Text-Independent Speaker Identification

Til T. Phan and Thomas Soong

1.0 Introduction

1.1 Motivation

The problem of speaker identification is an area with many different applications. The most practical use can be found in applications dealing with security, surveillance, and automatic transcription in a multi-speaker environment.

Speaker identification is a difficult task, and the task has several different approaches. The state of the art for speaker identification techniques include dynamic time warped(DTW) template matching, Hidden Markov Modeling(HMM), and codebook schemes based on vector quantization(VQ)[2]. In this project, the vector quantization approach will be used, due to ease of implementation and high accuracy [7].

1.2 Background

In order to implement the text-independent speaker ID system, one must go through several steps, including feature extraction, feature matching, and finally, identification of the speaker. Feature extraction is a method that takes a small amount of data from the voice signal which can later be used to generate a representation of each speaker. Feature matching

involves the actual procedure of using vector quantization to identify the speaker according to the characteristics of the known speakers. We will discuss each procedure in detail in later sections.

1.3 Goals

The goal of this project is to develop and implement an text-independent speaker identification system. The system should be able to identify speakers based on the different voice characteristics of each of the known speakers. This identification should be accomplished regardless of the sentence spoken.

2.0 Feature Extraction

2.1 Introduction

Feature Extraction is the means by which speech data is reduced to much smaller amounts of data which represent the important characteristics of the speech. Many varieties of features can be used for speech processing, such as LPC coefficients, Mel Cepstrum, spectrograph, and others. LPC coefficients are perhaps the best known and most popular, and these will be used in this project.

2.2 LPC Coefficients

Speech can be (and often is) modeled by a series of pulses from the larynx, passing through an all-pole transfer function representing the effect of the vocal tract on the waveform [6]. *Linear Prediction Coefficients*, or LPC, represent the specific locations of the poles of the transfer function (see EQ 1), and can be efficiently computed. Specifically, LP coefficients are the result of attempting to predict each speech sample as a linear combination of a certain number of previous samples. The weights used in this combination are the coefficients.

$$H(z) = \frac{G}{1 - \sum_{i=1}^{p} a_i z^{-i}} \qquad \textbf{(EQ 1)}$$

The LP coefficients are typically adaptively computed for short time intervals over which time invariance is assumed [4]. These frames are usually between 10ms and 30ms long, and usually windowed by a Ham-

ming window or a similar windowing function. The two computation methods, known as the *autocorrelation* method and the *covariance* method both attempt to minimize the mean-square value of the estimation error, given by

$$e(n) = s(n) - \sum_{i=1}^{p} a_i s(n-i) \quad \textbf{(EQ 2)}$$

The autocorrelation method is the most common approach to determining the LP coefficients. The autocorrelation of *s(n)* is defined as $r_s(k)$ in Eq 3. and the predictor coefficients can be found by solving the matrix in Eq 4. [4].

$$r_s(k) = \sum_{n=0}^{N-1-k} s(n)s(n+k) \quad \textbf{(EQ 3)}$$

$$\begin{bmatrix} r_s(0) & r_s(1) & \dots & r_s(p-1) \\ r_s(1) & r_s(0) & \dots & r_s(p-2) \\ \dots & \dots & \dots & \dots \\ r_s(p-1) & r_s(p-2) & \dots & r_s(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \dots \\ a_p \end{bmatrix} = \begin{bmatrix} r_s(1) \\ r_s(2) \\ \dots \\ r_s(p) \end{bmatrix} \quad \textbf{(EQ 4)}$$

This matrix equation can be efficiently solved via the *Levinson-Durbin Recursion*.

2.3 Other Features

The LP coefficients can be further transformed to improve accuracy of classification. One popular transformation produces *Cepstral Coefficients*, which are defined as the inverse FFT of the logarithm of the FFT of the LP coefficients. These coefficients have been shown [1] to provide the best results in a speaker identification application, as they are considered to be uncorrelated. The actual FFTs do not need to be explicitly computed, as there exists a recursive formal for this computation, given as [4]

$$c_{lp}(n) = a_n + \sum_{i=1}^{n-1} \left(\frac{i}{n}\right) c_{lp}(i) a_{n-1} \quad \textbf{(EQ 5)}$$

