2009-07-02

Musical Query-by-Content Using Self-Organizing Maps

Kyle B. Dickerson
Brigham Young University - Provo

MUSICAL QUERY-BY-CONTENT USING SELF-ORGANIZING MAPS

by

Kyle Britton Dickerson

A thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computer Science

Brigham Young University

August 2009

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Kyle Britton Dickerson

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date	Dan Ventura, Chair
Date	Michael A. Goodrich
Date	Robert P. Burton

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the thesis of Kyle Britton Dickerson in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Date	Dan Ventura Chair, Graduate Committee

Accepted for the Department

Date	Kent E. Seamons Graduate Coordinator

Accepted for the College

Date	Thomas W. Sederberg Associate Dean, College of Physical and Mathematical Sciences

ABSTRACT

MUSICAL QUERY-BY-CONTENT USING SELF-ORGANIZING MAPS

Kyle Britton Dickerson

Department of Computer Science

Master of Science

The ever-increasing density of computer storage devices has allowed the average user to store enormous quantities of multimedia content, and a large amount of this content is usually music. Current search techniques for musical content rely on meta-data tags which describe artist, album, year, genre, etc. Query-by-content systems, however, allow users to search based upon the actual acoustical content of the songs. Recent systems have mainly depended upon textual representations of the queries and targets in order to apply common string-matching algorithms and are often confined to a single query style (e.g., humming). These methods also lose much of the information content of the song which limits the ways in which a user may search. We present a query-by-content system which supports querying in several styles using a Self-Organizing Map as its basis. The results from testing our system show that it performs better than random orderings and is, therefore, a viable option for musical query-by-content.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The ability to purchase music in digital formats has caused a dramatic increase in the music collections of even casual computer users. Many personal libraries contain thousands of songs which the user needs to search through when looking for a particular song. Current search techniques typically rely on meta-data tags which describe artist, album, year, genre, or similar information. These tags must be created by a human and attached to each file—an error-prone process which is, at best, inconvenient.

Much work has been done to create systems which try to automatically tag a song with genre information [28]. Having accurate and automatically generated meta-data is helpful, but only if the user can remember the information stored in the tags. If, however, the user can only remember the tune of the song it is necessary to search by content rather than by meta-data. These types of systems, which rely on information retrieved from audio files, are generally referred to as Music Information Retrieval (MIR) systems. Unfortunately, no system yet exists that searches audio by content and which is accurate, fast, robust, and intuitive.

Any MIR system will require a method for determining the similarity of songs. In fact, the system is heavily dependent on this distance function. Many current systems first transcribe the audio content to a text representation and then use common string-matching techniques as the distance function [3, 9, 13, 23, 25]. This process, however, is difficult to do accurately and reduces the content-rich music to a simple text string.

Instead, one could in principle extract various acoustic features from the audio using signal processing techniques with the distance function dependent upon which musical features are used. Determining which features to extract is a difficult problem. Whether or not a set of features is useful depends upon the context in which they will be used [1, 2, 11, 20, 21, 24, 29].

Once a good feature set is found, it is still necessary to determine a suitable distance function. The choice of distance function has also been heavily studied, resulting in varying levels of success [7, 8, 16, 22, 30]. Rather than attempt to design a specific distance function, we will use a Self-Organizing Map (SOM) to reduce the dimensionality of the feature space allowing us to use simpler distance metrics.

The rest of our paper is organized as follows: in Chapter 2 we describe how SOMs work and why they will be useful in MIR systems. We then present the current state of Query-by-Content systems. In Chapter 3 we present a proof-of-concept system which uses a SOM to create a musical similarity search system and describes our Query-by-Content system using SOMs. Note that Chapter 3 will appear as "Music Recommendation and Query-by-Content Using Self-Organizing Maps" in the *Proceedings of the International Joint Conference on Neural Networks,* 2009 [4]. Our system's efficacy is evaluated in Chapter 4. In Chapter 5 we summarize our findings and discuss future work. Appendices A through D contain the detailed results of our experiments.

Chapter 2

Related Work

Our work combines two well-established concepts—Self-Organizing Maps and Musical Query-by-Content Systems. SOMs are interesting in that they produce a two-dimensional space which is smoothly interpolated across training values. Musical Query-by-Content Systems allow a user to search for specific songs by simply mimicking the song they want, either by whistling, humming, singing, or by some other query style.

2.1 Self-Organizing Maps

The Self-Organizing Map is an unsupervised learning algorithm which uses data from a feature set of any dimensionality to generate a two-dimensional grid (map) while attempting to preserve as much of the intrinsic similarity of the data as possible [12]. The algorithm begins by initializing a map of random feature vectors of the length of the data feature vectors. This grid may be considered to wrap around both horizontally and vertically, creating a toroid, to prevent unusual edge effects. For each training datum, the closest matching grid location is found, and a neighborhood around the matching location is updated to become more like that datum. Over time the size of the neighborhood shrinks and the influence of the update decreases (see Figure 2.1). In effect, these neighborhood alterations create smooth interpolations

```
$V \leftarrow m^2$ real-valued vectors of length $n$
Arrange $V$ onto an $m \times m$ grid
Choose $\alpha, j, k \ \epsilon \ (0, 1)$
Choose $\rho \ \epsilon \ (1, m)$
**while** NOT DONE **do**
  **for** training datum $\bar{x} \epsilon X$ **do**
    $\bar{v} \leftarrow \text{argmin}_{\bar{v} \epsilon V} \text{ vector\_distance} (\bar{v}, \bar{x})$
    $v_i \leftarrow \alpha x_i + (1 - \alpha) v_i$
    **for** $\bar{u}$ in neighborhood$(\bar{v}, \rho)$ **do**
      $\alpha_u \leftarrow \alpha \times \frac{\rho - \text{grid\_distance}(\bar{v}, \bar{u})}{\rho}$
      $u_i \leftarrow \alpha_u x_i + (1 - \alpha_u) u_i$
    **end for**
  **end for**
  $\alpha \leftarrow k\alpha$
  $\rho \leftarrow j\rho$
**end while**
```

Figure 2.1: *SOM algorithm pseudocode.* X is the set of all training data. ρ is the radius used in determining neighborhoods. α is the weight given to the training datum when updating the vectors on the grid. The vector and grid distance functions can be any metric—we have used Euclidean distance for both. Typical parameter values: $\alpha = 0.1$, $\rho = \frac{m}{2}$, j and k are linear decay functions.

between data points across the map, a desirable property which allows us to train on a subset of the available data and still get a useful map.

SOMs have already been used successfully in MIR systems. One of the first such systems was presented by Feiten and Günzel [6]. Harford [10] uses a SOM to perform melody retrieval. Dittenbach, Merkl, and Rauber [5] introduce a growing hierarchical SOM which Rauber, Pampalk, and Merkl [26] use to create a musical archive based upon sound similarity. As far as we have been able to determine, no work has been done in applying SOMs to a musical query-by-content system. However, an image query-by-content system was created by Laaksonen, Koskela and Oja [14]. In this system the user does not input a free-form query but rather selects images from presented sets as the system locates the area of the SOM the user is interested in.

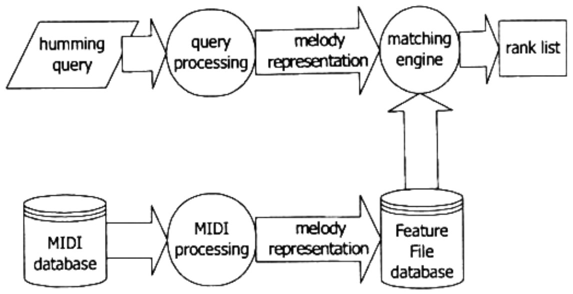


Figure 2.2: *Typical organization of current musical query-by-content systems [18].* Queries and targets are first converted to text representations, and then compared using common string-matching techniques.

Our work uses a set of SOMs to power a musical query-by-content system. We present the current state-of-the-art in musical query-by-content systems in the next section before we discuss our work.

2.2 Query-by-Content Systems

The musical query-by-content field has almost exclusively been comprised of systems relying on humming-only queries which then use a form of melody extraction to represent the query textually before applying common string-matching algorithms to find a match to a stored set of text representations of the target songs (Figure 2.2). The first such system uses only pitch change to represent the queries and songs [9]. Another system, presented by Kosugi et al. [13], uses beats instead of notes and relies on a MIDI format for stored songs. A number of similar string-matching based systems have been presented by Pauws [23], Raju, Sundaram, and Rao [25], and Birmingham, Dannenberg, and Pardo [3].

