Domain Background

Because of not dealing with speech recognition problems, I decided to strengthen my skills in such domain and work on a speech classification example that covers my need for working in speech recognition tasks. In my project, I would like to build a radio website that plays a radio station and provide additional information below on the same page about the speaker that is currently speaking on the radio.

I decided to focus this project in building a demo project that covers a simpler problem than the final solution so I will just focus on building a speech classifier solution between two speakers and I will increase it's difficulty incrementally "Future Work".

The project's main library that I will be using is Librosa and here is the paper that explains the use of this library in audio analysis.

http://conference.scipy.org/proceedings/scipy2015/pdfs/brian_mcfee.pdf

Here is the link to the dataset that I will be working one:

https://github.com/Jakobovski/free-spoken-digit-dataset

Problem Statement

The problem needed to be solved in my project was to classify who is currently speaking on a certain radio station in order to be able to display more data about the speaker such as name, age, previous work, photos, etc. I think that it's not an easy problem to classify huge number of speech clips that could be possible in a radio station, but the radio station that I am targeting is currently having a few numbers of speakers so I think the problem will not require much processing requirements in terms of hardware capabilities.

The scope that I am targeting and I will implement in the capstone project is just to classify simple conversations and assign each to their speaker so I will work on the best ways to classify speech data, the best libraries such as Libarosa library, and all the way towards classifying a speech.

Datasets and Inputs

The dataset that I will be using is the Free Spoken Digit Dataset (FSDD) that mentioned here: https://github.com/Jakobovski/free-spoken-digit-dataset and it simply some audio/speech dataset consisting of recordings of spoken digits in WAV files at 8kHz. The recordings are trimmed so that they have near minimal silence at the beginnings and ends.

FSDD is an open dataset, which means it will grow over time as data is contributed. In order to enable reproducibility and accurate citation, the dataset is versioned using Zenodo DOI as well as git tags.

The wav files that are ready-made in the dataset above is perfect to be fed in the library that I choose for the project which is librosa so there is no need for extra modification for the dataset other than possible scaling or something related to the classification task.

More information about the dataset:

*It has information about 4 speakers.

*It has around 2,000 recordings (50 of each digit per speaker)

*All recordings are in the English Language.

Solution Statement

The solution which I will focus on the capstone is a simple classification problem for a speech dataset. The dataset is simple “about 4 speakers”, but I will use it as a demo for building the structure of the complete radio station solution “Future Work”. I will use the dataset provided above to create a suitable classifier that able to successfully classify speakers “assign audio clips to it’s speaker”.

Benchmark Model

The complete solution will be a website that targets a sports radio station and display some information about the current speaker. The solution will simply make a wav file out of the currently speaking speaker and feed this to the classifier which should be able to classify the speaker based on it's stored dataset. "There is a given info that the number of speakers on that certain radio station in limited so it will be easy to make a dataset of all the speakers in that radio station".

After correctly classifying the speaker, I will display some information about him/her that I see that will be useful for the visitor of the website such as the name, age, previous work, etc.

The model that we are trying to achieve is a model that can at least score more than "60%" on the test data. So we can think of this as our benchmark model that we are looking forward to build in the capstone project.

Here is an article about a model which works on only 3 different speakers, 1500 recordings (50 of each digit per speaker) and we can consider this model as a benchmark model for final comparisons.

https://towardsdatascience.com/speech-classification-using-neural-networks-the-basics-e5b08d6928b7

In the article the writer uses MLP to build a model and with the settings that he achieved applied in the model, he was able to achieve about 60% accuracy.

Evaluation Metrics

The success of my capstone is to build a model that achieves a better score in classifying speakers that present in the above dataset. I will begin by sampling the wave files which is simply representing it in terms of numbers so it could be used in a numpy array or the rest of the libraries that I will use on the project. As the Nyquist–Shannon sampling theorem showed that if our sampling rate is high enough, we are able to capture all the information in the signal and even fully recover it.

Then, I will use CNN and I think with the help of 32 Conv layers,and 'relue' activation function as inputs. I will add extra dropout and dense layers for adjusting the model's performance.

For comparing my capstone’s model with the benchmark models, I will use F-beta score for comparing the results in both models.

Project Design

The workflow of the project will be loading the data mentioned above, access it, feed the wave recordings for each speaker, build a CNN model that could simply use parts of the recordings and the assigned speakers and form it’s training and testing sets, and finally embed the solution in a final complete solution that will be used in a radio station website to classify the currently speaking person in a certain time during that day, then use this information to display info about the speaker.

Preprocessing steps: The actual sample rate for the elements in our dataset is 8000 using the load default method in Librosa library:

```
wav, sr = librosa.load(DATA_DIR + random_file, sr=None)
```

Setting sr=None prevents the library from applying the standard sampling which is 22050 so we will remove the None value to be able to use the standard 22050 sampling rate.