The cepstral coefficients can further be weighted to emphasize certain coefficients and de-emphasize others. This process is known as *liftering* and also helps reduce the effects of noise. One approach uses simple linear weighting, also known as *frequency liftering*. A more complex

approach uses a raised sine window. This is called *bandpass liftering* (BPL) and is defined as:

$$w(n) = \begin{cases} 1 + \frac{L}{2}\sin\left(\frac{n\pi}{L}\right) & n = 1, 2, \ldots, L \\ 0 & otherwise \end{cases} \quad \textbf{(EQ 6)}$$

2.4 Implementation

The voice data is sampled at 16000 Hz, and is split up into frames of 240 samples, which corresponds to 15ms. The frames overlap by 80 samples, meaning there is a frame every 10ms. Each frame is run through a simple filter with transfer function

$$H(z) = \frac{1}{1 - 0.9375z^{-1}} \quad \textbf{(EQ 7)}$$

for the purpose of pre-emphasizing the high frequencies of the speech. The frame is then multiplied by a Hamming window, and 12th order LPC autocorrelation analysis is run on it.

3.0 Speaker Classification

3.1 Introduction

In a speaker identification system, each speaker must be uniquely represented in an efficient manner. The means to do this is called *vector quantization*. Vector quantization is a process of mapping vectors from a large vector space to a finite number of regions in that space. The data is thus significantly compressed, yet still accurately represented. Without quantizing the feature vectors, the system would be too large and computationally complex. In a speaker recognition system, the vector space contains a speaker's characteristic vectors, which are obtained from the feature extraction described above. After vector quantization takes place, only a few representative vectors remain, collectively known as that speaker's *codebook*. The codebook then serves as a delineation for the speaker, and is used when training a speaker in the system.

3.2 Vector Quantization

In our system, we are quantizing about 1200 feature vectors down to 128 codebook vectors. These vectors are individually known as *centroid vec-*

tors. Ideally, a centroid vector should represent a cluster of feature vectors. The goal is to obtain 128 vectors such that the overall distortion (Euclidean distance) from each feature vector to its nearest centroid in the vector space is minimized. With minimal distortion, an accurate representation of the speaker can be obtained.

3.3 Codebooks

There are several different approaches to finding an optimal codebook for a speaker. The idea is to begin with a vector quantizer and a codebook and improve upon the initial codebook by iterating until the optimal one is found. The major problem was generating the initial codebook of 128 vectors.

3.3.1 Binary Splitting

The first method we tried is called *binary splitting*. One begins with one centroid vector, which is the centroid for the entire set of training vectors. From there the centroid is split into two by multiplying the vector by certain factors. The new large codebook is optimized according to the *k-means* algorithm described below. The process is repeated until the size desired is obtained. We used the mean of each vector dimension to come up with the first centroid. Then we multiplied that centroid by two factors, (1+*e) and* (1-*e)*, to get the two new centroids. *e* is usually in the range of 0.01 to 0.05. Then we used the k-means algorithm (see below) to get the best set of centroids for the split codebook. These steps were repeated until a 128-vector codebook was obtained. We found that this method returned a very poor representation of the speaker's feature vectors, even after optimization [3].

Our next attempt was simply to choose 128 random vectors from the feature set and optimize those iteratively. This method is called *random coding*, and was found to be much more effective than binary splitting.

3.3.2 Optimization with K-means

We selected the iterative improvement algorithm known as *k-means* (also known as the LBG or the generalized Lloyd algorithm). Given a set of I training feature vectors, $\{\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_I\}$ characterizing the variability of a speaker, we want to find a partitioning of the feature vector space, $\{S_1, S_2, ..., S_M\}$, for that particular speaker where S, the whole feature space, is represented as $S = S_1 \cup S_2 \cup ... \cup S_M$. Each partition, S_i, forms a nonoverlapping region and every vector inside S_i is represented by the corre-

sponding centroid vector, $\mathbf{b}_i$, of S_i [3]. Each iteration of k-means moves the centroid vectors such that the accumulated distortion between the feature vectors is lessened. The more iterations you run, the less distortion you should have.

