REHEARSAL AUDIO STREAM SEGMENTATION AND CLUSTERING

Dawen Liang dawenl@andrew.cmu.edu
Guangyu Xia gxia@cs.cmu.edu
Mark Harvilla mharvill@cs.cmu.edu

Abstract

Rehearsal recordings are valuable for music ensembles to improve their performance. However, since rehearsal recordings are typically hours long and contain disordered, disrupted, and unclassified musical content (mixed with different sections of different music pieces, conductor's talk, all kinds of noise between rehearsal intervals and so on), they're hard to use directly. For example, recordings for a particular song are not easily found. This paper presents a variety of approaches to extract all the musical segments from the disarrayed rehearsal audio stream and then cluster the segments belonging to the same piece of music together. The procedures discussed herein have the potential to be applied in a large scale music database to accomplish the music information retrieval (MIR) tasks.

1 Introduction

The rehearsal audio stream segmentation and clustering task can be divided into two subproblems. The first one is to extract the musical sections from the rehearsal audio stream. That is, an algorithm that automatically and effectively discriminates between musical and non-musical segments of audio must be employed. The second step is to cluster the different segments belonging to the same music piece together. Multiple complementary approaches to both subproblems were considered. For example, we show that the first subproblem is solvable by applying principal component analysis (PCA) to music, leading to the basic but novel concept of *eigenmusic*. Similarly, we obtained successful discrimination results using support vector machine (SVM). For the second task, the feature used to represent the music segments is critical. Fundamentally, grouping together segments of music belonging to the same song is a clustering problem, but one with a perceptual basis. Finally, given a useful representation of the audio stream's musical segments, the actual clustering task is achievable using any number of typical clustering techniques, such as correlation or k-means, with varying rates of success.

2 Discrimination of Music and Noise

In this system, discrimination of music and noise is framed as a binary decision problem: A given segment of audio is either music, or it is not. We contend that a high quality binary classifier should be able to capture as music those clips that a human judge would also likely regard as music, despite a reasonable presence of noise.

2.1 Eigenmusic in the time-domain

The concept of *eigenmusic* is derived from the well-known representation of images in terms of eigen*faces*. The process of generating eigenmusic can be performed in either the time- or frequency-domains, and in either case, simply refers to the result of the application of PCA to audio data. The PCA results in a set of uncorrelated principal components which are able to well-represent data similar to that from which they were derived. The functionality of PCA is rooted in the symmetry of the

covariance matrix of a set of random variables. Formally, let $\mathbf{X} = [X_1, X_2, ..., X_n]^T$ be a random vector with covariance matrix $C_\mathbf{X}$. It can be shown that if P is the matrix of eigenvectors of $C_\mathbf{X}$ in its columns and $\mathbf{Y} = P^T\mathbf{X}$, then $\mathbf{Y}$ is an uncorrelated random vector. That is, the covariance matrix of $\mathbf{Y}$ is diagonal. The vector $\mathbf{Y}$ contains the principal components of the elements of $\mathbf{X}$ [1]. Assuming n dimensional observation vectors and m total observations, the covariance matrix can be estimated as in [2]. From a conceptual standpoint, PCA tries to reduce the dimensionality of a data set representation by expressing the data in terms of the k largest eigenvectors (that is, the k eigenvectors associated with the k largest eigenvalues of the data's empirical covariance matrix). *Eigenmusic* refers to the eigenvectors of an empirical covariance matrix associated with an array of music data. If it's performed in the time-domain, the array of music data literally contains segments of audio from sequential time intervals. In the frequency domain, the array of music data is structured as a spectrogram and hence contains the spectral information of the audio in those time intervals. When expressing noise data in terms of eigenmusic, the result is generally expected to be outlying based on the fundamentally different characteristics of music and noise; this fact is exploited in classification.

