Accent Classification

Phumchanit Watanaprakornkul, Chantat Eksombatchai, and Peter Chien

Introduction

Accents are patterns of speech that speakers of a language exhibit; they are normally held in common with people of similar ethnic, national, or social background. Accent classification is the problem of finding out what accent people have, that is taking speech in a language from speakers from various ethnic or national backgrounds and using that speech to figure out what backgrounds the speakers are from. This provides for a way to extract useful information - where a person is from - without directly asking that person about it, and instead just listening to them speak. However, this is not always possible for humans, as sometimes they encounter accents that they have not heard before, or they come across accents that are only faintly noticeable, and then they are not able to classify the accents correctly.

This paper considers applying machine learning to the problem of accent classification in the hopes of being able to reliably classify accents that are in some data set, which can be much greater than any one human has the time to listen to.

Further motivation for examining the problem of accent classification comes from the problem of speech recognition. Currently, accented speech poses problems for speech recognition algorithms, as different pronunciations of the same words are not recognized the same way by speech recognition algorithms. Indeed, sometimes it is hard even for native speakers of a language to discern exactly what someone else is saying if they have a thick accent. If speech could first be classified by accent, the speech could then be passed on to some more specialized speech recognition algorithm, such as one that was trained on people with the same accent as the speaker.

Dataset

We used the CSLU: Foreign Accented English v 1.2 dataset. (Linguistic Data Consortium catalog number LDC2007S08). It is composed of English speech from native speakers of 23 different languages. The data format is 1 channel 16-bit linearly encoded WAV files sampled at 8kHz. Although each file contains English speech, the files contain different words and are not necessarily complete sentences. We did not work on all 23 accents. We picked three accents - Cantonese, Hindi, Russian - to work on because it would take a lot more time to run experiments on 23 classes than 3 classes but classifying speech into 23 and 3 classes are algorithmically equally challenging, as they both require multi-class classifiers.

Feature Extraction: MFCC and PLP

To convert an array of sample amplitudes from sound files to a more useful reporesentation, the first features we used are Mel frequency cepstral coefficients (MFCCs). This is the most popular feature in speech recognition. MFCCs separate the samples into small windows because the whole sample arrays hold too much information. So, we divided the sound into small chunks of time. We run Discrete Fourier Transform(DFT) on each window to get the power spectral, which holds frequency information of the signal over time. Then we convert the resulting spectral into values on the mel-scale, which is a perceptual scale of pitches judged by

human listeners. Next, we do a discrete cosine transform of the log mel-scale values, which gives the vector of MFCCs. We used the first 12 coefficients as our base feature, because using all the coefficients would have given us too many features. We also added an energy of the window to the vector, resulting in 13 features. We then appended the vector with delta, and delta delta (first and second order derivatives of the MFCC) of the vector to get 39 features in the end. This accounts for changes in the base features.

The other feature that we tried is Perceptual linear prediction(PLP). PLP is similar to MFCC. They both separate the signal into small windows - we used equal window size for both PLP and MFCC so that we could combine the features - and run DFT to get the power spectral. PLP then does critical-band spectral resolution by converting the spectral into values on the Bark scale. Next, PLP does pre-emphasis based on loudness, runs inverse DFT, and does Linear predictive analysis (LPC) on the result. In our experiment, we did PLP to order 12. So, we got 13 features as we also added energy to the PLP vector. We then computed delta, and delta delta to get 39 features just like with MFCC.

To conclude, for each sound sample, our features are for a list of windows. In each window, we have 39 features from MFCC and 39 other features from PLP. The number of windows is different for each sound sample as they are of different lengths, but the window size is fixed.

Classifier: Support Vector Machine (SVM)

Our classifier is based on windows, instead of the whole sound sample, as we have features for each window. The classifier tries to determine which acccent a window belongs to. This is done by combining all the windows in the training data and labeling each window by accent of the sound sample file the window belongs to. Then we simply trained SVM on these windows. To predict the accent of a sound sample, we use SVM to label each window in that sample and then we pick the most frequent label as the label for that sample. This approach sounds very simple, but it ended up not working. We tried this on just 12 MFCC features (without energy, deltas, PLP) and it took forever to train. One reason for that is that we have about 2000 windows per sound sample and about 300 samples per accent. So, there is a lot of data.

We tried to make SVM run in a reasonable amount of time by throwing away most of the windows for each sound sample randomly before we trained the SVM. For 39 MFCC features with 10 windows per each sound sample, SVM took an hour to train. The overall accuracy was 48.16%. Doing the same thing on 39 PLP features, the accuracy was 41.18%.