The textual representation of the query is a considerable constraint in all of the above systems. Music is a very rich medium—songs often contain concurrent parts and a user may remember any single part while forgetting the others. Having a strong, unique bass with a vocal track accompanied by instrumentals is not uncommon, but by reducing the song to a single text representation much of that information is lost. Perhaps a user can remember how the bass sounded, but not the vocal or instrumentals. Current systems would be unable to help them find the correct song. One system was presented which allows more comprehensive searching by extracting features directly from the MP3 encoding format which are segmented into a set of “phases” to which queries are matched [15]. This work, however, is limited to only songs in MP3 format.

We use a SOM to power a musical query-by-content system, thereby allowing us to retain as much of the original audio content as possible. By retaining more content in the target songs, we hope to allow users to search using a broader range of query types—for example humming, whistling, or singing.

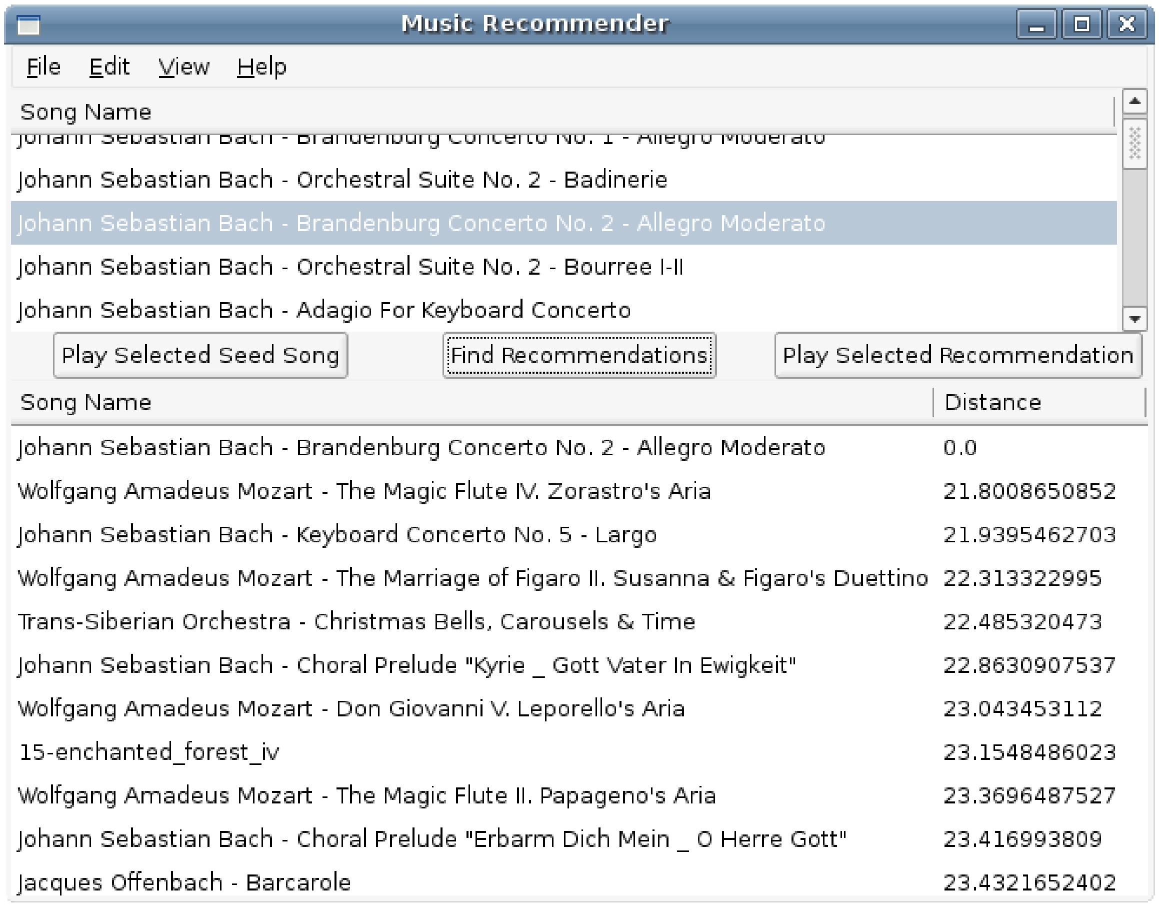
Chapter 3

System Design

In order to provide evidence that our project would work as anticipated we first created a proof-of-concept system which uses a SOM to power a music recommendation system. As mentioned above, this type of work has been done previously; by recreating this work we verified that it provides useful results. It also served as a starting point for our query-by-content system.

3.1 Music Recommendation

We have created a proof-of-concept, SOM-based music recommendation system similar to those mentioned above. A screenshot of our music recommendation system is shown in Figure 3.1. Our system generates a 128x128 map using a randomly chosen 25-second segment from 20% of the 881 songs available in our personal music library (Figure 3.2). Before creating the SOM the audio is preprocessed to extract feature vectors. The selection of features is an important aspect of any machine learning algorithm, and when using audio signals, the task is even more difficult because we must also decide what size windows to extract the features from. For simplicity, rather than attempt to determine a most effective set of features and parameters, we use features and parameters that are common in many MIR systems [17, 19]. The features we use include the power spectrum, strongest beat, beat sum, Mel-Frequency Cepstral Coefficients, and Linear Predictive Coding. The features were extracted for



Song Name
Johann Sebastian Bach - Brandenburg Concerto No. 1 - Allegro Moderato
Johann Sebastian Bach - Orchestral Suite No. 2 - Badinerie
Johann Sebastian Bach - Brandenburg Concerto No. 2 - Allegro Moderato
Johann Sebastian Bach - Orchestral Suite No. 2 - Bourree I-II
Johann Sebastian Bach - Adagio For Keyboard Concerto

Song Name	Distance
Johann Sebastian Bach - Brandenburg Concerto No. 2 - Allegro Moderato	0.0
Wolfgang Amadeus Mozart - The Magic Flute IV. Zorastro's Aria	21.8008650852
Johann Sebastian Bach - Keyboard Concerto No. 5 - Largo	21.9395462703
Wolfgang Amadeus Mozart - The Marriage of Figaro II. Susanna & Figaro's Duettino	22.313322995
Trans-Siberian Orchestra - Christmas Bells, Carousels & Time	22.485320473
Johann Sebastian Bach - Choral Prelude "Kyrie _ Gott Vater In Ewigkeit"	22.8630907537
Wolfgang Amadeus Mozart - Don Giovanni V. Leporello's Aria	23.043453112
15-enchanted_forest_iv	23.1548486023
Wolfgang Amadeus Mozart - The Magic Flute II. Papageno's Aria	23.3696487527
Johann Sebastian Bach - Choral Prelude "Erbarm Dich Mein _ O Herre Gott"	23.416993809
Jacques Offenbach - Barcarole	23.4321652402

Figure 3.1: *Music recommender screenshot.* After path descriptors for each of the songs have been generated, a user may request to see the top ten recommendations for any song in the system.

every 5-second window, with a 50% overlap of segments. This preprocessing work therefore excludes any system we create from being a real-time system without many hours of prior computation. Our goal, however, is to show that a query-by-content system can be built using SOMs and not, necessarily, that it will be fast enough (yet) to use in real-time.

Once the SOM is trained, each of the 881 songs is completely mapped into it from start to end (one location for each 5 seconds of audio), creating a path descriptor within the map for each song (see Figure 3.3). This path is unique in our use of SOMs. As usual for SOMs, each data instance will map to a single location within the SOM; but, because each clip of audio is represented by an ordered set of feature vectors, we