2.2 Eigenmusic in the frequency-domain

PCA is applied to the audio's magnitude spectrum after the STFT. First, all the audio is resampled to 16kHz. An FFT window size of 4000 samples (0.25 second) with no overlap is used. Every frame of the magnitude spectrum is normalized to minimize volume differences between music and non-music. Finally, features are averaged every 5 windows (1.25 seconds) since the musical characteristic are better reflected within a longer period of time and empirically it has been shown that 1.25 seconds is a good choice. Similar to 2.1, $\mathbf{X}$ would be the matrix in which each column represents the average magnitude at different frequencies within 1.25 seconds. PCA is performed to reduce the dimensions of $\mathbf{X}$ by representing it in terms of eigenvectors. The top k eigenvectors ($k = 10$) are used as bases to represent the audio stream. Each frame of audio stream (1.25 seconds) is represented as a liner combination of these 10 eigenmusic bases, as in $P\mathbf{Y} = \mathbf{X}$. The features in $\hat{\mathbf{Y}}$ are the weights of the linear combination.

2.3 ADABoost

ADABoost is adopted as the classifier learning algorithm [3].

During training, more than 40 hours of Western music rehearsal audio is used as matrix $\mathbf{X}$ to compute the eigenmusic P. Then matrix $\mathbf{Y}$, used as training data for ADABoost, is computed as discussed in 2.2. 100 weak learners are trained to build the strong classifier.

An approximately 5 hours audio stream of labeled music (both Western and Chinese) and noise is used for testing. From the result, we found that 100 weak learners overfit the data. After trying out different numbers of weak learners, we finally chose 30 weak learners to build the strong classifier for evaluation. The test results will be discussed in 2.5

Since the classifier may make mistakes and we know that the audio stream will not change from music to noise (and vice-versa) so abruptly and frequently, a simple sliding window method is adopted to smooth the results for post-processing. Here we make some reasonable assumptions: there will not be any music segments less than 30 seconds or any noise segments less than 5 seconds (Even if such segments exist, they are nearly meaningless for our system):

1. Cut the segments longer than 5 seconds, in which each frame is classified as noise by the strong classifier;

2. Retain the segments longer than 30 seconds, in which 75% of the frames are classified as music; at the same time, any music segments which are less than 1 minute should be classified as at least 90% music. Otherwise discard it.

3. Any segments between 2 music segments with the length of at most 5 seconds will also be classified as music.

2.4 Support Vector Machine using Marsyas

Marsyas [4] is an open source software framework for audio processing with specific emphasis on Music Information Retrieval applications. Our main purpose is to compare the results using the most commonly used features and classifier with the results we get from eigenmusic. Here we use the Marsyas command `bextract` to extract the typical timbral features to distinguish music from non-music. After normalizing the sampling frequency to 22.05kHz, the features are extracted using 512-sample analysis windows with no overlap. Then, features are averaged within roughly one second. The classifier we choose here is SVM with Sequential Minimal Optimization (SMO) [5] algorithm implemented by Weka [6]. The results will be discussed in 2.5.

2.5 Experimental Results

Experimental results obtained using ADABoost in combination with eigenmusic in the time-domain were subpar. The overall classification error rate was determined to be 31.26% with the system trained on a subset of labelled rehearsal audio containing both music and noise. The error rate for music was 33.64% and the error rate for noise was 9.8%. This indicates a higher false negative rate and generally poor performance. In contrast, normalized eigenmusic in the frequency-domain was found to be the most effective method of discriminating between music and noise, with around 6% classification error on both music and noise versus 2.7% classification error on music and 41.3% classification error on noise using SVM. Relatively, the performance of eigenmusic classification in the time-domain is poor and emphasizes that the time-domain representation of audio is not conducive to the identification of the distinguishing characteristics of music and noise. As intuition would suggest, eigenmusic in the frequency-domain provides a much more robust ability to discriminate between the classes.

3 Clustering of Musical Segments

Assuming perfect classification results from the previous step, the clustering task is a distinct problem. Note that any classification errors initially made will unintentionally degrade the performance of the clustering algorithm employed here. This underscores the importance of highly accurate classification results, and on a broader scale, instantiates the dependence of each system component on the others.