The result we got is better than what would be expected from randomly labeling the test set. However, randomly picking 10 from 2000 windows to use for training is not a reliable method. We wanted to find a reliable method that could run in a reasonable time.

Classifier: Gaussian Mixture Models (GMMs)

It turns out that we were able to use GMMs to do this efficiently. We did this by having 3 GMMs - one for each accent. We trained the GMM of each accent class by using the features extracted from the windows of the samples from that accent class as our training data. We predicted the accent class of the testing data by calculating the probability that the testing data belongs to each accent class and then outputting the accent class that gave the highest probability. To calculate the probability of testing data belonging to an accent, we did feature

extraction on the testing data first to get the same windows and the same features as before. We used the trained GMM for a particular accent to calculate the probability of each window belonging to that accent. Then we multiplied, or summed the logs of, the probabilities of the windows together to get the probability of each sample having the accent. In this way, we found which accent is most probable for that sample.

We trained GMMs with the EM algorithm. We initialized EM by using the mean from k-means and using a covariance matrix outputted by k-means - this seemed to get better results than initializing the covariance matrix to be the identity. One big problem we encountered was that our covariance matrix became singular often.

Because we were working with many features, we ran into a lot of problems with the covariance matrices becoming singular, or rather close enough to singular that floating point errors would occur. This comes about because if many entries in the covariance matrix are small, the determinant becomes very small as it is the sum of the products of entries of the covariance matrix. We mainly dealt with this problem in two ways. First, we switched from using full covariance matrices to only using diagonal covariance matrices, which were simpler. It turns out that we get better results by using the diagonal matrices as our covariance matrix. The other method we tried is to apply variance limiting on them. Variance limiting is when we do not allow any of the variances to be smaller than a certain value in magnitude, i.e. every entry of the diagonal of the covariance matrix must be larger than a certain value in magnitude. The second way we dealt with the problem is linearly scaling up the numbers that we were using as the features. This helped because linearly scaling should not change the results that the GMM gets us, but it should increase all of the variances, making it less likely that the covariance matrix will be singular.

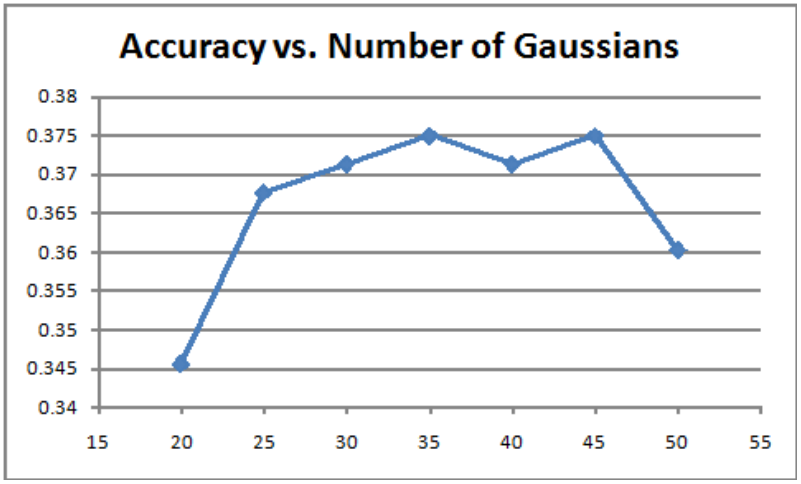
Using both MFCC and PLP

We tried to model on just MFCCs, just PLP, and also tried combining the two. When we combined PLP and MFCCs, we used 2 GMMs for each accent - one for MFCCs and one for PLP. We dealt with these two separately and we combined the result by multiplying the probabilities we got from each of them.

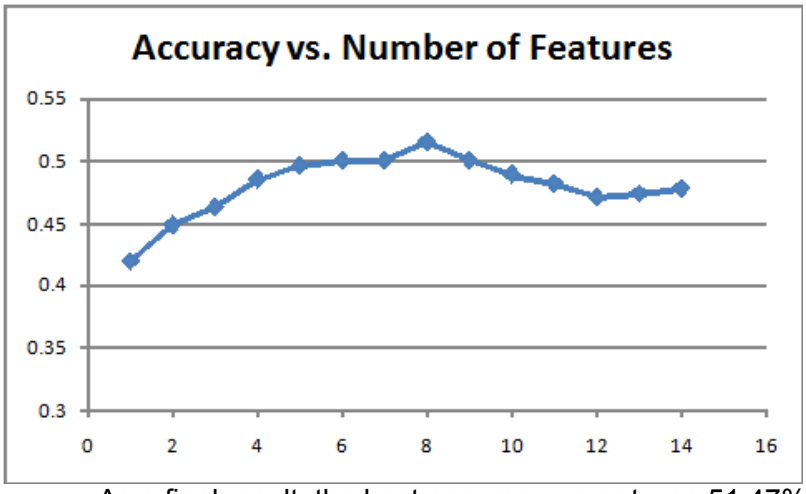
Another way we could have combined both features was that we could have just concatenated the vectors to get 78 features for each window. However, that seemed to be too many features for us. Therefore, we did feature selection using forward search on the 78 features to select just a few features to use. We picked forward search instead of other feature selection techniques because it is the fastest one.