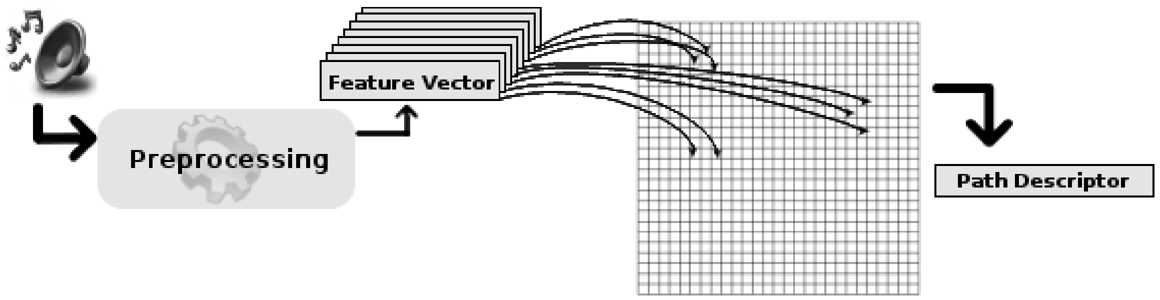


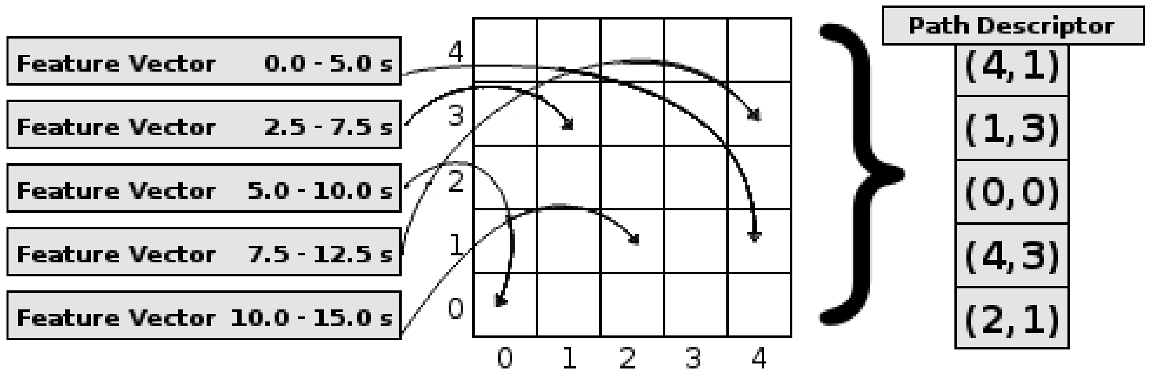
Figure 3.2: *System design for music recommendation.* During training, 9 consecutive feature vectors (representing 25 seconds of audio) are taken from 176 (20%) of the preprocessed songs (1584 unique feature vectors). The feature vectors are used as the training data for the SOM algorithm. After training has completed, each song's complete set of feature vectors is mapped into the SOM creating unique path descriptors.

can represent the clip as the ordered set of locations the individual vectors map to. This allows us to cleanly handle the temporal nature of the problem without needing to change the actual SOM algorithm in any way. These paths now act as unique descriptors for each song and are stored for later reference.

The distance between two songs is calculated as the average Euclidean distance between the points composing their unique path descriptors. When the descriptors vary in length, the extra points in the longer descriptor are simply ignored (see Figure 3.4). More advanced techniques could be used; however, subjectively, our proof-of-concept results are fairly good even using this simple algorithm. We assume that a more thorough algorithm that attempts segmentation and alignment matching would be more effective.

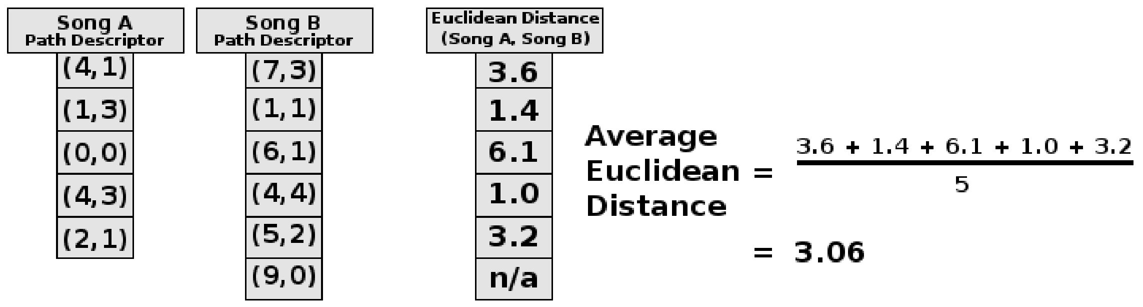
3.1.1 Music Recommendation Evaluation

To test the recommendation system, we select various songs and subjectively decide if the top five recommendations seem similar. The system performs well with some types of songs, such as classical and rock, while doing poorly with others, such as dance club type music. For example, when we choose *Bach - Brandenburg Concerto*



Feature Vector 0.0 - 5.0 s
Feature Vector 2.5 - 7.5 s
Feature Vector 5.0 - 10.0 s
Feature Vector 7.5 - 12.5 s
Feature Vector 10.0 - 15.0 s
4
3
2
1
0
0 1 2 3 4
Path Descriptor
(4,1)
(1,3)
(0,0)
(4,3)
(2,1)

Figure 3.3: *Path descriptor example.* After the SOM has been trained, the feature vectors representing a song can be mapped into the SOM. Each feature vector maps to a single location and the ordered set of locations produces a unique path descriptor describing that song. These path descriptors are then used to compute the distance between songs (see Figure 3.4).



$$\text{Average Euclidean Distance} = \frac{3.6 + 1.4 + 6.1 + 1.0 + 3.2}{5} = 3.06$$

Figure 3.4: *Song distance calculation example.* The distance between two songs is calculated as the average Euclidean distance between the points composing their unique path descriptors. If the path descriptors have different lengths, the extra points in the longer descriptor are ignored. This very simple technique yields fairly good results.

1. *Haydn - Divertimento No. 1 in B flat major*
2. *Mozart - The Magic Flute: Aria of the Queen of Night*
3. *Bach - Keyboard Concerto No. 4 - Larghetto*
4. *Pachelbel - Canon in D major*
5. *Mozart - The Magic Flute: Zorastro's Aria*

Figure 3.5: *Top 5 recommendations for Bach - Brandenburg Concerto No. 2, Allegro Moderato.* All songs are orchestral pieces representing good matches to the seed song.

1. *3 Doors Down - Be Like That*
2. *Evanescence - Bring Me To Life*
3. *Linkin Park - Pushing Me Away*
4. *Matchbox Twenty - All I Need*
5. *Lifehouse - First Time*

Figure 3.6: *Top 5 recommendations for Dashboard Confessional - Hands Down.* Recommended songs reflect the prominent vocals and the strong rock background of the seed.

No. 2, Allegro Moderato we receive the results shown in Figure 3.5. Each of these songs has a common classical, orchestral sound, so, in our opinion, the recommender system does a fairly good job. The pieces from *The Magic Flute*, however, contain vocals while the others do not.

When we use a more contemporary piece as the seed song the system still performs well. Figure 3.6 shows the recommendations for the song *Dashboard Confessional - Hands Down.* These songs feature vocals over a strong rock sound. *Be Like That*, despite having the best rank, is probably the least like the others, having a slower tempo and softer feel.

A real test of our recommendation system is if two different versions of the same song appear similar to each other. We have recordings of the song *You Raise Me Up* as performed by the group Celtic Woman as well as by the solo artist Josh

1. *Harry Potter - Double Trouble*
2. *Phantom of the Opera - Angel of Music*
3. *Trans-Siberian Orchestra - God Rest Ye Merry Gentlemen*
4. *Mannheim Steamroller - Enchanted Forest IV*
5. *Mozart - The Marriage of Figaro: Duettino*

Figure 3.7: *Top 5 recommendations for Celtic Woman - You Raise Me Up.* These songs contain soft music mainly featuring vocals over light instrumentals. Compare with Figure 3.8.

1. *Harry Potter - Double Trouble*
2. *Phantom of the Opera - Angel of Music*
3. *Trans-Siberian Orchestra - God Rest Ye Merry Gentlemen*
4. *Mannheim Steamroller - Enchanted Forest IV*
5. *Mannheim Steamroller - Enchanted Forest III*

Figure 3.8: *Top 5 recommendations for Josh Groban - You Raise Me Up.* Compare with Figure 3.7.

Groban. The recommendations for these songs are a little bit more unusual and may not be considered very helpful as recommendations. They are, however, consistent between the two pieces. So while the recommendations may be less intuitive, they are at least not arbitrary and do contain strong similarities to the seed songs. The top five recommendations for the Celtic Woman and Josh Groban versions are presented in Figures 3.7 and 3.8, respectively.

The top three songs in each list all feature strong vocals over light instrumentals, which is consistent with the seed songs. The songs from Mannheim Steamroller, however, are not really what we would call music. They come from one of the artist's Halloween CDs and are simply spooky sounds to be played as sound effects. It is interesting that neither list contains the other song, yet they contain the same set of similar songs. It is not until the sixtieth song in the Josh Groban list that the Celtic

1. *Bizet - Carmen Suite No. 1*
2. *Harry Potter - Double Trouble*
3. *Phantom of the Opera - Angel of Music*
4. *Creedence Clearwater Revival - Susie Q*
5. *Schubert - Symphony No. 5 in B flat major*

Figure 3.9: *Top 5 recommendations for Sean Paul - We Be Burnin'.* These are unusual recommendations which are unlike the seed song. This is probably a result of the training set not containing songs representative of the seed song.

Woman version appears. The Josh Groban version does not appear within the first 100 results of the Celtic Woman list. This effect may be due to the two songs having different length introductions and one may be trailing the other on their paths, a problem which could be overcome by an alignment mechanism. A contributing factor is that the area of the map in which the songs' paths mainly land is the same area in which several other songs are mapped to as well. This failure to differentiate well could potentially be addressed by selecting different features in the preprocessing stage.

Our system fails completely when we seed the search with the song *Sean Paul - We Be Burnin'*, a popular dance club type of music. The top five results for this song are presented in Figure 3.9.

