



# Accent Recognition

29.04.2017

---

Karan Vaidya (130050019)

Anmol Arora (130050027)

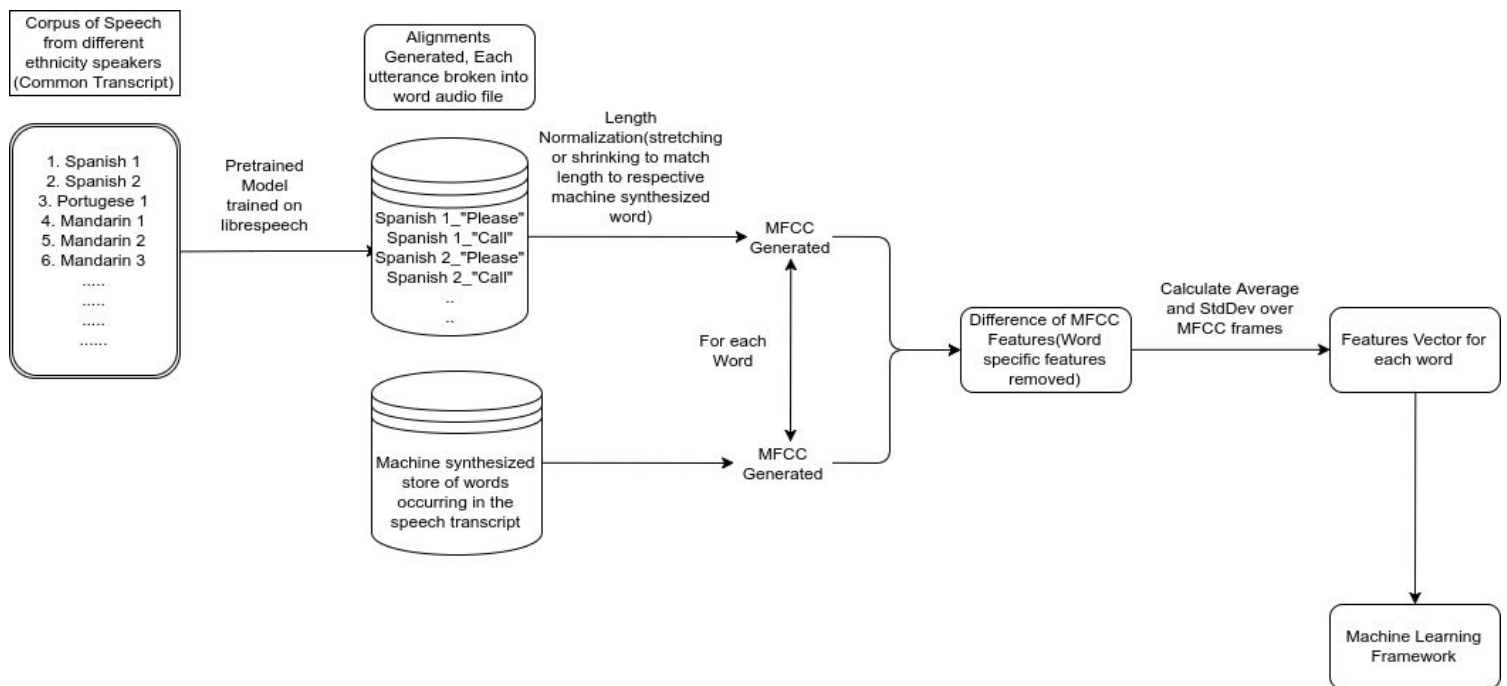
## Project Title

Nationality prediction through speech analysis

## Task Definition

Our end project takes the recording of the user speaking a reference language (in our case English) as input and recognize the accent and therefore nationality of the user using the accent information.

## Methodology



### i). Choosing a reference for removing the word related features

Our approach begins by choosing a speech reference. We, for example, chose the machine synthesized output of a text-to-speech engine (festival text2wave) as our reference, thus avoiding any human biases and noise in the data. We took a word-by-word approach and calculated mfcc features for the most common words in english by the engine and stored them. Using such an artificially generated speech as our reference, we can get accent free audio signals representing the word-particular audio features.

