# A TEXT RETRIEVAL APPROACH TO CONTENT-BASED AUDIO RETRIEVAL

**Matthew Riley**
University of Texas at Austin
mriley@gmail.com

**Eric Heinen**
University of Texas at Austin
eheinen@mail.utexas.edu

**Joydeep Ghosh**
University of Texas at Austin
ghosh@ece.utexas.edu

## ABSTRACT

This paper presents a novel approach to robust, content-based retrieval of digital music. We formulate the hashing and retrieval problems analogously to that of text retrieval and leverage established results for this unique application. Accordingly, songs are represented as a "Bag-of-Audio-Words" and similarity calculations follow directly from the well-known Vector Space model [12]. We evaluate our system on a 4000 song data set to demonstrate its practical applicability, and evaluation shows our technique to be robust to a variety of signal distortions. Most interestingly, the system is capable of matching studio recordings to live recordings of the same song with high accuracy.

## 1 INTRODUCTION

Large digital music libraries are becoming commonplace on consumer computer systems, and with their growth our ability to automatically analyze and interpret their content has become increasingly important. The ability to find acoustically similar, or even duplicate, songs within a large audio database is a particularly important task with numerous potential applications. For example, an automated system similar to MusicBrainz [11] might organize a user's music collection by properly naming each file according to artist and song title. Another application could attempt to retrieve the artist and title of a song given a short clip recorded from a radio broadcast or perhaps even hummed into a microphone.

Due to the rich feature set of digital audio, a central task in this process is that of extracting a representative audio fingerprint that describes the acoustic content of each song. We hope to extract from each song a feature vector that is both highly discriminative between different songs and robust to common distortions that may be present in different copies of the same source song. With the multitude of compression formats and signal extraction processes, two copies of the same song can sound perceptually identical while having very different digital representations. Additionally, it is desirable for the audio fingerprint to compress the existing audio information into a much smaller representation, thus enabling efficient retrieval and requiring less storage than that of the initial data set.

In this paper, we present a novel hashing methodology that satisfies these constraints. We show that a technique based on methods for text retrieval performs well for the desired applications, and benefits from established research results in the area. Section 2 reviews existing work related to our application and section 3 details our application of the text retrieval techniques to content-based audio retrieval. Section 4 details our experimental evaluation of the proposed algorithm. Section 5 discusses practical implementation considerations and section 6 concludes with final remarks and suggestions for future work.

## 2 RELATED WORK

The problem of audio fingerprinting has been studied widely. In 2002, Haitsma and Kalker proposed a method for extracting audio fingerprints that they showed were robust to a variety of signal distortions. In addition, they outlined a database searching algorithm for locating a fingerprint most similar to a given target fingerprint [7]. One of the drawbacks of their system is the amount of memory required to store an audio fingerprint (approx. 100 KBytes for a 5 minute song). In addition, it was unclear whether or not their fingerprints could feasibly be used to match a studio recording to a live performance or a cover version (i.e. a performance of the original composition by another artist, possibly rearranged).

Existing work on cover song detection was presented for a competition at The Music Information Retrieval Evaluation eXchange (MIREX). In 2006, Dan Ellis' team from Columbia University won the competition by posting accuracy of about 60% using a method that computed similarities between songs by cross-correlating sequences of their so-called Chroma features [5]. Their similarity measure is equally applicable to the problem of matching a studio recording to a live performance. However, the high computational complexity of cross-correlating Chroma feature vector sequences does not make sense in an audio retrieval context.

We have not found previous research that directly applies text retrieval methods to the task of audio retrieval, but a similar approach has been taken for Object Recognition in images [13]. Further, Casey and Slaney [3] present a system


**Figure 1**. Block diagram for "Bag-of-Audio-Words" Representation

for discovering derivative audio works and describe a locality sensitive hashing procedure for matching similar songs that is derived from the "text shingles" method originally proposed in [2]. Finally, A vector quantization scheme similar to ours but using self-organizing maps is described in [14].