In our opinion none of these songs are perceptually similar to the seed song. *Carmen Suite* is a traditional march piece, which does preserve the quick tempo and strong beat of the seed song, but contains no vocals. *Double Trouble* and *Angel of Music* both contain little instrumental and do not have a strong beat. *Susie Q* is closer to the seed song with a strong rock beat, instrumentals, and vocals. The Schubert piece, however, is a classical symphony song mainly featuring the violin section.

We believe the reason for the poor recommendations is that none of the songs from this album were selected in the 20% used to train the map; therefore the map failed to develop a neighborhood representative of this style of music. This could be solved using a stratified sampling technique. We did notice, however, that for ten of the eighteen songs on the album the song *Double Trouble* appeared within the top 3 results. This suggests that we may be seeing a similar effect as above—a crowded space requiring more discrimination. If information regarding each song's genre is known, then a stratified sampling approach could be used to drive the creation of representative neighborhoods to prevent this problem.

3.2 Query-by-Content Using SOMs

Because SOMs preserve similarity information while performing dimensionality reduction, they are well suited to powering a query-by-content system. Using this system we can avoid imposing a single query style upon the user.

3.2.1 Naïve Single-SOM Querying

The simplest and most obvious solution for querying a song library would be to train a SOM on the songs and then treat queries the same way, as if performing music recommendation as above. This is easily done by using a query as the seed for our recommendation system (Figure 3.10). However, this approach results in a degenerate solution in which the queries all map to a single area of the SOM which represents songs that have no instrumentation. This is an expected result because the query will not be similar to any of the songs based upon the musical content (other than the one theme expressed in the query). The queries tested were whistling queries and as such were unlikely to match themes occurring from guitars, singing, pianos, etc. The quasi-supervised approach allows us to compare songs and queries on a single SOM while avoiding this problem.

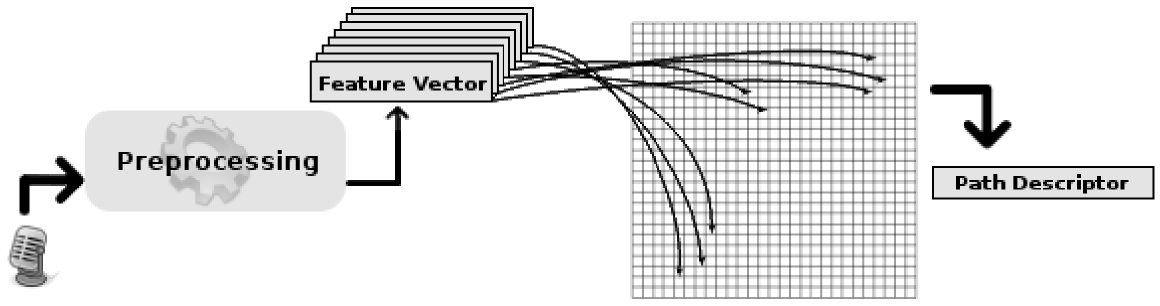


Figure 3.10: *Naïve SOM querying design.* This system works identically to the recommender system. The source, however, is provided by the user rather than from music files (see Figure 3.2). Queries are preprocessed and converted into feature vectors. The feature vectors are then mapped into the SOM to create a path descriptor, which is used to calculate the similarity to stored path descriptors of songs in the library.

3.2.2 Quasi-Supervised SOM Querying

Traditionally SOM training is unsupervised—the feature vectors themselves determine the resulting map. In quasi-supervised training, we extend the feature vectors of the target songs with the feature vectors of matching sample queries (Figure 3.11). The actual process of training the SOM is no different than before—each training instance is considered and the neighborhood around the closest matching location is updated. That is, as far as the actual SOM is concerned, nothing has changed. What has changed is how we view the vectors that make up the SOM. While the SOM is still presented with simple vectors for training, we interpret half of the vector to represent the features extracted from the actual song and the other half to represent the features extracted from the sample query.

Once the SOM is trained, we generate path descriptors for all the songs in our library just as we did in the recommendation system with one small change: when we compare the vectors describing the song to the SOM we only compare them to the first half of the vectors in the SOM. Simlarly, to generate a path descriptor for a query we compare the vectors describing the query to only the second half of the vectors in the SOM. For example, if the feature vectors for the songs each contain 13

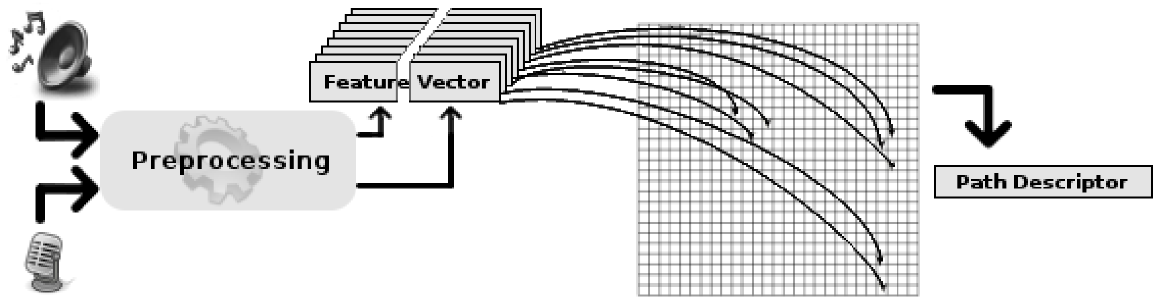


Figure 3.11: *Quasi-supervised SOM design.* Sample queries are matched with their target songs. These pairs are individually preprocessed and the resulting pair of feature vectors are concatenated to create a single feature vector. This feature vector is used to train the SOM. After training, the songs in the library are mapped into the SOM using the first half of the stored feature vectors to create path descriptors. Queries are similarly mapped, using the second half of the stored feature vectors. The path descriptors of targets and queries are thus directly comparable while still explicitly modeling the query style.

values and the feature vectors for the queries each contain 13 values, then the vectors in our SOM will each contain 26 values. This process will create a single interpolated map linking queries and targets.

As before, the entire song library is then mapped into the SOM, but using only the half of each of the vectors in the SOM which represent songs. Queries are matched to songs in a two-part process. First, a path descriptor for a query is generated by mapping the feature vectors for the query into the SOM using only the half of each of the vectors in the SOM which represent queries. Once the path descriptor for the query is generated it can be directly compared to the previously computed path descriptors for all the songs in the library just as in Section 3.1 for the recommendation system. We use the distance between path descriptors as the metric for how similar a query is to a song.

Chapter 4

Validation

The goal of any querying system is to return the specific item a user is searching for as the number one result. We can evaluate the quality of the Query-by-Content system using the position of the true target within the ordering of the results—the closer to the first result the better. Given a query $q \epsilon Q$ and an ordered list of results for q, $\tau(q)$ returns the index of the correct target song for q in the ordered results for q. Values for $\tau(q)$ range from 0 (a correct match) to $|T| - 1$, where T is the set of target songs. We compute the percentage of targets occurring as the first result using Equation 4.1. Two additional metrics are the percentage of targets occuring within the top 5 results (Equation 4.2) and top 10 results (Equation 4.3) [13]. These metrics are commonly used because, considering the difficulty of the task we are trying to solve, requiring the user to listen to five or ten possible matches is not unreasonable when searching through a database of possibly thousands of songs. The final metric we will use is the average position of the target in the ordered results (see Equation 4.4).

$$\%\, Correct = \left(\frac{\sum_{q \epsilon Q} \delta(\tau(q), 0)}{|Q|} \times 100 \right) \% \tag{4.1}$$

$$\%\, in\, Top\, 5 = \left(\frac{\sum_{q \epsilon Q} \delta(\lfloor \frac{\tau(q)}{5} \rfloor, 0)}{|Q|} \times 100 \right) \% \tag{4.2}$$

$$\%\, in\, Top\, 10 = \left(\frac{\sum_{q \epsilon Q} \delta(\lfloor \frac{\tau(q)}{10} \rfloor, 0)}{|Q|} \times 100 \right) \% \tag{4.3}$$

$$Average\ Position = \frac{\sum_{q \epsilon Q} \tau(q)}{|Q|} \tag{4.4}$$

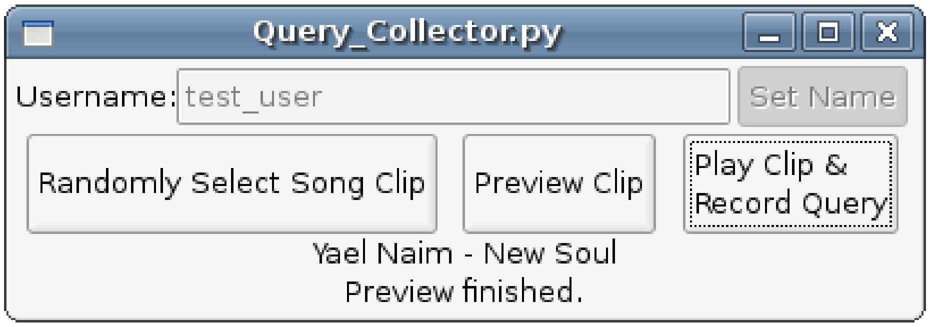
In our evaluation of the system we compare the average values of the four metrics to the expected values of those metrics if given random orderings. For example, if we have 15 songs in the training set the expected value of the 0-based position of the target song in a random ordering would be $\frac{15-1}{2} = 7$. The general form of the expected values of the percentage of songs correct, percentage of songs in the top 5, percentage of songs in the top 10, and average position are presented in Equations 4.5, 4.6, 4.7, and 4.8, respectively. To improve over a random ordering our system will need to produce a higher percentage of songs which are correct, in the top 5 results, and in the top 10 results; and produce a lower average position.