### ii). Getting features from training data

We used [GMU Speech Accent Archive](#) for our training dataset. Since the archive was not readily available, we first wrote scraping scripts to crawl the said page and download the entire audio corpus and related metadata.

All utterance collected from this archive were based on a common script. Using this transcript as our text, we sniped each utterance into multiple audio files corresponding to the composing words. This exercise was similar to the part 6 of assignment 2 of this course. We used a pretrained kaldi model which had been trained on 'LibriSpeech' corpus(LibriSpeech is a corpus of approximately 1000 hours of read English speech. The data is derived from read audiobooks from the LibriVox project and has been carefully segmented and aligned). We used tri2a exp model as our acoustic model.

Each audio file, corresponding to a word, was stretched/compressed in length to match the length of the reference word. This is to ensure that the number of MFCC frames generated for a word is same as number of MFCC frames of its reference word(machine generated) . We then generated the MFCC features for each of these segmented words.

### iii). Removing word specific features

At this stage we have MFCC features for various words spoken by speakers with various accent. Intuitively we would like to remove the component of features which comes from the word pronunciation to obtain feature component determined by accent. We want to remove the word specific features so that the clustering at later stage does not try to cluster together similar sounding words. Thus we subtract MFCC features of reference words from the MFCC of corresponding human spoken words. This operation is compatible since we ensure same word audio length at sniping stage, in turn generating MFCC vector equal in length to reference words'. The difference so obtained is termed Delta MFCC.

### iv). Converting spoken words into feature vector

We hypothesize that the word-specific data of the MFCC has been subtracted and the Delta MFCC so obtained is characteristic of speaker's style, including accent .

Since the length of MFCC so obtained is proportional to the length of the word audio file, we compute mean and standard deviation over these MFCC frames as our sufficient statistics. We think that the accent specific information is sufficiently captured by one MFCC frame(captured by mean of MFCC) and its variation over time(captured by standard deviation), hence combining these sequences of MFCC frames still retains the desired information. We have taken the cesp features (first 13) of MFCCs, giving us 26 length feature vector for every word spoken (13 for mean and 13 for std deviation of MFCCs of the word), irrespective of the length of the word.

#### v). Testing the main hypothesis

We have obtained same length feature vectors of spoken words,irrespective of the length of the word. The sufficient statistics so obtained will capture the accent information and will correlate more with the words of the similar accent, rather than similar sounding words (since we subtracted the features of reference word).

We have tried different supervised machine learning models to learn to classify the accent/nationality given the feature vector. Some of these include K-nearest neighbour, RandomForest Classifier, Decision Trees, SVM Classifier and Deep Neural Networks.

Once we learn this model, we can strive to predict the accent of given English utterance. To do so we perform same steps as we did with training data: break the utterance into words using suitable model; Resize the word audio size to match reference words; calculate the MFCC features of these words; Subtract the MFCC features of reference words; Perform mean and variance of MFCC frames over each word to obtain as many feature vectors as the number of words ; and finally apply the trained ML model to get the probability distribution of accent for each word. Finally we take a weighted average over the probabilities(confidence score) and length of the composing words(as longer words have more syllables and are bound to capture more accent related information) to finally label the utterance.

## Implementation Details

- Code Source @ <https://github.com/mewtomat/AccentDetection>
- Used python Library urllib to scrap all the audio files from GMU accent archive
- For getting machine-produced text, we found and worked with quite few engines including **espeak**, **festival** etc. and ended up using the TTS of festival as our reference. For reference dictionary, we currently gave words from various speeches of GMU accent archive and other multi-accent archives. [baseAudioGen.sh](#) takes a text document and produces a folder with all the TTS generated words
- For getting mfcc features, we are using python's scikits talkbox to get mfcc from the standard word files, which is being done in [convert\\_mfcc.py](#)
- For the training data, which is in the part of speeches of various accents, we are aligning the speeches and breaking them into individual words, using a kaldi model, pretrained on **librispeech** dataset. We used this as it was available as a pretrained archive, online for free. Then we used [wordsnip.sh](#), which internally uses sox to break the words from various speeches. The code for the same is in [break\\_word.py](#)
- We had to make the recordings of the words of same length to get same sizes of the mfcc features. To expand and contract the sound files, we used a python library **Paulstretch**, but as it didn't give any good results, we shifted to using **sox** finally.
- Now, we created the mfcc features of these words too using [getMFCC.py](#)
- Now, we train the model using sklearn cluster's library, on both supervised and unsupervised KNN models
- We also trained same features on various decision tree classification models, GBM, Random Forest, Neural Network, SVM using sklearn modules for the same
- Training and creation of features by preprocessing MFCC features is being done in [train.py](#).

