DSPBot

Advanced Personal Voice Assistant

DSP_FinalProject_Fall2019

Submitted By:

Harshul Balani - hb1636

Parnika Bhosale - pb2258

Utkarsh Saboo - us452

Introduction

Our project, 'DSPBot: Advanced Personal Voice Assistant' is an advanced implementation of a voice assistant similar to 'Alexa', 'Siri' etc. This project has the possibility of vast area for modifications and advancements. The main idea of the project is o basically get our tasks done by just taking human speech as the input.
You can do any of the following like:

- playing music
- opening any system application
- search anything on Google
- check any location or directions on map
- voice recording
- to calculate something
- to check the time.

Also if you want to have a good talk with the bot, it can be done too! By just inputting more jokes or scripts that you'd like.
Thus, this project has the capability of involving in daily lives of the people and making things easier for them.

Execution, Implementation and Applications

We have created this bot implementing multiple python modules, including what we have learnt through class, to bring more creativity with learning, into this project. The project is using Google speech recognition API, i.e., 'speech_recognition' in order to implement the speech recognition for converting input user voice to text. And 'gTTS(Google TTS)' library is used to convert this text to speech, which will be used when the Bot is interacting with the user.

The bot can search anything the user would want on Google, can play any song on YouTube, can search a given location on the Google Maps for you. To make these tasks happen, we used 'webbrowser' module, which helps in triggering the browser, mentioned in argument when implementing the module, to open in a new tab by default. For e.g., "webbrowser.get('safari')".
Following scripts are useful to execute the tasks:
Please note that it is preferable to use your initials when the DSP_Bot asks your name.

1) "Search/Play '//Song Name// ' on youtube". -For Songs
2) "Search '//weather in NYC/Anything//' on Google". -For searching anything on Google
3) "Search '//Location Name//' on maps". -For searching of any location on Google Maps

Additionally, one can open any of the applications present on any of the system like MAC or Windows. In order to implement this, the Python module "sys" has been used, which is used to provide functions and variables that can be utilized to manipulate different parts of the Python runtime environment. The "os" module has been used to interface with the operating system and to perform certain OS functionalities through Python. One can create/delete files/folders, open an application, change the working directory, etc.

Another module, “subprocess” also has been used that is very useful and used to start new applications from the program and can implement other shell commands using this module.
Some of the applications that one can open using the scripts we have given are:

1) Open “iTunes”
2) Open “Chrome”
3) Open “Firefox”
4) Open “windows media player”
5) Open “iTunes”
6) Open “recorder” (This voice recorder we have designed using TKInter)

User can also add more functionalities based on the applications/tasks that he/she wants to open or to get executed through this bot.

The way the audio processing in the project that takes place is as follows:
The main function takes input from the user initially to get the name of the user. Then it asks what tasks it can perform. Once the main function get the input from the get_audio().lower(), it passes the call to the input_audio function. The input_audio() function describes can perform various tasks:

1) Tell what all functions the DSP_Bot can perform
2) Answer questions like “What are you”, “Who built you”, “What is the time”
3) It can open applications
4) Search the web
5) Do difficult calculations for us

The input_audio function also handles ambiguities. For example if the DSP_Bot does not understand what the user has said, it gives us an option to search the web for us.
The “wolframalpha” is the library that is used for the calculation and timing functions. We import ‘wolframalpha’ and it’s dependencies. It requires python version of 2.7 and greater. t is a special API. Here we create the client with App ID which is requested from Wolfram Alpha. Then we send queries, which are returned in result objects. Now these results are obtained from ‘WolframAlpha’ by doing dynamic computations based on vast collection of built in data, algorithms and methods.

The usage of ‘wolframalpha’ library in our project can be explained as:

First, create the App ID which is requested from Wolfram Alpha:

```
import wolframalpha
client = wolframalpha.Client(app_id)
```

Then, Send queries, which return Result objects:

```
for pod in res.pods:
    do_something_with(pod)
```

```
print(next(res.results).text)
```

Now, query can be used for the pods which have ‘Result’ as titles or are marked as ‘primary’ using ‘result.results’.

Furthermore, we have also tried to integrate some of the things we learned in class into our project. So, we have integrated TKInter and PyAudio into the DSP Bot.

We have used PyAudio to take in the 16-bit audio input. So, whatever the user is speaking is taken in as a PyAudio object whose output (which is the same as the input because you’re not modifying it at all) is getting stored in the audio file “speak.wav”. The specifications we have given to the PyAudio object are as follows - the length of each sample is 16 bits (paInt16), we’re using 1 channel and the sampling rate is 16000 samples per second. This is in-turn being used as the audio file source for Google’s speech recognition module.

Next we have used TKInter in our audio record function that we provide to the user. The audio recorder is a function that we have created and which we can open by saying, “open recorder”.
This opens the recorder application, which uses TkInter to create an audio recording widget, having the buttons “start” and “stop” on it. On clicking the start button, we can begin the recording and we click on the stop button to stop recording. We give the user the option to give it whatever name they wish to the audio recording and it is stored in the directory where the user runs the python program as a .wav file.
In order to check in which directory the .wav file is saved, we can enter the command ‘pwd’.

Future Scope

The scope for this is endless. We can add as much functionality as we want to the DSP bot.
For example - the easiest addition would be to add more search websites that we can open using the voice command - for example we can open websites like wikipedia using the command - “search on wikipedia”, or add the functionality to “FaceTime” someone etc. We created the function to record voice, which we open by saying “open recorder”. So we can add other functions in our TKInter based recorder module to modify our voice - adding effects such as flanger, vibrato, filtering etc.

References

1) OS Python Documentation: https://docs.python.org/3/library/os.html
2) Sys Python Documentation: https://docs.python.org/2/library/sys.html
3) Subprocess module: https://www.pythonforbeginners.com/os/subprocess-for-system-administrators
4) SpeechRecognition module: https://pypi.org/project/SpeechRecognition/
5) Tkinter module: https://docs.python.org/3/library/tk.html
6) PyAudio Library: https://pypi.org/project/PyAudio/
7) Wolframalpha: https://www.wolframalpha.com/
8) Wolframalpha library: https://pypi.org/project/wolframalpha/