtual voice assistant is speech (voice). The text is again converted to speech by a python library called pyttsx3. It converts the text response from the voice assistant and speak it out through the default speaker.

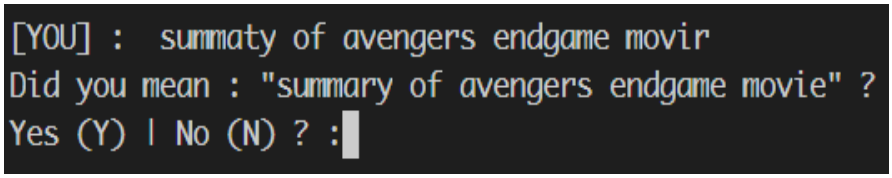
This module can efficiently convert text to speech and also provides additional features like setting the gender, age and rate of speech of the voice. But this module has a problem. The speak() method of this pyttsx3 is a separate thread but this thread blocks the whole process while it is speaking. So if it has to speak 3 to 4 lines, for that much time no other function can be performed.

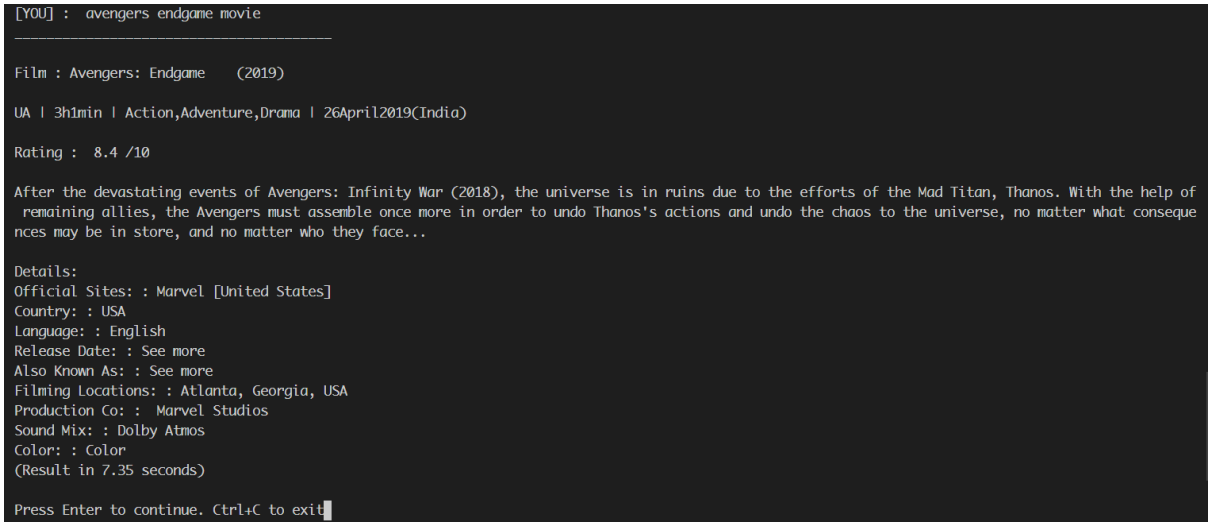
The solution for this is to execute the speak function in separate file as a separate process with command line argument.

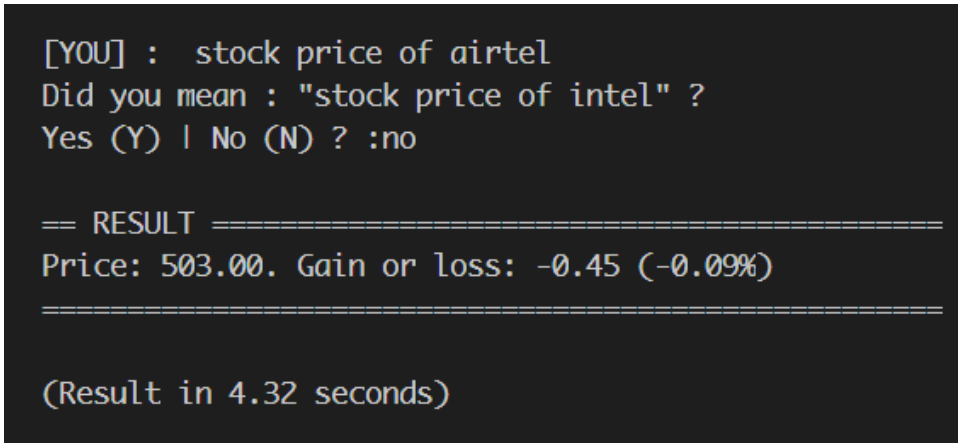
```
def speak(phrase):
    call(["python", r"C:\Python Projects\Normal\NLP_Project\speak.py", phrase])
```

Assistant in action: See few code of snippets for knowing what this assistant can do:









There are many other functions in this program but, it can be seen in live as it deals with voice.

<><>====== <<< THE END >>> =====<><>