## Experimental Setup

We scraped GMU Accent Archive and collected over 2300 utterance from people of 193 ethnicities. These audio files were processed into kaldi compatible wav format(44.1KHz) resulting in database of around 6.4 GB. Each of these utterance was aligned with the given transcript and snipped into words. This resulted in a corpus of 55(words in transcript) \* 2300(utterances) ~ 120000 audio files(8GB).


For each of these 55 words, we randomly chose one sample of each nationality. We couldn't work with the entire dataset, which from our extrapolations over few runs, would have taken over 30 hours to run. Our sampling gave us 55(words)\* 193(nationalities) ~ 10000. Each of these words was processed to contribute one data point in the machine learning framework.

The dataset so obtained was split into training(70%) and testing (30%) and trained using below mentioned algorithms. The accuracy so given is over the test dataset.

## Experiments and Discussion

### Results of KNN

1. weights: uniform neighbors: 15 -
  - a. Training data accuracy =31, Testing data accuracy = 14.09
2. weights: uniform neighbors: 2 score: 0.111495844875
3. weights: uniform neighbors: 4 score: 0.114612188366
4. weights: uniform neighbors: 6 score: 0.129501385042
5. weights: uniform neighbors: 8 score: 0.133310249307
6. weights: uniform neighbors: 10 score: 0.140927977839
7. weights: uniform neighbors: 15 score: 0.140927977839
8. weights: uniform neighbors: 25 score: 0.140235457064\
9. weights: uniform neighbors: 30 score: 0.144736842105
10. weights: uniform neighbors: 40 score: 0.145775623269
11. weights: uniform neighbors: 50 score: 0.141966759003
12. weights: distance neighbors: 2 score: 0.111495844875
13. weights: distance neighbors: 4 score: 0.114612188366
14. weights: distance neighbors: 6 score: 0.129501385042
15. weights: distance neighbors: 8 score: 0.133310249307

- 
16. weights: distance neighbors: 10 score: 0.140927977839
  17. weights: distance neighbors: 15 score: 0.140927977839
  18. weights: distance neighbors: 20 score: 0.139542936288
  19. weights: distance neighbors: 25 score: 0.140235457064
  20. weights: distance neighbors: 30 score: 0.144736842105
  21. weights: distance neighbors: 40 score: 0.145775623269
  22. weights: distance neighbors: 50 score: 0.141966759003

### Results of Decision Tree

min\_samples\_leaf=1, min\_samples\_split=2 - 0.088296398891966763

### Results of Random Forest

1. n\_estimators=10,max\_depth=3, random\_state=0,n\_jobs=2 - 0.071329639889196675
2. n\_estimators=10,max\_depth=5, random\_state=0,n\_jobs=2 - 0.11080332409972299
3. n\_estimators=10,max\_depth=10, random\_state=0,n\_jobs=2 - 0.13885041551246538
4. n\_estimators=20,max\_depth=10, random\_state=0,n\_jobs=2 - 0.17105263157894737
5. n\_estimators=100,max\_depth=10, random\_state=0,n\_jobs=2 -0.24272853185595566
6. n\_estimators=500,max\_depth=10, random\_state=0,n\_jobs=2 -0.26073407202216065
7. n\_estimators=500,max\_depth=15,random\_state=0,n\_jobs=4-0.30228531855955676
8. n\_estimators=500,max\_depth=20,random\_state=0,n\_jobs=4-0.31959833795013848
9. n\_estimators=1000,max\_depth=30,random\_state=0,n\_jobs=4-0-0.327908587257617