## 3 BAG-OF-AUDIO-WORDS REPRESENTATION

Our basic framework for extracting a song's Bag-of- Audio-Words representation is depicted in Figure 1. First, the song is converted from its original digital audio format into a 22.05 kHz, 16-bit, mono wav file. Next, the signal is divided into non-overlapping time segments and audio features are extracted from each segment. Then, a vector quantization (VQ) technique is used to map audio feature vectors to cluster numbers in the set $\{1, 2, \ldots, k\}$, where each cluster corresponds to what we refer to as an audio-word. Finally, each song is represented by a histogram of audio-word occurrences.

### 3.1 Audio Segmentation

In the segmentation process we extract non-overlapping 200 millisecond clips. We originally explored aligning audio segments to detected beats, which were extracted using the beat tracking algorithm proposed by Ellis and Poliner [5], but experimentally determined there to be little difference in system performance between the segmentation approaches. Additionally, uniform segmentation has the advantage of requiring less computational complexity.

### 3.2 Audio Feature Extraction

Several papers have characterized the suitability of numerous audio features for a variety of scenarios [5, 6, 10]. For our application, we chose the so-called normalized Chroma feature.

The Chroma feature is a 12-dimensional, real-valued vector that approximates an audio signal's strength at each musical note (e.g. A, A#, B, etc.), regardless of octave. Normalization of a Chroma is then performed by dividing by its vector norm.

We chose normalized Chroma features because it is very important to the performance of our audio-word histogram representations that the feature vectors for an audio segment and a distorted version are very similar. First, Chroma features are invariant to types of distortions that affect timbre because they only attempt to capture tonal information. Second, Chroma features are useful in detecting live/cover songs because they disregard information about octave, and are therefore somewhat invariant to certain differences between renditions or arrangements. Finally, the normalization of these Chroma features reduces the effects of a particular recording's loudness.

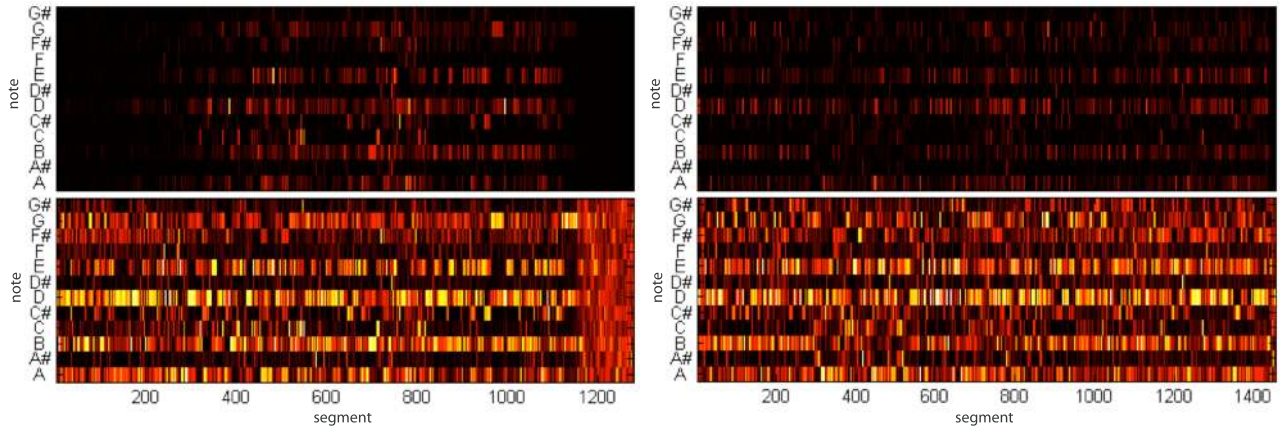
A song's chromagram is the sequence of the audio segment's Chroma features. Example chromagrams, with and without normalization, for the original and live performance recordings of the same song are depicted in Figure 2. We can see here the importance of normalization as the fading in volume from the original recording does not appear in the live performance. The normalization of Chroma vectors helps to eliminate the differences between the chromagrams of the two renditions.

### 3.3 Vector Quantization and Song-Level Histograms

Vector quantization primarily consists of performing clustering in the 12-dimensional Chroma space. The clustering process identifies $k$ dense regions within a set of Chroma features extracted from our data set, which we collectively refer to as audio-words. Thereafter, when a Chroma feature is extracted from a song segment we calculate the nearest audio-word and consider the segment to be an occurrence of that audio-word. This quantization procedure forms the basis of matching distorted signals – song segments that sound very similar will have slightly different Chroma feature vectors but are expected to be assigned to the same audio-word.

