Accent Classification in Spoken English

Kunal Vijay Ninawe
Computer Engineering, New York University
kvn238@nyu.edu

Advised by: Prof. Yong Liu

ABSTRACT

In this report, I explain the reason behind developing this project, how it is implemented and what results I got for my Accent Classifier. The aim of this project is to provide an Accent Classification in one of the four accents just from an audio file, which will contain a passage that the user records. Related work in the field attempts to predict similarly 3-4 accents with a Neural Network. My work uses a different idea to pre-process the data from the Speech Accent Archive dataset. I then use simple machine learning models to classify 4 different accents which achieves an accuracy that exceeds related work in the field by 30-40%.

KEYWORDS

Speech Accent Detection, Audio Processing, Data Cleaning, Machine Learning

1 INTRODUCTION

The problem that I am addressing here is to classify an speech audio sample using a multi-class accent classifier and to improve the accuracy of detection for each accent.
The motivation behind developing this project is to create a tool that could help people(International Students, workers, actors, etc) improve their accents.

Automatic Speech Recognition(ASR) like Amazon Alexa or Apple Siri is another area that can be benefited from an accent classifier, as after accent detection, they can then target a particular region's model and dataset to get better results.

There are certain challenges that this dataset and project posed, like having a small and highly imbalanced data, having as low as 1 sample for one language and as high as 373 for another.

Another challenge that I faced was extracting relevant features from the audio file which would best differentiate one accent from another.

2 RELATED WORK

There are many ways to differentiate one accent from the other, like a "phoneme", which is essentially how one word sounds different in different languages. Another way is that we can see how their vocal chord vibrate when they talk in that accent. Other factors are, we can consider the "Intonation" in different accents and the pronunciation of certain letters like "R" and "T" in accents.

As pointed out in [1], Some of the preceding researches focused on the link between the "word duration" and "accentedness", while some others focused on Short Time Fourier Transforms. Just like [2], I will also stick with Mel Frequency Cepstral Coefficients or "MFCCs" as they are more commonly called.

3 IMPLEMENTATION AND EXPERIMENTS

I used Google Collaboratory, which is essentially a Jupyter Notebook on Google Cloud, and my final published code can be found on Github.

3.1 Dataset

All of the speech files used for this project come from the Speech Accent Archive, a repository of spoken English hosted by George Mason University. Over 2000 speakers representing over 100 native languages read a common elicitation paragraph in English. I imported this dataset from Kaggle.

- The common nature of the dataset makes it ideal for studying accent, being that the wording is provided and the recording quality is (nearly) uniform across all speakers.
- All speakers are speaking the same paragraph in each audio sample.
- From all those 2000 samples, after data cleaning, only the accents with highest number of samples were selected resulting in total of 636 out of 2000 samples. The resulting accents where American, Indian, Chinese and British after filtering.
- Now all that needs to be done is to map the similarities between these samples using a common feature.

3.2 Audio Processing and Feature Extraction

3.2.1 Data Cleaning. After extensive data pre-processing to get 636 samples, which includes removing empty and unnecessary columns(age, birthplace, speakerid, filemissing), filtering NaN values, removing missing audio files, and files from accents other than our 4 classes.

Then, similar to [6], the audio files were converted to a .wav format(since it is more universal and less compressed format of audio) using python's pydub library.

During feature extraction, I needed to analyze which feature needs to be extracted, which can differentiate one accent from another. For this, a few features were extracted from an audio file, like a simple Waveform and Short Time Fourier Transform as shown in Figure 1 generated using the Librosa Library in python.

3.2.2 Mel Frequency Cepstral Coefficients. Now, in the final implementation, the focus is on extracting the Mel Frequency Cepstral Coefficients(MFCC) from each audio file.

A MFCC is a Frequency Domain Feature of a sound, which is fundamental in speech recognition research and captures textural aspects of a sound. In layman terms, as explained in [5], MFCCs can be considered to be representations of the motions of the vocal tract(x-ray of a mouth's vocal tract).

I extracted 13 of these mfccs at each frame from an audio using the Librosa library and each of those 13 coefficients approximates the human auditory system(how we interpret differences sound). These mfccs will now be used as input to our machine learning models.

3.2.3 Handling Imbalanced Dataset. The 636 samples extracted after cleaning were heavily imbalanced and the sample count of each accent was: American(373 samples), Indian(110 samples), Chinese(88 samples) and British(65 samples).

Training a Support Vector Machine on this imbalanced data just to see some preliminary results was insightful, since the metrics showed that the f1-scores of accents other than American(0.74) were extremely poor (ranging from 0.1-0.2). Now here, a challenge that needs to be resolved is to atleast balance the f1-scores.

F1-Score is a good metric when the data is imbalanced, just like the cleaned data in my case. It is essentially the harmonic mean of recall and precision(other metrics). I accessed F1-scores for my Machine Learning models using scikit-learn's metrics.