### Results of MLP Classifier

1. MLPClassifier(activation='relu', alpha=0.0001, epsilon=1e-08, hidden\_layer\_sizes=(600, 300), learning\_rate='constant', learning\_rate\_init=0.001, max\_iter=200, momentum=0.9 - 0.29674515235457066
2. hidden\_layer\_sizes=(400,200),solver='adam', alpha=1e-4,random\_state=42 - 0.31128808864265928

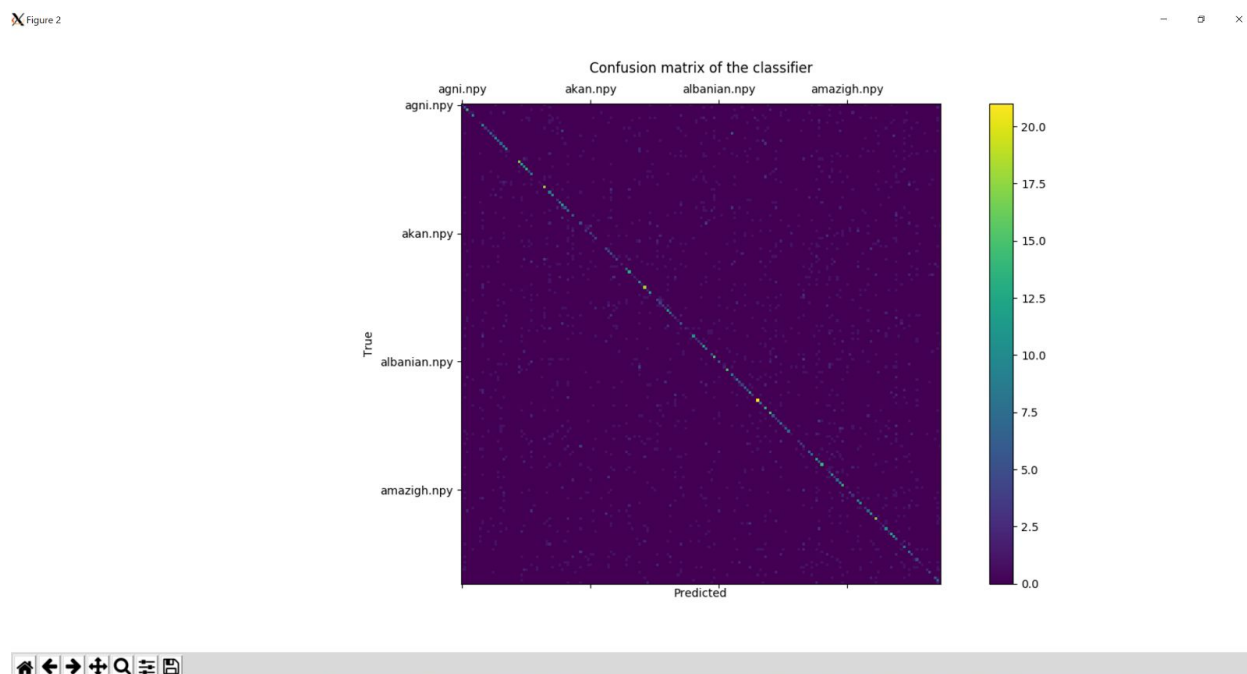
### Results of SVM

1. Default - 0.26558171745152354

## Summary

The best result obtained is from Random Forest ( $n\_estimators=1000, max\_depth=30$ ) which gives an accuracy of over 32.8% over test data (against 0.5% from randomly guessing over 193 classes).

The confusion matrix for the same is as below:



This shows that the proposed methodology can be used to extract accent related information from voice signals.

## Resources Used

- I. Dataset - <http://accent.gmu.edu>
- II. Speaker-basis Accent Clustering Using Invariant Structure Analysis and the Speech Accent Archive - <http://cs.uef.fi/odyssey2014/program/pdfs/46.pdf>



- 
- III. Accent Clustering in Swedish Using the Bhattacharyya Distance -  
<http://www.speech.kth.se/prod/publications/files/1125.pdf>