ICS2203 – ARI2203 – NLP Methods and Tools

Assignment: Single Word Speech Recognizer

The scope of this assignment is to create a speech recognizer capable of identifying a number of pre-trained keywords. For example, you may train your recognizer on a few number words like “one”, “two” and “three”.

There a number of steps you need to perform to complete this assignment. Most of these steps have already been covered in your lectures and practical session. This code should be re-used as much as possible, though you are free to try out alternatives and add any extra features.

Step 1 – Word List

You will need to decide what your keywords should be. At the very least, your recognizer should be capable of recognizing two short words, though it is always fun being able to recognize 3-5 words.

Step 2 – Record Training Audio

For each word in your word list, you will need to record yourself speaking the word multiple times. The differences between successive recordings will allow the model to encapsulate more of your vocal behavioural patterns. You may do this recording in two ways:

1) Write Python code that asks you to record a particular word, multiple times, saving each recording in a wav file.
2) Use a wav editor to record the files, without using Python.

In both cases, it will be good to view and listen to the output of the recordings in Ocenaudio. You are also allowed to play around with each file to clean the audio e.g. removing silent portions at the beginning/end of the recording.

It will help you to organise your recordings of each individual word into a particular folder for that word. Make sure that during your recordings, you are not too far from the microphone (normal sitting distance from a laptop should be fine), and that the room you are in is a quiet one.

Step 3 – Recording Testing Audio

In a similar fashion to Step 2, you are required to record testing prompts to use when evaluating your speech recognizer. For each word in your word list, record TWO test prompts. You should also organise these recordings into individual folders for the particular words. A good folder setup for your training and testing data is as follows:

```
/train
----->/one
---------->1.wav
---------->2.wav
…
---------->n.wav
----->/two
---------->1.wav
---------->2.wav
…
---------->n.wav
/test
----->/one
---------->1.wav
---------->2.wav
----->/two
---------->1.wav
---------->2.wav
```

Step 4 – Feature Extraction

In order to train individual models for each word in the word list, all training files of each word must be looped over, and MFCC features collected and amalgamated together. The features extracted for each word can be saved in a Python binary file. Consider using numpy.save for this purpose.

The binary files containing features for each word should be placed in the respective training folder for each word e.g.

```
/train
----->/one
---------->mfcc.npy
---------->1.wav
---------->2.wav
…
---------->n.wav
```

Step 5 – Build Word Acoustic Models

For each word in your word list, you will need to build a Gaussian Mixture Model (GMM) that models the MFCC features of the particular word. It is left up to you to decide on how many components are adequate for each GMM.

The GMM models you create should be saved to disk as well. Consider using sklearn’s inbuilt tools to do this e.g.

```
from sklearn.externals import joblib
joblib.dump(gmm, 'model.pkl') #saves
gmm = joblib.load('model.pkl') #loads
```

Save this model file in the same directory as the MFCC files e.g.

```
/train
----->/one
---------->mfcc.npy
---------->model.pkl
---------->1.wav
---------->2.wav
…
---------->n.wav
```

Step 6 – Testing

For testing purposes, you should write a single function that performs the following steps for every test file:

1) Reads the wave file from a given path
2) Extracts MFCCs
3) Scores the MFCCs under all word models
4) Pick up the model that registers the best score
5) Evaluate whether this model is for the training word corresponding to the evaluated word

Record these results for all your test cases.

Step 7 – Testing II

By sharing your test data with one of your class mates, evaluate an entirely different testing data set with your models, with the same procedure used in Step 6.

Note any similarities or differences between results in both test cases.

Deliverables

You will need to deliver all the code you wrote for this project, a ZIP of the entire directory of your work with training/testing data, features, models etc., as well as a short report of not more than 5 pages describing your work and results obtained. Do not forget to attach a plagiarism form with your report.

Deadline

15th January 2018