3.1 Feature Representation

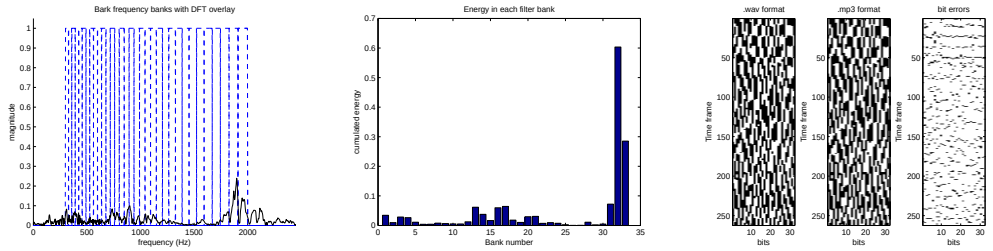
As noted in the introduction, the preeminent qualifying characteristic of the feature chosen to represent the musical segments should be the measure of its ability to capture visceral qualities of the input data. As evidenced by the use of frequency-domain approaches employed in Section 2.2, heeding these perceptual attributes can be expected to result in marked performance improvements. The features considered for this task are chosen with these facts in mind, and attempt to strike a compromise between efficiency and effectiveness.

3.1.1 Audio Fingerprinting

The *audio fingerprint* is an audio feature extraction method proposed by [7] and intended to be used for such things as searching an audio database for a certain file given only a possibly distorted and partial example. Distortion might include things like background noise, nonlinear rate differences, and mismatched file formats. The audio fingerprint is a possibly ideal feature to set the stage for successfully clustering the musical portions of the audio stream. The audio fingerprint was implemented as described below and is available as `audio_fingerprint.m`; this script also relies on a few other related routines that were all included in the report submission.

3.1.2 The Fingerprint Extraction Algorithm

As more thoroughly outlined in [7], the algorithm consists of five primary steps. The first of these steps is signal decimation followed by division of the incoming audio stream into segments. A 32-bit binary *sub-fingerprint* is derived for each. The signal is decimated to a sampling frequency around



(a) (b) (c)

Figure 1: 1(a): The 33 non-overlapping bandpass filters (blue) encapsulating an example DFT (black) for processing. The energy under each specific filter passband is cumulated as the output of that band. 1(b): The cumulated energy in the signal within each filter bank passband with reference to 1(a). The time and frequency variations in these energy values are used to characterize the audio data. 1(c): Comparison of bit error rate (BER) for audio fingerprints of the same music data encoded in different formats.

5kHz, for reasons explained later. The sequence containing the minimum number of sub-fingerprints such that the data is identifiable by it is a so-called *fingerprint block*. Based on the strategy of [7], 256 subsequent sub-fingerprints equal a fingerprint block. For each segment of music tagged by the classifier in the first step, only these fingerprint blocks need to be retained. The frames have a fixed length of 0.37 seconds and overlap one another by a factor of $\frac{31}{32}$.

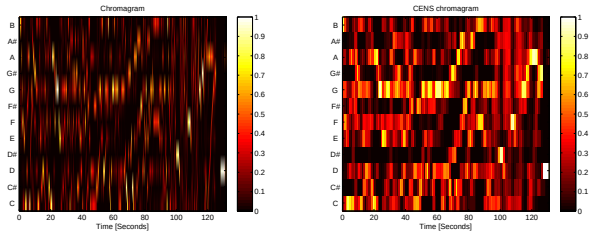
The discrete Fourier transform is applied to each frame resulting in a spectrogram of approximately 3.4 seconds in duration. The phase is discarded. A filter bank is then applied to the spectrogram. The philosophy of the use of the filter bank is to replicate the way in which audio is perceived by humans, based on the tonotopic functionality of the basilar membrane, which can be modeled as an array of overlapping band-pass filters [8]. Motivation for the choice of the filters' bandwidths is based on experimental results such as [9] which suggest that the perceptual relation of pitches to one another is approximately logarithmic. In our implementation of this algorithm, the applied filter bank consists of 33 non-overlapping bandpass filters, each of whose passbands are constant in *bark frequency* [10]. Haitsma and Kalker suggest that the "most relevant spectral range for the human auditory system" is 300Hz to 2kHz–this is the range processed by the bark filter bank. Within this relevant frequency range, the formula used for conversion from frequency f in Hertz to f_b in bark is $f_b = \frac{26.81f}{1960+f} - 0.53$ based on [11]. An example of such a filter bank and its corresponding output are shown in Figure 1. Haitsma and Kalker indicate that the sign of energy differences across time and frequency is a robust property that is very useful for signal characterization. In accordance with their scheme, 32-bit sub-fingerprints are extracted from the 33 filter bank outputs for each time frame of the signal within an arbitrarily positioned interval of approximately 3.4 seconds (our implementation uses the first 3.4 seconds of the input clip). Suppose $F[n,m]$ is the m^{th} bit of the n^{th} time frame of the input, $E[n,m]$ is the cumulated energy in the m^{th} filter bank of the n^{th} time frame, N is the number of time frames in the signal, and $A = E[n,m]-E[n,m+1]-(E[n-1,m]-E[n-1,m+1])$, then

$$F[n,m] = \begin{cases} 1 & \text{if } A > 0 \\ 0 & \text{if } A \leq 0 \end{cases} \quad m = 0,1,...,32; n = 0,1,2,...,N \tag{1}$$