Configuring our learning model

We tried to find the best number of gaussians to use for the GMMs for each accent class by training on the 39 MFCC features with different numbers of gaussians. The results are shown in the following diagram. We found that it was best to use 35 gaussians for each accent class. However, the effect that the number of gaussians on accuracy is very small.



For feature selection, we used forward search to select features from the 78 PLP and MFCC features. The classifier that we used was one GMM with 5 gaussians for each accent class. We did not use the number of gaussians we got from the testing above because we tried these two things in parallel. We picked 5 because we wanted the number of gaussians to be small to make feature selection run faster. We know that the impact of the number of gaussians is very small, so this should not have posed much of a problem for the accuracy of our feature selection. In any case, the feature selection process took a long time to run. The results are shown in the following graph. We can see that starting from around six features, the change in the accuracy is no longer strictly increasing, and it also becomes negligible. We found it surprising that more features would not help us. We still believe that six features is not enough; however, our experiment says otherwise. We do not think we overfitted the training data because we still got similar results, that is accuracy of .4 (using 39 MFCC features just like for testing the number of gaussians), when we used the same training and test data. Therefore, the problem probably lies in the features that we used.



As a final result, the best accuracy we got was 51.47% using 8 features. The forward selection picked 7 of the 8 features from MFCCs.

Future Work

The most problematic part of our research is that our features do not model different

kinds of accents well. MFCCs and PLP are suited for general audio, not specific to voice or accent. For the future, we would like to get more specific features. We considered trying prosodic features, as many papers suggested this; however, we need to be able to recognize words in the sound sample to be able to apply prosodic features. This is because we would have to label words in the dataset, which takes a lot of time if it is done manually. Or, we would have to do speech recognition before running our classifier to determine where words are, which seems inappropriate for our case given that speech recognition does not run well on accented speech, and one of the purposes of accent classification is to aid in speech recognition. Therefore, it was not feasible for us given the limited time we had. However, it might be worth considering in the future.

One other thing that we failed to take into account is pauses. There are a lot of windows which are just pauses. These pauses are the same for every accent. So, the features from these windows do not help in classification.

We could also try decreasing variation on our dataset by only using male or female voice or dividing the dataset into male and female voices given that pitch is often very different in male and female speech.

References

Dan Jurafsky.

Hong Tang and Ali A. Ghorbani, “Accent Classification Using Support Vector Machine and Hidden Markov Model”. University of New Brunswick

Ghinwa Choueiter, Geoffrey Zweig, and Patrick Nguyen, “An Empirical Study of Automatic Accent Classification”

Karsten Kumpf and Robin W. King,”Automatic Accent Classification of Foreign Accented Australian English Speech”. Speech Technology Research Group, The University of Sydney NSW 2006, Australia

Scott Novich, Phil Repicky, andAndrea Trevino, “Accent Classification Using Neural Networks”, Rice University. <http://cnx.org/content/col10320/1.1/>

John H.L. Hansen and Levent M. Arslan, “Foreign Accent Classification Using Source Generator Based Prosodic Features”. Robust Speech Processing Laboratory, Duke University.

Douglas A. Reynolds, and Richard C. Rose, “Robust Text-Independent Speaker Identification Using Gaussian Mixture Speaker Models”, IEEE.

Muchael J. Carey, Eluned S. Parris, Harvey Lloyd-Thomas, and Stephen Bennet, “Robust Prosodic Features For Speaker Identification”. Ensigma Ltd.

Andre G. Adami, Radu Mihaescu, Douglas A. Reynolds, and John J. Godfrey. “Modelling Prosodic Dynamics For Speaker Recignition”.

Hynek Hermansky, “Perceptual linear predictive (PLP) analysis of speech”, Speech Technology Laboratory, Division of Panasonic Technologies Inc., Santa Babara CA