ACCENT DETECTION IN ENGLISH LANGUAGE USING DEEP NEURAL NETWORKS

Nishant Jain (nishjain@iu.edu), Nitesh Jaswal (njaswal@iu.edu)

ABSTRACT

With the advent of deep neural networks, deep learning has been widely used in many applications, with its first applications being in object recognition domain. And in recent years deep learning have been widely adopted for audio domain as well. This time we wanted to solve the problem of speech accent recognition, which hasn't been taken up by many people, although some work has been done in this domain, we have listed that in the references.

Index Terms— *accent identification, deep neural networks, speech processing, acoustics*

1. INTRODUCTION

English accent classification is defined as the problem of classifying the native language of a speaker from the speech signal of a non-native English speaker. Automatic Speech Recognition Systems (ASR) perform relatively poorly in identifying the speech of non-native English speaker [1]. By developing speech accent recognition algorithms, the existing ASR systems can be modified to better capture and recognize the speech signal of foreign accented speakers. In addition to ASR systems, accent identification can have a variety of other applications such as forensic analysis, targeted marketing and advertising etc.

Previous studies have analyzed how elemental components of speech change depending on the native language of the speaker [2] [3]. It has been shown that both spectral and temporal features of sound change with the accent of the speaker. Here, spectral features of sound refer to the frequency domain analysis of the speech signal and consists of attributes such as formant frequencies (A formant is a concentration of acoustic energy around a particular frequency in the speech wave [4]), spectral densities, frequency components etc. The temporal features refer to the time domain attributes of the speech signal such as intonation (variation in spoken pitch), prosody (elements of speech that are not individual phonetic segments (vowels and consonants) but are properties of syllables and larger units of speech, duration etc.

Previously, various statistical methods and machine learning models like Gaussian Mixture Models, Hidden Markov Models and Support Vector Machines (SVM) have been used to perform and automate this classification task [5] [6] Artificial Neural Networks such as DNNs and RNNs have been widely used in advanced speech systems, however, their application in the particular task of accent identification have been relatively unexplored.

In this project, we try to implement and compare the performance of various deep learning architectures such as CNNs, RNNs and Dense Feed-Forward systems in classifying the accent of the speaker given the speech signal. This is limited to English language only for now, and later on can be extended to multiple languages.

2. DATASET

While looking for a dataset for this task, we wanted our dataset to be big enough to be able to use the power of deep learning, as well as we wanted a data which is diverse. Another important point to keep in mind with audio data is that it should be clean and without any noise, i.e. the audio should have been recorded in a noise-free environment. Also, it would be beneficial if the audio signals contain the same transcript read by different people.

Our requirements perfectly matched with the open database provided by Speech accent archive hosted by George Mason University [7].

This dataset contains 2140 speech samples, each from a different speaker reading the same passage. The speakers come from 177 countries and have 214 different native languages. Each person is speaking in English.

Each of the speakers speaks out the following transcript:

"Please call Stella. Ask her to bring these things with her from the store: Six spoons of fresh snow peas, five thick slabs of blue cheese, and maybe a snack for her brother Bob. We also need a small plastic snake and a big toy frog for the kids. She can scoop these things into three red bags, and we will go meet her Wednesday at the train station."

Distribution of speech samples by native language (top 20):

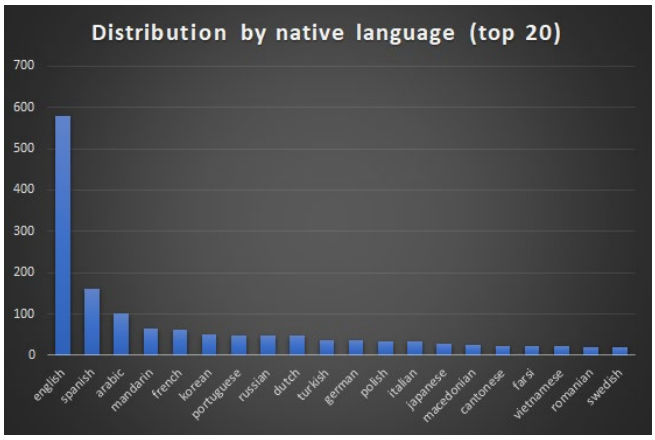

Figure 1: Distribution of speech samples by native language (top 20)