The algorithm takes each feature vector and compares it to every codebook vector which are closest to each. That distortion is then calculated for each codebook vector *j* as:

$$D_j = \sum_{i=1}^{I} (t_i - v_{j,i})^2 \qquad \textbf{(EQ 8)}$$

where *v* are the vectors in the codebook and *t* is the training vector[11]. The minimum distortion value is found among all measurements. Then the new centroid of each region is calculated. If *x* is in the training set, and *x* is closer to v_i than to any other codebook vector, assign *x* to C_i. The new centroid is calculated as where C_i is the set of vectors in the training set that are closer to v_i than to any other codebook vector [8]. The next iteration will recompute the regions according to the new centroids. The total distortion will now be smaller. Iteration continues until a relatively small percent change in distortion is achieved.

3.4 Training

Each speaker records several training sentences, which are concatenated and from which features are extracted. The accumulated feature vectors are used to generate a codebook according to the algorithm described above. This codebook is used for identification.

3.5 Matching

The speaker identification system works by taking the feature vectors of an arbitrary input from one of the trained speakers and comparing them with all the codebooks in an *exhaustive search*. First, feature extraction is applied to the unknown speaker's input sample. Then, for each known speaker, each vector from the test utterance is quantized to that speaker's codebook, and the distortion involved in doing so is saved. The entire test utterance is evaluated this way, and the sum of the distortions of each frame represent the quality of the match. This process is repeated for each speaker, and the one with the least total distortion is chosen as the speaker [3].

4.0 Testing

4.1 Testing

To test our system, we used MatLab. From our large dataset, we divided it into two partitions. One partition is used to train the system and the other partition is used to verify the system. The results of the can be seen from the graphs in our presentation slides. In addition, we came to the conclusion that feature extraction of the dataset was successful due to the smaller number of data points on the graphs generated by MatLab.

4.2 Results

The overall implementation of the speaker identification system was very interesting. The MatLab implementation of feature extraction was very straight forward. This was due to our knowledge of the MatLab software. The results can be found in our presentation slides.

After the MatLab implementation was done, we attempted to model the speaker system with Ptolemy. We decided to model the feature extraction using a synchronous dataflow model. The overall system can be found in our presentation slides. But in a nutshell, we used stars that were available in the SDF domain and connected them together and attempted to code the stars up for our application. Due to our lack of experience with Ptolemy, we were not succesful to obtain any substantial results.

5.0 Conclusions

The vector quantization approach to speaker identification is an efficient and accurate approach to the problem. The accuracy rate could be improved by some of more complex voice features mentioned above. Overall, the system meets the original goals.

6.0 References/Bibliography

1. B. Atal “Automatic Recognition of Speakers from Their Voices”. Proceedings of the IEEE, vol. 64, April 1976, pp. 460-475.
2. Y. Bennani, “Multi-Modular and Hybrid Connetionist Approaches for Pattern Recognition: Speaker Identification Task”. Carnegie Mellon Unviversity Department of Electrical and Computer engineering.
3. A. Gersho, R. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic Publishers, Boston, 1992.
4. R. Mammone, X. Zhang, R. Ramachandran, “Robust Speaker Recognition”, IEEE Signal Processing Magazine, September 1996.
5. Linguistics Data Consortium Web Site, http://www.ldc.upenn.edu/ldc/
6. J. Markel, A. Gray, *Linear Prediction of Speech,* Springer-Verlag, New York: 1976.
7. K. Shikano, “Text-Independent Speaker Recognition Experiments using Codebooks in Vector Quantization”. CMU Dept. of Computer Science, April 9, 1985.
8. P. Silsbee, “ECE 696 Fall 1996 Homework 5”, http://www.ee.odu.edu/~silsbee/ece696/hw5_f96/hw5_f96.html, Old Dominion University, 11 Nov, 1996.
9. F. Soong, E. Rosenberg, B. Juang, and L. Rabiner, “A Vector Quantization Approach to Speaker Recognition”. AT&T Technical Journal, vol. 66, March/April 1987, pp. 14-26.
10. Texas Instruments Web Site, http://www.ti.com/
11. J. Wiejaczka, et. al., “Word Recognition for the Puzzle Project”, http://sig2.colorado.edu/~puzzle/wordrecog/wordrecog.html#VQ.
12. K. Shikano, “Evaluation of LPC Spectral Matching Measures for Phonetic Unit Recognition”. Carnegie Mellon Dept. of Computer Science, Feb 3, 1986.