In the above, $E[-1,m]$ is taken to be zero. One can consider A to be the change in energy-differences-with-respect-to-frequency over time. As an example of the functionality of our implementation, Figure 1(c) compares two audio fingerprints of the same audio snippet and displays their bit errors. This image is visually comparable to Figure 2 of [7] which reinforces the veracity of our realization.

3.1.3 Chroma Feature

Chroma features have been widely used as a robust harmonic feature in all kinds of MIR tasks. On a piano keyboard, there are 12 chromatic notes in one octave (C, C#, D ... A#, B, often refered as pitch class). The chroma-based approach proposed by Bartsch and Wakefield [12] represents the spectral energy contained in each of these 12 pitch classes. Such features strongly correlate to



(a) (b)

Figure 2: The traditional chromagram 2(a) and CENS features 2(b)

the harmonic progression of the audio. As suggested by Müller [13], the chroma feature can be improved to reduce the effect of timbral factors by discarding the lower MFCCs which are generally considered closely related to timbre.

Considering the objective that our system should be robust to some external factors, like audience applauding, we don't want our feature too sensitive to these minor variations. Therefore, in addition to calculating the 12-demensional chroma feature with a small window (200 ms with 50% overlap), we perform a second windowing which is larger (here we choose a window of 41 consecutive chroma vectors) and normalize the vectors through downsampling by factor of 10 to represent the statistics during a relatively long period of time (roughly one second), described as *CENS features* (**C**hroma **E**nergy distribution **N**ormalized **S**tatistics) [14], which are elements of the set $\mathcal{F} = \{\vec{v} = (v_1, v_2, \ldots, v_{12}) | v_i \geq 0, \sum_{i=1}^{12} v_i^2 = 1\}$. The length of the second window and the downsampling factor can be changed in order to take the global tempo variations into consideration. This will be discussed later in 3.2.2. In summary, this procedure has two advantages: First, it could diminish the effect of short-time variance that we expect might be problematic; second, it significantly reduces the size of data and therefore increases the speed of our system for clustering. Figure 2 shows both the traditional chromagram and the corresponding CENS features. We can see clearly that compared with the chromagram, CENS has a larger block and therefore better contrast. CENS is used later as features for correlation in 3.2.2.

3.2 Grouping Similar Songs

To generate the overall output of the system, accurate clustering of the musical segments is required. This step should yield N groups of k_i audio clips, where N is the number of songs present in the decoded stream and k_i is the number of clips belonging to the i^{th} song, for $i = 1, 2, ..., N$. In general, k_i is immaterial and hinges on the total length of each individual song, as well as the segment size, S.

3.2.1 K-means

Given an adequate representation of the musical segments of the audio stream, it is reasonable to expect a useful result from a basic clustering algorithm like *k-means* ([15, 16]) to group together those clips belonging to the same song. The k-means algorithm is principally applicable as a component of a vector quantization (VQ) process. The k-means algorithm can be used to design the codebook for VQ by partitioning a set of training data into M cells and associating a quantized vector (codeword) with each cell [17]. K-means partitions a data set such that the following two optimality criterion are satisfied: $q(\mathbf{x}) = \mathbf{z}_i$ iff $i = \arg\min_j d(\mathbf{x}, \mathbf{z}_j)$ and $D_i = \Pr(\mathbf{x} \in C_i) E[d(\mathbf{x}, \mathbf{z}_i) | \mathbf{x} \in C_i]$ where $q(.)$ is the quantization function, $\mathbf{x}$ is an input vector, $\mathbf{z}_i$ is the codeword representing the i^{th} partition of the vector space, $d(\mathbf{x}, \mathbf{z}_i)$ is the distance measure between the input and the codeword of the i^{th} partition, and D_i is the total average distance (or distortion) in the i^{th} partition. Applying k-means to the features outputted from the previous system block is a simple iterative procedure that follows the generic steps given in [18]. We use the Hamming as the distance function $d(\mathbf{x}, \mathbf{z}_j)$ which is applicable to the *binary* audio fingerprint.