For clustering we first collect a large sample of Chroma vectors from a variety of songs that are separate from our test set (approx. 100,000 vectors). We use K-Means to compute the $k$ cluster centers, or audio-words.

The vector quantization procedure takes as input a song's sequence of Chroma vectors, and for each outputs one or more numbers in $\{1, 2, \ldots, k\}$ corresponding to the closest audio-word(s), as measured by Euclidean distance. Finally, the song $x$ is mapped to a $k$-dimensional vector encoding the frequency of occurrence of each audio-word in the song: $\phi(x) = [f_1, f_2, \ldots, f_k]$ where $f_i$ denotes the number of occurrences of the $i$-th audio-word in song $x$.




(a) Original Recording

(b) Live Recording

**Figure 2**. Chromagrams (with and without normalization) of the original and live performance of the same song.

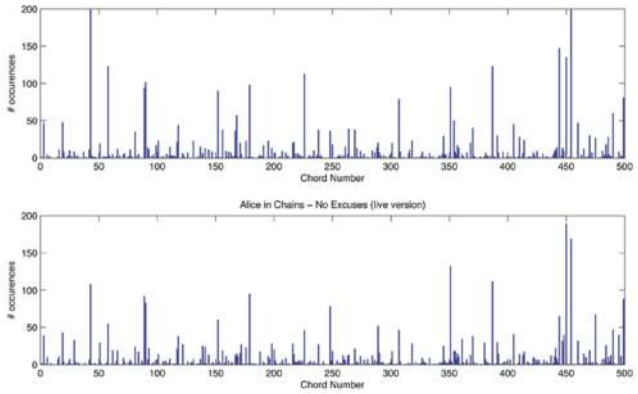



**Figure 3**. Audio-word histograms for the original and live recordings of Alice in Chains' "No Excuses"

By representing individual Chroma vectors by the cluster center to which they belong, we make equivalent in the histogram representation any segments that were musically very similar but may have had slightly different Chroma vectors. In terms of text retrieval, this is analogous to the stemming procedure often performed on the words of a text document [9].

### 3.4 Audio-Word Histogram Term Weighting

After forming the audio-word histograms we weight each of the $k$ terms according to the term-frequency inverse document frequency (TF-IDF) scheme. We determine the term weightings $t_i, i \in \{1, 2, \ldots, k\}$, according to the following equation [1]:

$$t_i = \frac{n_{id}}{n_d} log \frac{N}{n_i} \tag{1}$$

Here $n_{id}$ is the number of times the feature $i$ occurs in song $d$, $n_d$ is the total number of features in song $d$, $N$ is the total number of songs in the database, and $n_i$ is the number of songs in which feature $i$ is present. The log term is larger for features that occur in few songs (rare features) and the leading term is large for features that occur many times in a given song. The $n_d$ term serves to normalize the weights so that songs with many features can match songs with fewer features.

### 3.5 Calculating Song Similarity

Three different measures were considered and tested for computing the similarity between chord-histograms. We considered Cosine Similarity, Chi-Squared Similarity, and Euclidean distance (with normalization), given in equations (2), (3), and (4) respectively. In each of these equations, $A$ and $B$ represent the $k$-dimensional histogram vectors $\phi(a)$ and $\phi(b)$ of songs $a$ and $b$.

$$sim_{ab} = \cos^{-1}\left(\frac{A \cdot B}{||A||\,||B||}\right) \tag{2}$$

$$sim_{ab} = \frac{1}{2}\sum_{i=1}^{k}\frac{(A_i - B_i)^2}{A_i + B_i} \tag{3}$$

$$sim_{ab} = \sqrt{\sum_{i=1}^{k}\left(\frac{A_i}{|A|} - \frac{B_i}{|B|}\right)^2} \tag{4}$$

For each of these similarity measures, a smaller value indicates a better song match. The cosine similarity measure helps when calculating the similarity between two songs without TF-IDF weightings because the dot product is normalized by the vector magnitudes, thus allowing songs of

different lengths to be similar. Analogously to a typical internet search engine query, given a query song or song clip, we use this similarity measure to return a ranked list of similar songs from the database. For our application we use the original WAV of the studio recording of a song as the query and expect the top results returned to be the distorted versions of the same song.