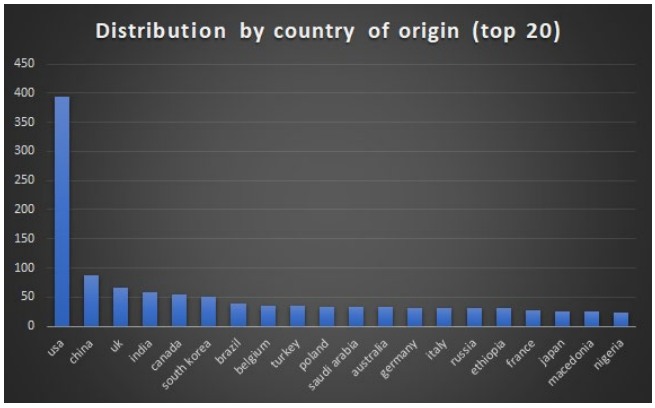

Figure 2: Distribution of speech samples by country of origin (top 20)

It was observed in the data that sometimes the native language was different than expected as per the country of origin, hence we decided to distinguish speech samples on the basis of native language only.

We grouped the data for similar native languages (as per linguistic similarity) into 5 distinct groups as per below:

Approach 1:

Group 1: German, Dutch, Swedish, Norwegian, Danish (117 samples)
Group 2: Hindi, Urdu, Punjabi, Nepali, Telegu, Bengali (74 samples)
Group 3: English (200 samples)
Group 4: Spanish, Portuguese, French, Italian (150 samples)
Group 5: Japanese, Korean, Mandarin, Cantonese, Taiwanese (175 samples)
Group 6: Arabic, Farsi (Persian) (125 samples)

Approach 2:

We tried an alternative as well with smaller subset of data, keeping only 50 samples per group:

Group 1: French
Group 2: Hindi and Urdu
Group 3: English
Group 4: Spanish
Group 5: Mandarin
Group 6: Arabic

This helps us solve multiple problems we had with the data. Firstly, for some languages like Hindi, Urdu, Telegu etc, the number of overall speakers was very low. By combining these linguistically similar languages into groups we were able to increase the amount of data we had for each group.

Secondly, according to this paper by Yishan Jiao , Ming Tu et. al [8] , first trying to develop a hierarchical classifier that works by identifying groups of languages first (for example, Japanese, Chinese and Korean could be grouped) and then proceeding to give a more fine-grained decision could yield better and more robust results..

3. FEATURE EXTRACTION AND PREPROCESSING

We used several techniques widely used in the domain of audio and speech processing. We used the following: -

Short Time Fourier Transform (STFT):
When using whole signal, we kept hop length as 512, and frame size as 1024.

After splitting into 500 ms, we kept hop length as 256, and frame size as 512

MFCC (Mel-frequency cepstral coefficients):
We kept the same hop length and frame size as above. The number of coefficients used were 40.

Log-Mel Spectrogram:
Again, we kept the same hop length and frame size. The number of mels we used was 40. After obtaining the spectrogram, we took its logarithm.

Pre-Emphasis: Pre-emphasis is the first part of a noise reduction technique in which a signal's weaker and higher

frequencies are boosted before it is transmitted or recorded. We applied pre-emphasis to our audio signal before transforming them with the above-mentioned operations. We used pre-emphasis to amplify the high frequencies. A pre-emphasis filter is useful in several ways: (1) It balances the frequency spectrum since high frequencies usually have smaller magnitudes compared to lower frequencies, (2) It avoids numerical problems during the Fourier transform operation and (3) may also improve the Signal-to-Noise Ratio (SNR). The pre-emphasis [9] filter can be applied to a signal x using the first order filter in the following equation:

$$y(t) = x(t) - \alpha x(t-1)$$

where, the typical values for α can be 0.95 or 0.97.