$$E[\%\, Correct] = \left(\frac{1}{numSongs} \times 100 \right) \% \tag{4.5}$$

$$E[\%\, in\, Top\, 5] = \left(\frac{5}{numSongs} \times 100 \right) \% \tag{4.6}$$

$$E[\%\, in\, Top\, 10] = \left(\frac{10}{numSongs} \times 100 \right) \% \tag{4.7}$$

$$E[Average\ Position] = \frac{numSongs - 1}{2} \tag{4.8}$$



Yael Naim - New Soul
Preview finished.

Figure 4.1: *Data collection system screenshot.* Users record 10-second queries matched to target song clips using this collection program. For any randomly presented song clip the user must listen to the clip at least once and then record a matching query in the style of their choice.

4.1 Data Collection

We collected sample queries from eight test-users. Each user chose the query style that they felt most comfortable with (see Table 4.1). A custom program was built to gather the data from the users. To begin the query collection process a song is randomly chosen from the available 783 songs, and then a 10-second segment of that song is randomly selected. The user is then asked to listen to the 10-second clip until they feel comfortable that they know the clip. The user then records a query sample of that clip while listening to the clip again. To prevent the true signal from being included in the query samples, the audio is only played through headphones. The query sample was captured using an Apex 181 USB condenser microphone at 44.1 khz. This process was repeated during two 1-hour sessions at a pace set by the user. All of the audio was then processed to extract a desired feature set. This entire process yielded a set of eight datasets of matched query-target pairs (q, t); where q is, for example, the features representing ten seconds of humming, and t is the features representing the actual audio sample the user was trying to match. These 8 datasets varied in size from 26 to 210 pairs. Figure 4.1 shows a screenshot of our data collection program.

4.2 Experimental Design

With each of these eight datasets we performed 5-fold cross validation. Each dataset was randomly divided into 5 groups as equally as possible; each group differs by, at most, one song. Each fold (approximately 20% of the dataset) was used once for training and appeared in the test set of all other folds, yielding test sets of approximately 80% of the dataset. The evaluations were run twice, once with a feature set of only the Mel-Frequency Cepstral Coefficients (MFCCs), and once with a feature set of the MFCCs and the Linear Predictive Coding (LPC) coefficients. Both the MFCC and LPC values are commonly used in MIR and Natural Language Processing systems and we chose these features because of their popularity in other systems. MFCCs are well suited to MIR tasks because they have been designed specifically to represent features which the human auditory response system perceives (see Figure 4.2) [17]. LPC is a compression algorithm which represents the n-th sample of a signal using a linear combination of the p previous samples combined with an error correction term [27]. The algorithm is based on a simple model of the human vocal system and works very well for speech.

Before feature extraction the audio signals for songs and queries are normalized so that differences in sound levels are negligible. We believe that a user querying for a part of a song will be more likely to preserve volume differences only within the query and not across the entire song. That is, only local volume changes are important. So by normalizing the signal we preserve the local volume changes while preventing volume discrepancies between songs and queries from affecting the results. All the features were extracted for every window of 100ms with 50% overlap. As with the features selected, these parameters were chosen because of their popularity in other systems.

In addition to evaluating SOMs created for each user's dataset individually, we used stratified-sampling to test the cross-user robustness of the system. We had

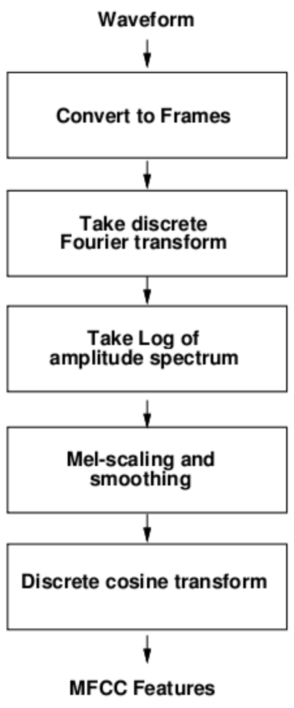

Waveform
Convert to Frames
Take discrete Fourier transform
Take Log of amplitude spectrum
Mel-scaling and smoothing
Discrete cosine transform
MFCC Features

Figure 4.2: *Process to create MFCC features [17].* The entire process of extracting MFCCs from an audio signal was designed to extract features which better model the perception of the human auditory response system.

User	Musical Skill (1-10)	Training	Query Style	Skill in Style (1-10)
A	5	None	Singing (words)	4
B	6	None	Humming	6
C	7	Some	Singing (words)	6
D	4	Some	Humming	5
E	7.5	Lots	Singing (notes)	6
F	3	Little	Whistling	5
G	7	Some	Whistling	7
H	6	None	Singing (words)	6

Table 4.1: *Test-user survey information.* Test-users provided information about how much musical training of any kind they have had, the style of querying they would use, and rated themselves on their overall musical skill and their skill in the query style they selected. These skill ratings are on a scale of 1-10 where 10 is the best.

multiple users choose to query by humming, whistling, and singing words. Therefore, for each of those query styles, we performed 5-fold cross validation using data from all users with that query style. To create the five groups, each user's dataset was randomly divided into five groups and then the groups for each user were combined. So each user's dataset was spread equally across the five groups; users with larger datasets had more songs in each group than users with smaller datasets. As a final evaluation of the cross-user robustness of the system, we also performed 5-fold cross validation over all the users regardless of query style. The groups were created as above so that each user's dataset was spread equally over the five groups. The cross-user experiments only used the MFCCs for features. All of the SOMs trained were 128x128 in size. Preliminary testing suggested this size would provide an adequate balance between required training time and system performance. Larger SOMs make the task easier but training time increases quadratically.

4.3 Results

All of the individual experiments' results are detailed in the appendix. In our overall results for each user we use the average value of each metric across the five folds and compare that average to the expected value of the metric when using randomly generated orderings (Equations 4.5-4.8). We report this comparison using the percent improvement,

$$\iota = \frac{\alpha - \upsilon}{\upsilon} \times 100 \tag{4.9}$$

where ι is the percent improvement, α is the average value of the metric, and υ is the expected value of the metric when using randomly generated orderings.

For the average position metric, lower results are better so we adjust the percent improvement equation to reflect this change:

$$\iota = \frac{\upsilon - \alpha}{\upsilon} \times 100 \tag{4.10}$$

This allows us to read all of the percent improvement values in the same manner: higher is better. It is important to remember when viewing results in this section that percent improvement is very sensitive when the values are near zero.

4.3.1 Single user, MFCC

Initially the only features extracted were the MFCCs. Overall there is definite improvement over the expected value of random orderings (Table 4.2). The SOMs trained for users A, C, and E all had 100% accuracy on all five folds over their respective training sets. The SOMs trained for user F averaged 86.15% correct with 2 folds reaching 100%. This result is most likely because these four users' training sets were relatively small—the large size of the SOM compared to the number of queries simplifies the task. This success on the training sets does seem to transfer to the test sets, though not as strongly as we would like. The standard deviations of the

	Training Set				Test Set			
	Correct	Top 5	Top 10	Position	Correct	Top 5	Top 10	Position
User A	600.28	40.00	0.00	100.00	100.00	28.00	18.01	5.63
User B	316.27	139.67	68.09	35.59	60.71	16.11	20.00	3.70
User C	960.45	112.00	6.00	100.00	0.00	0.00	6.11	4.20
User D	39.92	32.02	26.00	10.54	18.33	-4.36	22.02	1.99
User E	420.02	4.00	0.00	100.00	21.83	4.20	-6.01	1.15
User F	985.01	152.02	26.01	96.72	-40.40	3.83	6.00	1.21
User G	339.77	143.97	80.03	47.36	0.00	16.02	18.03	-0.37
User H	598.21	199.64	85.84	57.20	-20.54	15.95	0.00	0.52
μ	532.49	102.92	36.50	68.43	17.49	7.97	9.63	2.25
σ	324.26	69.34	36.14	35.44	44.89	10.76	9.97	2.05

Table 4.2: *Single user MFCC percent improvement.* μ and σ are the mean and standard deviation respectively.

metrics are rather large compared to their means. This suggests that the result of training a SOM varies from user to user. Despite the large standard deviations, the average across users of each of the metrics shows definite improvement over random orderings.