Now To resolve the imbalance in the dataset and to balance the f1-scores of each accent, different methods were tried and tested.

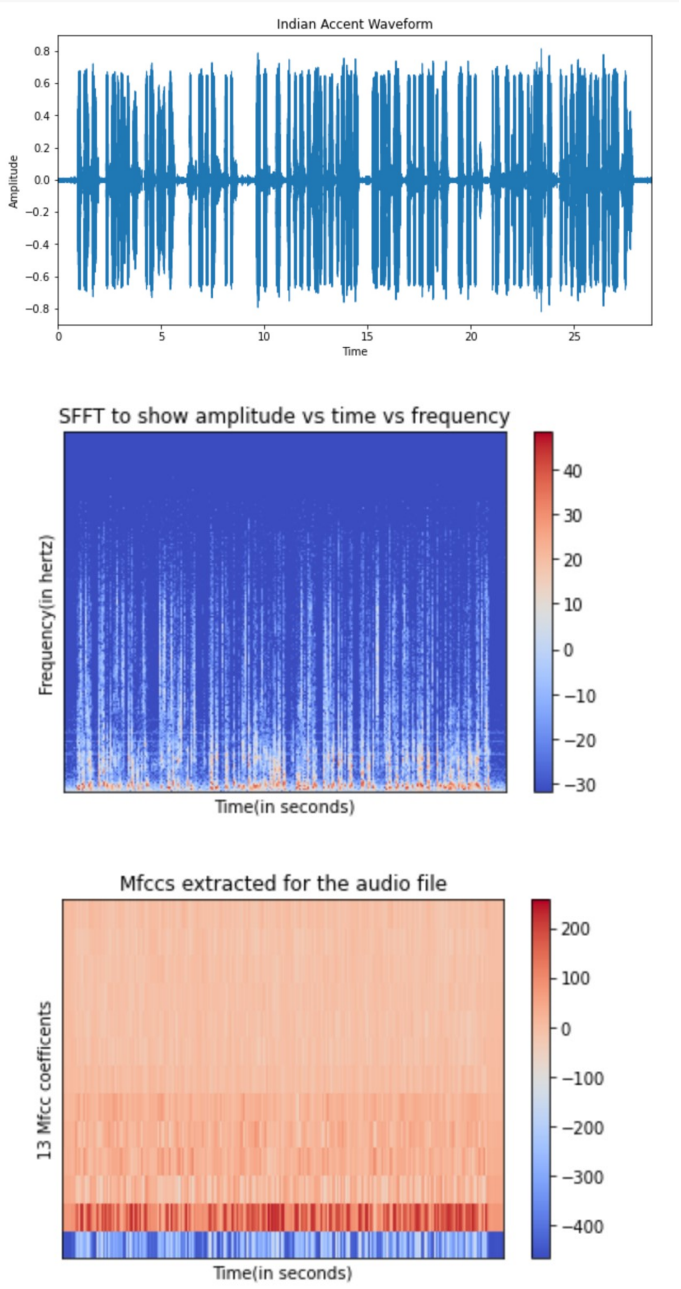

Figure 1: A sample plot of audio sample's waveform followed by a Short Time Fourier Transform and MFCC plot.

- Initially, following the techniques referenced from [3], which used SMOTE, improved accuracy by 5% and increased the f1-scores of minority classes by 0.2.

- Next, under-sampling the American Accent showed disappointing results with a dropped accuracy from 59 to 41%, but a better f1-score.
- Lastly, the technique that resulted in best metrics was to over-sample the minority classes and add a random Gaussian Noise derived from the same audio file, the technique mentioned in [4].

3.2.4 Machine Learning models. In literature review, the previous researches like [2] focused on training a Neural Network, specifically a Convolutional Neural Network(CNN) and a Recurrent Neural Network(RNN) where they got an accuracy close to 60%. Since my dataset was not too big I choose to experiment on various Machine Learning models instead and using the hit and trial method to pick the model that performed best. GridSearchCV was performed on each model to get the best hyper-parameters.

I examined 4 different models using the scikit-learn library in python namely, Support Vector Machine(SVM), Random Forest Classifier(RFC), Logistic Regression(LR) and K-Nearest Neighbors(KNN) on the over-sampled MFCC dataset created and then compared the metrics, specifically the Overall Accuracy and F1-scores for each accent in the models.