After applying STFT and other filters, we applied **mean normalization** to balance the spectrum and improve the Signal-to-Noise (SNR), which is simply subtracting the mean of each coefficient from all frames.

Breaking Data Into Parts:

We decided to further break down our speech signals into 500 ms segments. Then we constructed a one hot vector of labels for each of the data examples. This way we expand the data points to be fed into the system. After getting the output from the neural network, we then recombine the predictions by taking the average probability corresponding to each class across all the segments.

We used sklearn library's "train_test_split" function to split the data into training and testing data in the 80:20 ratio.

As most of the samples were 20 sec long, we decided to use only 10 sec worth of data to train the architecture. For testing we had 3 approaches in mind,

1. Test using next 10 sec of the same samples
2. Test using same 10 sec of the different samples
3. Test using next 10 seconds of the different samples

4. NETWORK ARCHITECTURE

We tried different types of model architectures with varying number of parameters. We essentially focused on using 2D convolutions along with dense layers and a final SoftMax layer. We also tried RNN followed by dense layers as well but that didn't help in the testing accuracy. For all the architectures, a learning rate of 0.0001 was employed. We tried multiple batch sizes of 32, 64, 128. We noticed keeping a batch size smaller or larger than these values made the convergence much slower.

Architecture 1:

Conv1: 32 filters of size 3x3 and strides of 2
Pool1: pool size of 3x3 with strides of 2
Conv2: 16 filters of 3x3 with strides of 2
Pool2: pool size of 3x3 with strides of 2
Flatten: flattening after pool2 to convert to 2D
Dense1: 1024 units (or 512 units)
Dense2: 256 units
Dense3: 6 units, SoftMax

This model took the STFT of the 500ms data segments as the input. We tried using filters of 5x5 in both the convolutional layers, and also tried changing the strides to 1 instead of 2. But we found it did not give any advantage in the performance of the model and hence we stuck with the current explained model.

Architecture 2:

Conv1: 32 filters of size 5x5 and strides of 1
Pool1: pool size of 3x3 with strides of 2
Conv2: 16 filters of 5x5 with strides of 1
Pool2: pool size of 3x3 with strides of 2
Conv3: 8 filters of 3x3 with strides of 1
Pool3: pool size of 3x3 with strides of 1
Flatten: Flattening after pool3 to convert to 2D
Dense1: 512 units
Dense2: 256 units
Dense3: 128 units
Dense4: 6 units, SoftMax

Like Architecture 1, this model took the STFT of the 500ms data segments as input. We hoped that by adding an additional convolutional layer and reducing the stride of the final pooling layer to 1, our model could learn better and more robust features from the input spectrograms.

Architecture 3:

RNN Layer (using LSTM cells): 128 output units with tanh activation function. This layer was configured to return the entire sequence
Flatten: Flattens the output of the RNN layers
Batch Normalization: Normalizes the batch statistics
Dense1: 512 output units with ReLU activation function
Dropout 1: A dropout probability of 0.8

Dense2: 256 output units with ReLU activation function
Dropout 2: A dropout probability of 0.8
Dense3: 6 output units, SoftMax activation

The input to the RNN was the log mel-spectrogram of the input samples that had been split into segments of 500ms each. The layer was configured to return the full output sequence for each timestep. This output was then flattened and fed into a 3 Layer Dense network with the last layer outputting the SoftMax probabilities for each of the 6 classes of our groups.
We tried another variation of this architecture in which we added two convolutional layers with max pooling aster the RNN layer. However, this architecture was found to easily overfit and perform much poorly hence it was dropped.

The Architecture 1 was easiest and fastest to train due to less number of trainable parameters.

We used the Google Colab environment using TensorFlow to build and train the models. We used the GPU runtime offered by Google which uses Nvidia K80 GPU with 15GB of GPU ram and 12GB of CPU ram available. We also tried training on our personal system using Nvidia GTX 1060 Cuda cores with 3GB of ram, but the amount of ram limited our batch size and many times resulted in crashing of the session.