## 4 EXPERIMENTAL EVALUATION

We evaluated our system on a data set of 4000 songs drawn randomly from a variety of musical genres. A 4000 song data set under mp3 compression is roughly 40 gigabytes in size, which, if we use the size of current portable mp3 players as a guide, is a generous estimation of the size of a random user's music collection. In addition to the base 4000 tracks, we selected 60 additional tracks as query songs and quantify the system's ability to correctly retrieve distorted versions of the tracks from within the 4060 total song set. The idea here is to determine the discriminative power of the song-level feature vectors in a practical setting.

### 4.1 Vector Quantization

We experimented with several types of clustering algorithms besides K-Means, including hierarchical agglomerative clustering (HAC) and Gaussian mixture modeling (GMM) with expectation maximization. However, most of these other clustering algorithms involved greater computational complexity without improving the VQ and resulting song matching. We determined experimentally that $k = 500$ works well. Figure 4 depicts the Chroma vectors belonging to four different, and shows how some of the clusters resemble musical chords with two to three significant notes (lighter color indicates greater signal strength).

In addition, we determined experimentally that our matching performance could be improved by modifying the way in which we built our audio-word histograms. Instead of assigning a Chroma vector to the single closest cluster center, we assigned each one to the three closest centroids. We also tried some other soft assignment techniques. For example, instead of simply adding 1 to a Chroma's assigned histogram bins, we tried adding values weighted by how close the Chroma was to the cluster center. However, this approach actually hurt our results, especially in the case of identifying live/cover songs.

### 4.2 Robustness to Signal Distortion

The main experiment we performed in order to evaluate our Bag-of-Audio-Words song representation is as follows. First, we computed audio-word histograms for each of our 4060 songs. We then applied a signal distortion (overdrive, echo, etc.) using Adobe Audition to each of the 60 query songs,

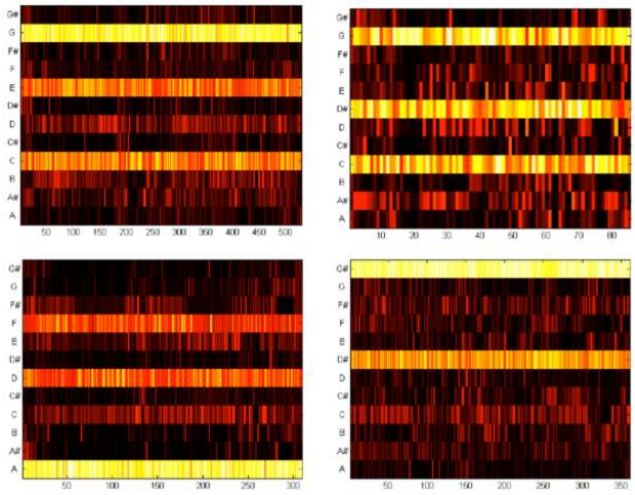


**Figure 4**. Four example clusters showing chord structure. (C major, C minor, D minor, D# power chord)

| | Similarity Measure | | |
|---|---|---|---|
| **Distortion Type** | Chi-Sq | Cosine | Euclidean |
| Overdrive | 100% | 100% | 100% |
| Echo | 98.3% | 98.3% | 98.3% |
| Reverb | 100% | 100% | 100% |
| Speedup (1%) | 98.3% | 98.3% | 98.3% |
| mp3 (32 Kbps) | 100% | 100% | 100% |
| Live/Cover | 67.8% | 50.8% | 50.8% |

**Table 1**. Retrieval results

and computed the audio-word histograms for those distorted versions. Next, we computed our three similarity measures between each of the 4120 audio-word histograms. Finally, we calculated, for each similarity measure, the percentage at which the distorted query songs were most similar to the original query songs. Our results are outlined in Table 1.