3.2.2 Audio Matching and Clustering by Correlation

Given CENS features, audio matching can be achieved by simply correlating the query $Q = (\vec{v}^1, \vec{v}^2, \ldots, \vec{v}^M)$ with the sliding subsequences of musical segments $D = (\vec{u}^1, \vec{u}^2, \ldots, \vec{u}^N)$ in database ($N \gg M$), defined as $D^{(i)} = (\vec{u}^i, \vec{u}^{i+1}, \ldots, \vec{u}^{i+M-1}), i \in [1, N-M+1]$. The distance between the query Q and the sliding subsequence $D^{(i)}$ can be computed by *dist*$(Q, D^{(i)}) = 1 - \frac{1}{M}\sum_{k=1}^{M}(\vec{v}^k \cdot \vec{u}^{i+k-1})$. All these distances from $i = 1, \ldots, N-M+1$ together can be considered as a distance function Δ between query Q and the musical segments in database D. We could find the minima of this distance function Δ as our best matches. Unlike the traditional song retrieval system which has a large database in advance, our system has no information about the rehearsal audio stream beforehand. In order to solve this problem, we try to construct the database from the rehearsal audio dynamically. The input is all the music segments obtained from Section 2:

1. Sort all the music segments according to their length. Put them into a priority queue.
2. As long as the priority queue is not empty, extract the longest segment S from the priority queue.
   (a) If the database D is empty, put S into D as a cluster;
   (b) Otherwise match segment S with every segment D_i in D by measuring the distance function Δ.
      i. If there exists a D_m which contains segment S, cluster S with D_m, and go back to step 2;
      ii. Otherwise make S a new segment (cluster) in database D.

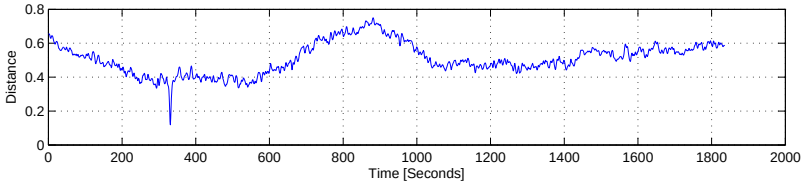
Here we made an assumption: the longest segment is the most likely to be a whole piece or at least the longest segment for this piece. So it is reasonable to let it represent a new cluster. On the other hand, using a priority queue will guarantee that the segment S is no longer than any of the segments in database D, which means it can either be part of an existing piece in the database (and we will cluster it with the segment it belongs to in 2(b)i) or it's a segment for a new piece which doesn't exist in the database (and we will make it a new cluster in 2(b)ii).

We also need to consider the tempo changes since the performers can never hold the tempo absolutely constant during a rehearsal. For this problem, we can obtain different versions (for example, from 20% slower to 20% faster) of CENS features for the same segment to represent the possible tempo variations. This can be achieved by adjusting the length of the second window and the downsampling factor in 3.1.3. More results will be discussed in 3.3 about the experiments.

