Sbotify

Ruben Teimas, ***m47753***
Universidade de Évora

November 15, 2020

1 Introduction

In the first assignment of *Sistemas Multimodais* we were asked to develop a new application or interaction model (*Text-To-Speech*, *Speech-Recognition*, *Handwriting-Recognition*) to an existing game/application.

The application chosen to develop was a simple bot to control *Spotify* using voice commands. The reason behind this choice is the need for me to skip/play songs while playing videogames without taking my hands off the gamepad.

The application uses 2 out of the 3 types of interaction. It uses *Speech-Recognition* for the user to give commands to the bot, and *Text-To-Speech* for the bot to respond to the user.

2 Tools

The most interesting part of this project was the research made in order to find the best tool to create said types of interactions and understand the way they work.

2.1 Text-To-Speech

For the *Text-to-speech* interaction a list of tools was gathered, such as: ***Mozilla TTS***, ***IBM's Watson Text-to-speech*** and ***gTTS***.

To prototype the application it was used ***gTTS***, as it is a simple module that makes calls to *Google*'s API. It is, without a doubt, a very useful tool, but not the one suited for this project as the main focus is to go more in depth in how *text-to-speech* can be achieved, used techniques and core concepts.

Mozilla TTS uses ***Tacotron 2*** (Natural TTS Synthesis module). For this reason it synthesizes the text to a very fluent speech.
I've read the documentation and got to understand how the process works (even though I had some difficulties due to lack of knowledge in *deep learning*) but I could not build a model and train it.

In order to do more than using a simple API (since there was no luck with ***Mozilla TTS***) the ***IBM's Watson Text-to-speech*** tool was explored, using the *Lite* plan. Although it was not found an option to synthesize speech, it was found in the documentation ***the science behind the service***.

In the end, the tool used for the *tts* interaction was the ***IBM's Watson Text-to-speech*** because even though was similar to the ***gTTS*** it seemed to have a more fluent speech.

2.2 Speech-Recognition

Throughout the lessons we were presented with some tools with *Speech-Recognition* capabilities. I focused primarily in ***CMU Sphinx***, ***SpeechRecognition*** and ***IBM's Watson Speech-to-text***.

The first tool to be explored was ***SpeechRecognition***. This tool allowed to capture audio in real time and then recognize the audio using different services, such as *Google*, *IBM* and *Sphinx*.

Even though this tool had a accurate speech-recognition it did not allow much customization when it comes to the acoustic model, language model or even grammars, as it was mainly an *API* for those services.

It was then tried the ***IBM's Watson Speech-to-text*** with the *Lite plan*. This tool was used alongside ***SpeechRecognition***(to capture real-time audio), but when the custom *Language Model* was being created it was realized this feature was not available with the current plan.

The last tool used was ***CMU Sphinx***, more specifically, ***pocketsphinx***.

Even though this tool did not have a very accurate speech-recognition it allowed to create a custom *language model* and *acoustic model* by using ***lmtools***(which is part of the ***CMU Sphinx*** ecosystem) and providing it a corpus.

3 Application

The application makes use of the ***spotipy wrapper*** and allows the user to control *Spotify*'s desktop/mobile application with voice commands.

The actions that can be performed, and it's respective command, are listed in the table below.

Action List	
Action	Command
Next track	[next,next track, play next track]
Previous track	[previous track, previous]
Resume song	[resume, resume song, continue]
Pause song	[pause, pause song, stop]
Play song	[play, search]
Song name	[what is playing, currently playing]
Inactive	[sleep]
Active	[wake up]
Artist top songs	[artist]

Speech-to-text/Speech-Recognition is the most used interaction. The *Text-to-speech* (when the bot is talking) is only used to report a problem or to inform the user what song is currently playing.

Sbotify does only recognize a very restricted set of words (due to the dictionary provided), therefore there are a lot of songs that cannot be played.

The ideal solution for this problem would be the adaptation of a pre-trained model with some word additions.

IBM's Watson Speech-to-text would have been my choice if could make those additions with the *Lite* plan.

Another point to be taken in consideration is that application works better when it's in a quiet environment, as it would have been expected.

4 Conclusion

The assignment was partially completed with success. The time spent on this project was for the most part doing research.

Unfortunately a lot of the research did not materialized in results, instead simple tools ended up being used to demonstrate the unconventional interaction capabilities.