### 4.3 Live/Cover Song Identification

In addition to facilitating matching between original and distorted recordings, our audio-word histogram representation was found to be useful in matching original studio recordings with live performances and cover songs to a lesser degree. To test this type of matching, we performed the same procedure as described in the previous section, but used a different set of 59 query songs for which we had live/cover versions (only a quarter of these were actually covers).

The results of this test are given in Table 1. It is somewhat surprising how good our live/cover song detection results were when compared to low-bitrate mp3 compression. In addition, it is interesting to note that our audio-word histograms were used to correctly match several cover songs.

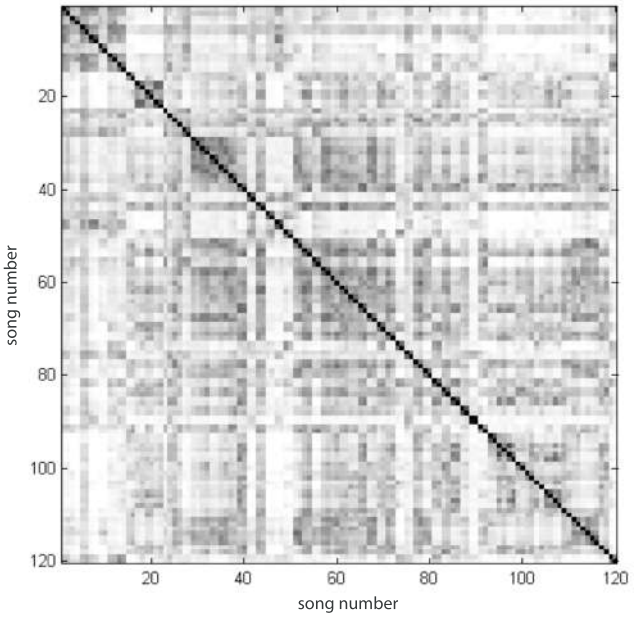



**Figure 5**. Similarity matrix for live performance retrieval.

For example, we correctly matched Stevie Ray Vaughn's cover version of "Little Wing" (6:49) to the original by Jimi Hendrix (2:24). The cover version is significantly longer and includes a much longer guitar solo.

Figure 5 shows a cosine similarity matrix for the query songs and their live/cover versions. In this matrix, a song at an odd index represents an original query song, and the song at the following even index is the live/cover version. If you look closely at this figure, you can see dark 2x2 squares along the diagonal, indicating that the original and live/cover versions have high similarity measures.

In addition to testing live/cover song retrieval from within a 4060 total song-set, we performed some other tests in the framework used by Ellis and Poliner [5]. Their procedure is as follows. First, they collected a set of approximately 80 original songs, and a separate set of cover versions of those 80. Then, their goal was to match a given song in the set of originals to its corresponding cover in the other set. For their 80 songs, they had a matching accuracy of around 60%. When we tested our matching algorithm on these same songs and in their framework, we only achieved an accuracy of 37.5% (using chi-squared similarity).

We then repeated the same procedure using our 59 songs, and the performance achieved by Ellis and Poliner's matching scheme was 70%, whereas our algorithm gave 90%. Our explanation for these surprising results is the following. Suppose we consider a cover song whose arrangement is such that every E minor chord (E, G, B) from the original version is replaced by an E power chord (E, B). In the case where song matching is based on chromagram cross-correlations, the matching between original and cover will be affected but not significantly. Under our scheme, these two chords would result in Chroma vectors that would map to different audio-word bins. Thus, our audio-word histogram representation can effectively be used to identify live song performances, but perform poorly on cover songs whose arrangements are significantly different from the original.

## 5 IMPLEMENTATION CONSIDERATIONS

Scalability was a central concern in the design of our proposed system. Here we discuss two implementation considerations – the Inverted Index and Locality Sensitive Hashing – that extend naturally to our application, and indicate that our algorithm will scale well to very large data sets.