As previously mentioned, the size of the dataset compared to the dimensions of the SOM has a strong effect on how effective the SOM is at separating different data while grouping similar data. If we look at the results of the training sets again ordered by average fold size we see that there is an inverse correlation between average fold size and improvement in average position (see Table 4.3). Clearly the SOM size compared to the fold size is the dominating factor in how effective our system is in this case. However, when we look at the results for the test sets ordered in the same way the correlation disappears (see Table 4.4). So, the size of the SOM affects its ability to recall specific instances but does not seem to directly impact its generalization.

4.3.2 Single user, MFCC and LPC

A common feature to include in speech processing systems is Linear Predictive Coding. We added the coefficients for LPC to our feature set in an attempt to improve

	Training Set				
	Average Fold Size	Correct	Top 5	Top 10	Position
User E	5.20	420.02	4.00	0.00	100.00
User A	7.00	600.28	40.00	0.00	100.00
User C	10.60	960.45	112.00	6.00	100.00
User F	12.60	985.01	152.02	26.01	96.72
User H	22.40	598.21	199.64	85.84	57.20
User G	23.00	339.77	143.97	80.03	47.36
User B	29.60	316.27	139.67	68.09	35.59
User D	42.00	39.92	32.02	26.00	10.54

Table 4.3: *Single user MFCC percent improvement ordered by training set size.*

	Test Set				
	Average Fold Size	Correct	Top 5	Top 10	Position
User E	20.80	21.83	4.20	-6.01	1.15
User A	28.00	100.00	28.00	18.01	5.63
User C	42.40	0.00	0.00	6.11	4.20
User F	50.40	-40.40	3.83	6.00	1.21
User H	89.60	-20.54	15.95	0.00	0.52
User G	92.00	0.00	16.02	18.03	-0.37
User B	118.40	60.71	16.11	20.00	3.70
User D	168.00	18.33	-4.36	22.02	1.99

Table 4.4: *Single user MFCC percent improvement ordered by test set size.*

	Training Set				Test Set			
	Correct	Top 5	Top 10	Position	Correct	Top 5	Top 10	Position
User A	100.07	12.00	0.00	25.67	-19.89	-20.04	4.00	1.11
User B	19.53	32.37	32.09	11.19	101.19	56.16	32.11	5.20
User C	40.72	24.13	2.14	13.96	39.83	7.71	-6.02	1.59
User D	0.00	4.03	-2.02	-0.54	38.33	-4.03	-17.98	-0.42
User E	97.61	0.54	0.00	26.67	-38.67	-7.49	6.07	0.61
User F	-19.27	12.12	6.14	2.59	0.00	-3.93	-10.03	-0.93
User G	20.00	0.00	0.00	2.18	39.45	36.10	5.98	0.79
User H	39.91	24.33	4.12	6.82	0.00	15.95	24.01	5.19
μ	37.32	13.69	5.31	11.07	20.03	10.05	4.77	1.64
σ	42.77	12.13	11.13	10.48	43.97	25.22	16.79	2.34

Table 4.5: *Single user MFCC & LPC percent improvement.* μ and σ are the mean and standard deviation respectively.

performance. The results shown in Table 4.5 clearly show that, on the training set, adding LPC coefficients to our features was detrimental. However, the percentage correct and percentage in the top 5 improved overall in the test sets. Users B and H showed improvement on the test sets for all the metrics compared to using only MFCCs. User G improved on the test sets for all the metrics except percentage in the top 10. The results for user C were mixed, improving on the first two metrics and getting worse for the last two. Users A, D, E, and F, however, showed an overall decrease in performance. Overall there was improvement on the test sets on the first two metrics, but decreased performance on the last two. This mixed improvement comes at the cost of substantial increases in the standard deviations for all but the first metric. Clearly, the effect of including LPC coefficients to the feature set varies greatly with the user. If we order the results by dataset size, the inverse correlation between training set size and percent improvement of average position still exists, though it is much less pronounced compared to when only MFCCs were used, and, again, disappears in the test sets (see Tables 4.6 and 4.7).

	Training Set				
	Average Fold Size	Correct	Top 5	Top 10	Position
User E	5.20	97.61	0.54	0.00	26.67
User A	7.00	100.07	12.00	0.00	25.67
User C	10.60	40.72	24.13	2.14	13.96
User F	12.60	-19.27	12.12	6.14	2.59
User H	22.40	39.91	24.33	4.12	6.82
User G	23.00	20.00	0.00	0.00	2.18
User B	29.60	19.53	32.37	32.09	11.19
User D	42.00	0.00	4.03	-2.02	-0.54

Table 4.6: *Single user MFCC & LPC percent improvement ordered by training set size.*

	Test Set				
	Average Fold Size	Correct	Top 5	Top 10	Position
User E	20.80	-38.67	-7.49	6.07	0.61
User A	28.00	-19.89	-20.04	4.00	1.11
User C	42.40	39.83	7.71	-6.02	1.59
User F	50.40	0.00	-3.93	-10.03	-0.93
User H	89.60	0.00	15.95	24.01	5.19
User G	92.00	39.45	36.1	5.98	0.79
User B	118.40	101.19	56.16	32.11	5.20
User D	168.00	38.33	-4.03	-17.98	-0.42

Table 4.7: *Single user MFCC & LPC percent improvement ordered by test set size.*

	Training Set				Test Set			
Style	Correct	Top 5	Top 10	Position	Correct	Top 5	Top 10	Position
Whistling	240.21	80.41	49.98	17.11	0.00	7.98	-7.98	-1.22
Humming	0.00	0.00	14.03	6.32	-60.00	4.60	18.05	2.43
Singing	438.40	155.52	85.80	35.79	-20.00	-4.00	-11.68	-0.63

Table 4.8: *Cross user, single style MFCC percent improvement.* Each SOM was trained on a stratified sample across all users of each query style.

4.3.3 Cross User, Single Style

For three query styles (humming, whistling, and singing words) there was more than one test user. Three users queried by singing words, two users queried by whistling, and two users queried by humming. To analyze the robustness of our system, we performed 5-fold cross validation for each of those query styles. MFCCs were the only features used in these tests. The results suggest that our system can learn the training set, but is ineffective at generalizing to the test data (see Table 4.8). The results for the humming style do show improvement for three of the four metrics on the test set and only little improvement in the training set. This suggests that the training did result in a more general model and not just superficially memorizing training patterns (as experiments on the other two styles suggest happened). This may be due to the fact that the users that hummed provided the largest datasets and in this combined experiment had an average fold size of 71.60 query-target pairs—the larger training set allows the SOM to learn a more general relationship and not simply recall individual instances.

4.3.4 Cross User, Cross Style

An ideal solution to musical query-by-content would be robust across different users and query styles. The results of our stratified sampling test are presented in Table 4.9. As in the single style experiment, the system shows improvement on the training set. The test set results show slight improvement over random orderings for the average

Training Set				Test Set			
Correct	Top 5	Top 10	Position	Correct	Top 5	Top 10	Position
140.91	40.55	42.38	5.80	62.50	-15.85	-1.83	1.98

Table 4.9: *Cross user, cross style MFCC percent improvement.* Each SOM was trained on a stratified sample across all users regardless of query style.

position metric. The improvement in the percent correct metric in Table 4.9 is misleading because the values are close to zero and one fold showed enough improvement to skew the average, whereas the average of the other four folds would be exactly the expected value of random orderings. The trained SOM was not able to generalize to the test sets in this experiment.

4.3.5 Comparison to Other Systems

It is beneficial to compare our results with those of other query-by-content systems. Two issues to consider when making such a comparison are that other query-by-content systems restrict the query options of the user to a single style and these systems have been in development for over 15 years, allowing many improvements to have been made to initial designs. Our system, however, is the first we know of that allows the user to query in any style they choose. Therefore, we can expect that the more restricted and mature systems will yield better results than our new system. In addition to these issues we found that the amount of testing done by other systems we analyzed was minimal and provides very little information with which to compare our results. Of the three systems that provided experimental results, Lu, You, and Zhang tested an unspecified number of users with a total of 42 queries from 1000 songs [18]; Liu and Tsai tested four users with 10 songs each [15]; and Raju, Sundaram, and Rao tested five users with 20 songs each [25]. In comparison, we tested eight users with a total of 762 queries drawn from 783 songs. Lu, You, and Zhang report that 88% of queries produced a result within the top 10 results, which is a 270% improvement over the expected value of random orderings [18]. Liu and Tsai did not provide enough detail for us to be able to compare their results to our own. Raju, Sundaram, and Rao report that 95% of queries returned the correct result first, an 1800% improvement over the expected value of random orderings [25]. These results are, expectedly, better than ours; however, our experiments were more comprehensive and our system more robust to query style. Given the inherent differences between other, more limited query-by-content systems and our more robust system, as well as the limited results provided by those systems, we consider our comparison against random orderings to be a fair and reproducible way of evaluating our system.