5. RESULTS

We were able to achieve training accuracies ranging from 50% to ~95% using Architecture 1 and 2. However, we noticed training accuracies higher than 85%, the testing accuracies began to drop suggesting we were overfitting.
For these two CNN Architectures, we got a maximum testing accuracy of 35%, which is better than 16%., the probability of randomly guessing the class label. However, this is still far from the point of view of practical usability. Both for Architecture 1 and 2, we noticed that the system performs exceptionally well in predicting Group 3, Group 4 Group 5 and Group 6 containing English, Spanish, Mandarin and Arabic respectively. However, it performed extremely poorly in predicting Group 1 and Group 2 containing French and Hindi/Urdu. In many cases it did not give a prediction for these groups even once.

For Architecture 3, we could get training accuracies of 85%.and testing accuracy of 33%. However, we noticed that the results of this architecture seemed more robust and it gave some meaningful prediction for each of the groups unlike the results of Architecture 1 and 2.

6. CHALLENGES

Deciding on which filter to use, we had to try each of them to came to a conclusion that STFT works better in our case.
Selecting language groups and number of speech samples in each group.
Avoiding single class predictions. At first our model was converging in a way where it was predicting the same class for all the samples. We applied two techniques to overcome this, firstly applying pre-emphasis and mean normalization as explained in section 3. Secondly, by feeding randomized input sample in each batch.

7. CONCLUSION AND FUTURE WORK

We conclude that though we were able to achieve better results than the random choice probability, more can be done to improve the system to take it to state of art level.

One can explore more datasets to work with in order to improve the testing accuracy. Also, more architectures could be explored, for e.g. more complex network can be used given enough computational resources and time. A discriminant network like a Siamese architecture can be explored as well. One can also try working directly with unfiltered speech signals as well and make use of 1D convolution and time-domain RNN sequencing to create end-to-end accent recognition systems. And a smaller problem can be tackled first by taking only three groups and then upon successful implementation of that, it could be scaled to a bigger classification model. Lastly, an ensemble technique could be applied combining multiple promising models/architectures.

8. REFERENCES

[1] - William Byrne, Eva Knodt, Sanjeev Khudanpur, Jared Bernstein “Is Automatic Speech Recognition Ready for Non-Native Speech?A Data Collection Effort and Initial Experiments in Modeling Conversational Hispanic English” e 1998 ESCA Conference on Speech Technology in Language Learning. Marholmen, Sweden.
[2] L. M. Arslan and J. H. Hansen, “Frequency characteristics of foreign accented speech,” in Acoustics, Speech, and Signal Processing (ICASSP), IEEE International Conference on, vol. 2. Munich, Germany: IEEE, 1997, pp. 1123–1126.

[3] E. Ferragne and F. Pellegrino, “Formant frequencies of vowels in 13 accents of the british isles,” Journal of the International Phonetic Association, vol. 40, no. 01, pp. 1–34, 2010.

[4] What are formants? [https://person2.sol.lu.se/SidneyWood/praate/whatform.htm]

[5] S. Deshpande, S. Chikkerur, and V. Govindaraju, “Accent classification in speech,” in Automatic Identification Advanced Technologies, Fourth IEEE Workshop on. Buffalo, NY, USA: IEEE, 2005, pp. 139–143.

[6] Y. Zheng, R. Sproat, L. Gu, I. Shafran, H. Zhou, Y. Su, D. Jurafsky, R. Starr, and S.-Y. Yoon, “Accent detection and speech recognition for shanghai-accented mandarin.” in Interspeech. Lisbon, Portugal: Citeseer, 2005, pp. 217–220.

[7] Speech Accent Archive: http://accent.gmu.edu/

[8] Yishan Jiao, Ming Tu, Visar Berisha, Julie Liss, "Accent Identification by Combining Deep Neural Networks and Recurrent Neural Networks Trained on Long and Short Term Features", INTERSPEECH 2016

[9]https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html