- For the SVM, I implemented an "rbf" (Radial basis function) kernel with Regularization parameter, C=10 and turned on the probability parameter, which will be used in the evaluation section. This worked exceedingly well, with an overall accuracy of 93% and a balanced f1-scores of above 0.85 for each accent.
  This model was trained on two datasets. First was the over-sampled with noise data, and other was over-sampled without noise.
- For RFC, I choose a max_depth = 16 and n_estimators=250 with a balanced class weights. RFC performed almost as good as SVC with an overall accuracy of 92% and similarly balanced f1-scores. RFC's model was similarly trained on two datasets like SVM.
- Logistic Regression and KNN did not work as expected on the test data, with a lower accuracy of 88% and 71% respectively. Due to a low score in these models and SVC and RFC performing well, there was no point exploring these models more.
- Since previous researches focused on RNN and CNN, I choose to include them in the hit and trial method, but I got a low score, similarly of almost 60%. Again, due to SVC and RFC working well and these neural networks had a low score on test data, the idea to explore these models was scrapped.

4 EVALUATION

After comparing results and metrics of different models against each other, as it can be seen in Figure 2, Support Vector Machine(SVM) work the best in this case. Random Forest Classifier(RFC) comes in close proximity of SVM, and only these two models are chosen when predicting on a real sample.

Model	Accuracy
Support Vector Machine	93%
Random Forest Classifier	92%
K-Nearest Neighbors	76%
Logistic Regression	87%

Model	F1-Scores			
	American	British	Chinese	Indian
Support Vector Machine	85%	98%	99%	89%
Random Forest Classifier	85%	100%	96%	87%
K-Nearest Neighbors	55%	83%	89%	75%
Logistic Regression	79%	96%	94%	82%

Figure 2: Comparing different models and their metrics like Accuracy and F1-Scores for each accent.

4.1 Prediction Algorithm

In this algorithm, both models, i.e. SVM and RFC that were trained on 2 different datasets each, were taken(4 models total) and prediction was made by first seeing if any model predicts with a probability of 90%(strong prediction), then return that prediction; Else take the average of all the classes returned by 4 models and return the prediction that has the maximum average.

4.2 Evaluation Techniques

The models were evaluated in threefold part.

- Firstly, use the 636 raw audio samples in our dataset, extract mfccs from each in a regular fashion and pass each sample's mfccs to the "predict" function of the 4 models and use prediction algorithm to get a prediction. Store the results and determine if this is correct or not. Naturally, the accuracy was high of 95%, with the only 30 samples incorrectly predicted.
- Now, to test on real world samples, I collected almost 75 additional samples from volunteers, asking them to narrate and record the same paragraph in the dataset. The accuracy dropped to almost 70%, which was insightful since there is a need to tweak the models to work well with the real world samples.
- Lastly, make an upload section to upload a random audio file, extract its mfccs and use the prediction algorithm to test how well it worked on random samples. This is what is going to be the basic steps when converting this pipeline into a Flask Web API.

While the gap in the accuracy between present dataset and the real world samples is high, The overall accuracy and f1-scores for each accent is significantly higher than those reported by previous researches and the prediction algorithm is particularly good in predicting the Indian accent in real world samples.

While collecting the real world samples, the need to do an extensive audio cleaning was highlighted. Observations were made that the model predicted more accurately if the speaker's pace is normal to slow(not too fast), the duration of the file is between 25-30 seconds, there is not too much disturbance in the speech and the user does not speak weirdly.

It can also be seen that if we get a significantly more number of diverse samples for each accent, we can improve the real world sample accuracy and possibly have enough samples for a neural network to understand the trends in the data better.

5 CONCLUSION AND FUTURE WORK

After evaluating models using different techniques, it can be seen that the Support Vector Machine(SVM) Classifier is most accurate in identifying the trends in the real world samples. RFC lags behind than SVM on many samples, but there are cases observed where RFC could predict correctly where SVM could not. Those trends can be analysed in the future work and the Prediction Algorithm can be improvised to consider these situations.

As part of future enhancements, this entire pipeline can be converted into a flask application to make an interactive website that the user can use and test their accents in real time, which can be hosted on Amazon's S3 cloud.

Furthermore, more samples can be collected from the NYU community as well as samples from the website can be stored with user's consent and models can be retrained in a regular interval to update the dataset and increase the prediction accuracy on real world samples.

REFERENCES

[1] 2011. Word durations in non-native English. *Journal of Phonetics* 39, 1 (Jan. 2011), 1–17. https://doi.org/10.1016/j.wocn.2010.10.006

[2] Mariam Elgamal and Yeojin Jung. 2020. English Accent Detector Final Project Report. https://github.com/mariamelgamal/English-Accent-Detector/blob/master/English_Accent_Detector_Final_Report.pdf

[3] Brownlee Jason. 2021. SMOTE for Imbalanced Classification with Python. https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

[4] Wijayasingha Lahiru. 2021. Adding noise to audio clips. https://medium.com/analytics-vidhya/adding-noise-to-audio-clips-5d8cee24ccb8

[5] Eu Jin Lok. 2019. Part 2 - Extracting Audio Features. https://www.kaggle.com/ejlok1/part-2-extracting-audio-features

[6] David Weisberger. 2016. dwww2012/Accent-Classifier. https://github.com/dwww2012/Accent-Classifier