Chapter 5

Conclusion

Musical query-by-content systems, like the one we have introduced, allow users to find music even when they cannot remember the artist, title, or lyrics. Most other current query-by-content systems reduce the information-rich audio signal to a string representation of only the melody; queries are similarly processed and compared using string matching techniques. Many of these systems also require that the user query in a certain style. These restrictions reduce the flexibility of the system and force the user to interact in a specific way which may be unintuitive. Our system does not have these restrictions. Using a Self-Organizing Map, we are able to create a query-by-content system which is independent of the query style of the user and is able to retain more of the acoustical content of the audio signal. This avoids the difficult task of automatically extracting a single melody and representing it in some simplified form.

Our results show that it is feasible to use a Self-Organizing Map in a query-by-content system. The system is most effective when using a dataset from a single user with a single query style. However, unlike most other musical query-by-content systems, our system does not dictate the style of the query. The user may query in whatever style is most comfortable and the system is able to remain effective. This is an important step forward in making query-by-content systems more intuitive and useful to users.

5.1 Future Work

As far as we have been able to determine this work is the first to use Self-Organizing Maps in a musical query-by-content system. Our work shows that SOMs are a viable option for creating a query-by-content system, but there is still room for improvement. As in all machine learning systems, feature selection is very important. We chose to use the most common features in the field of MIR. A SOM-based system would probably benefit from a targeted feature selection process. Along with feature selection it is important to carefully choose how the features will be extracted from the audio signal. The window size and amount of overlap of windows are the two most basic criteria in audio processing. Having used common settings for these parameters, we anticipate that the system would benefit from fine tuning. In addition to these parameters there is much work to be done in adjusting the distance metric used when training the SOM and when comparing path descriptors. We used Euclidean distance for both; however, any distance metric can be used and some may perform better than others.

The size of the SOM compared to its training set is important for determining the SOM's efficacy. We used a fixed size that we felt balanced training time with performance. The size of the SOM can be determined in a more online fashion by using a hierarchical SOM, a self-growing SOM, or both. These automatically sizing SOMs allow the system to more easily adjust to the amount of data available; however there is an associated increase in computation time. The results suggest that larger training sets allow the SOM to learn a more general relationship. In each fold we used only 20% of the dataset for training; simply using more of the data in training would likely increase performance as well.

We did not test the single user, cross-style robustness of the system and this area should be investigated. The overall cross-user, cross-style robustness of the system might be improved by training on a dataset where each user provides query samples for the same set of targets. To better analyze the suitability of this type of system, a larger study should be conducted including more users, larger datasets, datasets which match multiple users' queries to the same target, and more varied query styles.

Appendices

Appendix A

Detailed Results: Single User, Single Style, MFCCs

In this section we present the results for each user's dataset using only the Mel-Frequency Cepstral Coefficients as features. The metrics for each of the five folds are shown accompanied by the average values of the metrics across the folds, the expected values of the metrics given random orderings, and the percent improvement of the average value of each metric over the expected value. Each user has two tables, one for the training sets and one for the test sets.

Fold #	**Set Size**	**% Correct**	**% in Top 5**	**% in Top 10**	**Ave Pos**
0	7.00	100.00	100.00	100.00	0.00
1	7.00	100.00	100.00	100.00	0.00
2	7.00	100.00	100.00	100.00	0.00
3	7.00	100.00	100.00	100.00	0.00
4	7.00	100.00	100.00	100.00	0.00
Averages	**7.00**	**100.00**	**100.00**	**100.00**	**0.00**
Exp Val	**7.00**	**14.28**	**71.43**	**100.00**	**3.00**
% Imprv		***600.28***	***40.00***	***0.00***	***100.00***

Table A.1: User A Training Set

Fold #	Set Size	% Correct	% in Top 5	% in Top 10	Ave Pos
0	28.00	7.14	28.57	46.43	11.21
1	28.00	7.14	28.57	46.43	12.61
2	28.00	7.14	17.86	28.57	14.75
3	28.00	7.14	14.28	42.86	13.04
4	28.00	7.14	25.00	46.43	12.07
Averages	**28.00**	**7.14**	**22.86**	**42.14**	**12.74**
Exp Val	**28.00**	**3.57**	**17.86**	**35.71**	**13.50**
% Imprv		***100.00***	***28.00***	***18.01***	***5.63***

Table A.2: User A Test Set

Fold #	Set Size	% Correct	% in Top 5	% in Top 10	Ave Pos
0	30.00	10.00	33.33	40.00	11.43
1	30.00	30.00	56.67	73.33	6.43
2	30.00	20.00	40.00	53.33	9.77
3	29.00	4.35	31.03	48.28	10.69
4	29.00	6.90	41.38	68.96	7.72
Averages	**29.60**	**14.07**	**40.48**	**56.78**	**9.21**
Exp Val	**29.60**	**3.38**	**16.89**	**33.78**	**14.30**
% Imprv		***316.27***	***139.67***	***68.09***	***35.59***

Table A.3: User B Training Set

Fold #	Set Size	% Correct	% in Top 5	% in Top 10	Ave Pos
0	118.00	0.85	4.24	8.47	55.28
1	118.00	1.69	5.93	11.02	55.55
2	118.00	1.69	5.93	12.71	56.52
3	119.00	1.68	4.20	10.92	58.81
4	119.00	0.84	4.20	7.56	56.45
Averages	**118.40**	**1.35**	**4.90**	**10.14**	**56.53**
Exp Val	**118.40**	**0.84**	**4.22**	**8.45**	**58.70**
% Imprv		***60.71***	***16.11***	***20.00***	***3.70***

Table A.4: User B Test Set

Fold #	Set Size	% Correct	% in Top 5	% in Top 10	Ave Pos
0	11.00	100.00	100.00	100.00	0.00
1	11.00	100.00	100.00	100.00	0.00
2	11.00	100.00	100.00	100.00	0.00
3	10.00	100.00	100.00	100.00	0.00
4	10.00	100.00	100.00	100.00	0.00
Averages	**10.60**	**100.00**	**100.00**	**100.00**	**0.00**
Exp Val	**10.60**	**9.43**	**47.17**	**94.34**	**4.80**
% Imprv		***960.45***	***112.00***	***6.00***	***100.00***

Table A.5: User C Training Set

Fold #	Set Size	% Correct	% in Top 5	% in Top 10	Ave Pos
0	42.00	4.76	9.52	19.05	20.60
1	42.00	0.00	11.90	28.57	19.45
2	42.00	2.38	14.28	30.95	17.57
3	43.00	2.32	9.30	20.93	20.63
4	43.00	2.32	13.95	25.58	20.86
Averages	**42.40**	**2.36**	**11.79**	**25.02**	**19.83**
Exp Val	**42.40**	**2.36**	**11.79**	**23.58**	**20.70**
% Imprv		***0.00***	***0.00***	***6.11***	***4.20***

Table A.6: User C Test Set

Fold #	Set Size	% Correct	% in Top 5	% in Top 10	Ave Pos
0	42.00	2.38	46.67	30.95	17.40
1	42.00	2.38	14.28	28.57	17.86
2	42.00	7.14	23.81	28.57	17.86
3	42.00	2.38	14.28	35.71	19.12
4	42.00	2.38	9.52	26.19	19.45
Averages	**42.00**	**3.33**	**15.71**	**30.00**	**18.34**
Exp Val	**42.00**	**2.38**	**11.90**	**23.81**	**20.50**
% Imprv		***39.92***	***32.02***	***26.00***	***10.54***

Table A.7: User D Training Set

Fold #	Set Size	% Correct	% in Top 5	% in Top 10	Ave Pos
0	168.00	0.60	2.98	8.93	82.14
1	168.00	0.60	2.98	6.55	80.76
2	168.00	1.78	4.47	9.52	83.47
3	168.00	0.60	2.38	5.95	79.32
4	168.00	0.00	1.78	5.36	83.52
Averages	**168.00**	**0.71**	**2.85**	**7.26**	**81.84**
Exp Val	**168.00**	**0.60**	**2.98**	**5.95**	**83.50**
% Imprv		***18.33***	***-4.36***	***22.02***	***1.99***