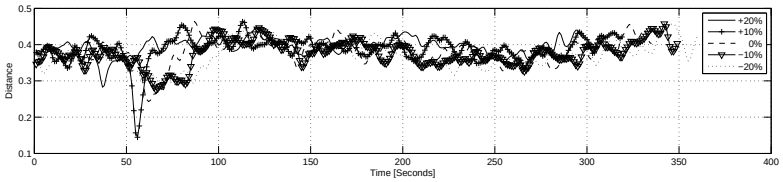
3.3 Experimental Results

Preliminary tests using k-means were not satisfactory enough to motivate application to a large test set. While intuitively pleasing, these preliminary tests show that Haitsma and Kalker's fingerprint does not yield a uniform representation of music with respect to which portion of the same overall audio is analyzed. Initially, using three arbitrary songs, six non-overlapping 10 second segments of each song were read, resulting in 18 different 10 second clips. Each 10 second clip was transformed to a fingerprint block, and then k-means was used to cluster the fingerprints. The result of the clustering was confusing at best. With $k = 3$, the fingerprints belonging to the first song were placed in the following clusters: $\{3, 2, 3, 2, 1, 3\}$. Similarly undesirable results were obtained for the other two songs. In all cases, there was at least one clip for each song in any one of the categories. Ultimately, it was determined that, for the particular songs being analyzed, the intervals over which the fingerprints are extracted cannot be offset by more than approximately 150 milliseconds from one another if they are expected to be grouped together using k-means. The implication of this is that the fingerprint feature is unacceptably sensitive to the time interval over which it is extracted. Ironically, the reputed robustness to this exact situation is one of the primary reasons the feature was so ambitiously exploited in the beginning of the work.

For audio matching by correlation, the results are satisfactory. We tested our approach in two different ways: First, we try to match a clip which is the subset of one song in the database. From Figure 3(a) we could see that there is a notch at the best matching position round 330s. Secondly, we try to match a clip which is 10% faster than its corresponding cluster in the database by using different tempo versions of the feature. We could see clearly from Figure 3(b) that only the 10%



(a)



(b)

Figure 3: Results for audio matching by correlation

slower version (the closest with the cluster in the database) has the most obvious notch, showing that using CENS for clustering can lead to really good results.

4 Summary & Conclusion

Based on experimental results, eigenmusic proves to be a robust feature for distinguishing music and noise; chroma features are a strong fit for matching audio clips efficiently. Results associated with k-means clustering of audio fingerprints were subpar. Analysis suggests that the fingerprint representation is sensitive to time shifts and therefore an impractical basis for grouping together different segments of similar songs. Generally, the project was successful, highly instructive, and will serve as a pertinent foundation for related work in the future.

References

[1] O. Tonguz, "Applied Stochastic Processes Lecture Notes, Lecture 11, CMU," Fall 2010.

[2] B. Raj, "MLSP Lecture Notes, Class 5, Slide 12, CMU," Fall 2010.

[3] Y. Freund and R.E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," *Journal of Computer and System Sciences*, 1997.

[4] George Tzanetakis and Perry Cook, "Marsyas: A framework for audio analysis," 2000.

[5] J. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods - Support Vector Learning*, B. Schoelkopf, C. Burges, and A. Smola, Eds. MIT Press, 1998.

[6] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten, "The weka data mining software: an update," *SIGKDD Explor. Newsl.*, vol. 11, pp. 10–18, November 2009.

[7] J. Haitsma and T. Kalker, "A Highly Robust Audio Fingerprinting System," *ISMIR*, 2002.

[8] R. Munkong and B.H. Juang, "Auditory perception and cognition," *IEEE Signal Processing Magazine*, pp. 98–117, 2008.

[9] S. S. Stevens, J. Volkman, and E. Newman, "A scale for the measurement of the psychological magnitude of pitch," *Journal of the Acoustical Society of America*, pp. 185–190, 1937.

[10] E. Zwicker, "Subdivision of the audible frequency range into critical bands," *The Journal of the Acoustical Society of America*, 1961.

[11] H. Traunmüller, "Analytical expressions for the tonotopic sensory scale," *The Journal of the Acoustical Society of America*, pp. 97–100, 1990.

[12] M. A. Bartsch and G. H. Wakefield, "Audio thumbnailing of popular music using chroma-based representations," *IEEE Transactions on Multimedia*, vol. 7(1), pp. 96–104, Feb. 2005.

[13] Meinard Müller, Sebastian Ewert, and Sebastian Kreuzer, "Making chroma features more robust to timbre changes," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Taipei, Taiwan, Apr. 2009, pp. 1869–1872.

[14] Meinard Müller, Frank Kurth, and Michael Clausen, "Audio matching via chroma-based statistical features," in *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR)*, 2005, pp. 288–295.

[15] J. McQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 1967, pp. 281–297.

[16] S. P. Lloyd, "Least Square Quantization in PCM," *IEEE Transactions on Information Theory*, vol. IT-28, no. 2, March 1982.

[17] X. Huang, A. Acero, and H. Hon, *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*, Prentice Hall PTR, 2001.

[18] B. Raj, "MLSP Lecture Notes, Class 14, Slide 113, CMU," Fall 2010.