### 5.1 Query Processing with an Inverted Index

The Inverted Index data structure is critical to the rapid processing speed of many text retrieval systems. This index contains all words in the text corpus and with each stores a list of every document in which that word is present. When performing a retrieval operation, the system looks at the inverted index to quickly retrieve a list of documents containing one or more of the words present in the query. When using the cosine similarity metric, only the words present in the query will affect the similarity measure, so documents not returned by a lookup in the inverted index can be safely ignored. In practice, this usually results a dramatic performance increase because the computationally expensive similarity metric must only be computed on a small subset of the entire database. Analogously, an Inverted Index for our application would contain each audio word and with each store a list of every song in which that word is present. This would be especially useful for an application in which only a short clip of a song is used to query the system.

### 5.2 Fast Search with Locality Senstive Hashing

The task of nearest-neighbor calculation in high-dimensional data can be efficiently implemented using Locality Sensitive Hashing (LSH). LSH is a hashing technique in which the probability that two objects are hashed to the same bin is proportional their similarity according to some metric. Hence, songs with very similar song-level histograms will likely be hashed to the same bin, allowing sub-linear determination of the nearest-neighbor of a given song within the data set, and therefore very rapid retrieval. LSH techniques exist for a number of similarity measures, including cosine similarity [4] and euclidean distance [8].

## 6 CONCLUSIONS

We have shown that a Bag-of-Audio-Words approach to audio retrieval can be both discriminative in a large data set

and robust to common signal distortions. We have also discussed numerous considerations for the practical application of our approach, addressing issues of scalability and efficiency. Excellent retrieval accuracy for a wide variety of distortions indicates that our approach will be useful for numerous applications. A natural extension of this work would be to add temporal information into the song-level feature vectors. Presently, the Bag-of-Audio-Words approach ignores all time-series information present in the initial song. Perhaps augmenting the song-level vectors to be a pyramid of audio word histograms, formed at different resolutions of song division, would lead to even better performance results.

## 7 REFERENCES

[1] Baeza-Yates, R. and Ribeiro-Neto, B. *Modern Information Retrieval*. Addison-Wesley Longman Publishing, USA, 1999.

[2] Broder, A.Z., Glassman, S. C., Manasse, M. S. and Zweig G. "Syntactic clustering of the Web", *Proceedings of the 6th International World Wide Web Conference*, pp. 391-404, 1997.

[3] Casey, M. and Slaney, M. "Song Intersection by Approximate Nearest Neighbour Retrieval", *Proceedings of the 7th International Conference on Music Information Retrieval*, 2006.

[4] Charikar, M. "Similarity Estimation Techniques from Rounding Algorithms", *ACM Symposium on Theory of Computing*, 2002.

[5] Ellis, D. and Poliner, G. "Identifying Cover Songs with Chroma Features and Dynamic Programming Beat Tracking", *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, 2007.

[6] Flexer, A., Gouyon, F., Dixon, S. and Widmer, G. "Probabilistic Combination of Features for Music Classification", *Proceedings of the 7th International Conference on Music Information Retrieval*, 2006.

[7] Haitsma, J. and Kalker, T. "A Highly Robust Audio Fingerprinting System", *Proceedings of the 3rd International Conference on Music Information Retrieval*, 2002.

[8] Indyk, P. and Motwani, R. "Approximate Nearest Neighbors: towards removing the curse of dimensionality", *Proceedings of the 30th Symposium on Theory of Computing*, pp. 604-613, 1998.

[9] Lovins, J. "Development of a stemming algorithm", *Mechanical Translation and Computational Linguistics*, vol. 11, pp. 22-31, 2006.

[10] Mandel, M. and Ellis, D. "Song-Level Features and Support Vector Machines for Music Classification", *Proceedings of the 6th International Conference on Music Information Retrieval*, 2005.

[11] MusicBrainz, `http://www.musicbrainz.org`

[12] Salton, G., Wong, A. and Yang, C. S. "A Vector Space Model of Automatic Indexing", *Commun. ACM*, Plymouth, USA, 2000.

[13] Sivic, J. and Zisserman, A. "Video Google: A Text Retrieval Approach to Object Matching in Videos", *Proceedings of the International Conference on Computer Vision*, 2003.

[14] Vignoli, F. and Pauws, S. "A Music Retrieval System Based on User Driven Similarity and Its Evaluation", *Proceedings of the 6th International Conference on Music Information Retrieval*, 2005.