Table A.8: User D Test Set

Fold #	Set Size	% Correct	% in Top 5	% in Top 10	Ave Pos
0	6.00	100.00	100.00	100.00	0.00
1	5.00	100.00	100.00	100.00	0.00
2	5.00	100.00	100.00	100.00	0.00
3	5.00	100.00	100.00	100.00	0.00
4	5.00	100.00	100.00	100.00	0.00
Averages	**5.20**	**100.00**	**100.00**	**100.00**	**0.00**
Exp Val	**5.20**	**19.23**	**96.15**	**100.00**	**2.60**
% Imprv		***420.02***	***4.00***	***0.00***	***100.00***

Table A.9: User E Training Set

Fold #	Set Size	% Correct	% in Top 5	% in Top 10	Ave Pos
0	20.00	15.00	30.00	45.00	9.35
1	21.00	0.00	28.57	42.86	9.81
2	21.00	4.76	19.05	52.38	10.33
3	21.00	4.76	23.81	52.38	10.48
4	21.00	4.76	23.81	33.33	11.38
Averages	**20.80**	**5.86**	**25.05**	**45.19**	**10.28**
Exp Val	**20.80**	**4.81**	**24.04**	**48.08**	**10.40**
% Imprv		***21.83***	***4.20***	***-6.01***	***1.15***

Table A.10: User E Test Set

Fold #	Set Size	% Correct	% in Top 5	% in Top 10	Ave Pos
0	13.00	84.62	100.00	100.00	0.23
1	13.00	53.85	100.00	100.00	0.54
2	13.00	92.31	100.00	100.00	0.15
3	12.00	100.00	100.00	100.00	0.00
4	12.00	100.00	100.00	100.00	0.00
Averages	**12.60**	**86.15**	**100.00**	**100.00**	**0.19**
Exp Val	**12.60**	**7.94**	**39.68**	**79.36**	**5.80**
% Imprv		***985.01***	***152.02***	***26.01***	***96.72***

Table A.11: User F Training Set

Fold #	Set Size	% Correct	% in Top 5	% in Top 10	Ave Pos
0	50.00	0.00	8.00	22.00	24.24
1	50.00	0.00	10.00	26.00	23.58
2	50.00	2.00	10.00	16.00	26.56
3	51.00	1.96	13.72	23.53	23.02
4	51.00	1.96	9.80	17.65	24.65
Averages	**50.40**	**1.18**	**10.30**	**21.03**	**24.40**
Exp Val	**50.40**	**1.98**	**9.92**	**19.84**	**24.70**
% Imprv		***-40.40***	***3.83***	***6.00***	***1.21***

Table A.12: User F Test Set

Fold #	Set Size	% Perfect	% in Top 5	% in Top 10	Ave Pos
0	23.00	17.39	60.87	91.30	4.17
1	23.00	21.74	60.87	82.61	5.74
2	23.00	30.43	39.13	69.56	6.26
3	23.00	17.39	43.48	78.26	6.17
4	23.00	8.70	60.87	69.56	6.61
Averages	**23.00**	**19.13**	**53.04**	**78.26**	**5.79**
Exp Val	**23.00**	**4.35**	**21.74**	**43.47**	**11.00**
% Imprv		***339.77***	***143.97***	***80.03***	***47.36***

Table A.13: User G Training Set

Fold #	Set Size	% Perfect	% in Top 5	% in Top 10	Ave Pos
0	92.00	0.00	6.52	13.04	46.78
1	92.00	1.09	5.43	8.70	45.55
2	92.00	1.09	5.43	15.22	46.27
3	92.00	1.09	8.70	13.04	45.41
4	92.00	2.17	5.43	14.13	44.33
Averages	**92.00**	**1.09**	**6.30**	**12.83**	**45.67**
Exp Val	**92.00**	**1.09**	**5.43**	**10.87**	**45.50**
% Imprv		***0.00***	***16.02***	***18.03***	***-0.37***

Table A.14: User G Test Set

Fold #	Set Size	% Correct	% in Top 5	% in Top 10	Ave Pos
0	23.00	30.43	60.87	78.26	5.48
1	23.00	43.48	82.61	95.65	2.74
2	22.00	9.09	63.64	81.82	5.09
3	22.00	18.18	40.91	63.64	7.50
4	22.00	54.54	86.36	95.45	2.14
Averages	**22.40**	**31.14**	**66.88**	**82.96**	**4.58**
Exp Val	**22.40**	**4.46**	**22.32**	**44.64**	**10.70**
% Imprv		***598.21***	***199.64***	***85.84***	***57.20***

Table A.15: User H Training Set

Fold #	Set Size	% Correct	% in Top 5	% in Top 10	Ave Pos
0	89.00	0.00	3.37	11.14	44.80
1	89.00	1.12	6.74	11.24	40.58
2	90.00	2.20	8.89	10.00	46.51
3	90.00	0.00	6.67	12.22	43.99
4	90.00	1.11	6.67	11.11	44.43
Averages	**89.60**	**0.89**	**6.47**	**11.16**	**44.07**
Exp Val	**89.60**	**1.12**	**5.58**	**11.16**	**44.30**
% Imprv		***-20.54***	***15.95***	***0.00***	***0.52***

Table A.16: User H Test Set

Appendix B

Detailed Results: Single User, Single Style, MFCCs and LPCs

In this section we present the results for each user's dataset using the Mel-Frequency Cepstral Coefficients and Linear Predictive Coding coefficients as features. The metrics for each of the five folds are shown accompanied by the average values of the metrics across the folds, the expected values of the metrics given random orderings, and the percent improvement of the average value of each metric over the expected value. Each user has two tables, one for the training sets and one for the test sets.

Fold #	**Set Size**	**% Correct**	**% in Top 5**	**% in Top 10**	**Ave Pos**
0	7.00	28.57	71.43	100.00	2.57
1	7.00	28.57	85.71	100.00	1.86
2	7.00	28.57	85.71	100.00	2.28
3	7.00	28.57	71.43	100.00	2.57
4	7.00	28.57	85.71	100.00	1.86
Averages	**7.00**	**28.57**	**80.00**	**100.00**	**2.23**
Exp Val	**7.00**	**14.28**	**71.43**	**100.00**	**3.00**
% Imprv		***100.07***	***12.00***	***0.00***	***25.67***

Table B.1: User A Training Set

Fold #	Set Size	% Correct	% in Top 5	% in Top 10	Ave Pos
0	28.00	7.14	14.28	32.14	14.50
1	28.00	0.00	14.28	35.71	13.86
2	28.00	3.57	14.28	42.86	12.96
3	28.00	3.57	17.86	46.43	11.57
4	28.00	0.00	10.71	28.57	13.86
Averages	**28.00**	**2.86**	**14.28**	**37.14**	**13.35**
Exp Val	**28.00**	**3.57**	**17.86**	**35.71**	**13.50**
% Imprv		***-19.89***	***-20.04***	***4.00***	***1.11***

Table B.2: User A Test Set

Fold #	Set Size	% Correct	% in Top 5	% in Top 10	Ave Pos
0	30.00	3.33	26.67	50.00	12.20
1	30.00	3.33	13.33	36.67	13.77
2	30.00	6.67	20.00	43.33	13.53
3	29.00	3.45	27.59	44.83	12.28
4	29.00	3.45	24.14	48.28	11.66
Averages	**29.60**	**4.04**	**22.34**	**44.62**	**12.70**
Exp Val	**29.60**	**3.38**	**16.89**	**33.78**	**14.30**
% Imprv		***19.53***	***32.27***	***32.09***	***11.19***

Table B.3: User B Training Set

Fold #	Set Size	% Correct	% in Top 5	% in Top 10	Ave Pos
0	118.00	0.85	5.08	9.32	53.09
1	118.00	2.54	8.47	11.86	55.86
2	118.00	1.69	7.63	11.86	57.49
3	119.00	1.68	5.88	10.08	54.69
4	119.00	1.68	5.88	12.60	57.11
Averages	**118.40**	**1.69**	**6.59**	**11.15**	**55.65**
Exp Val	**118.40**	**0.84**	**4.22**	**8.44**	**58.70**
% Imprv		***101.19***	***56.16***	***32.11***	***5.20***

Table B.4: User B Test Set

Fold #	Set Size	% Correct	% in Top 5	% in Top 10	Ave Pos
0	11.00	9.09	54.54	90.91	4.64
1	11.00	18.18	63.64	100.00	4.18
2	11.00	9.09	54.54	90.91	4.00
3	10.00	10.00	50.00	100.00	4.50
4	10.00	20.00	70.00	100.00	3.30
Averages	**10.60**	**13.27**	**58.55**	**96.36**	**4.13**
Exp Val	**10.60**	**9.43**	**47.17**	**94.34**	**4.80**
% Imprv		***40.72***	***24.13***	***2.14***	***13.96***

Table B.5: User C Training Set

Fold #	Set Size	% Correct	% in Top 5	% in Top 10	Ave Pos
0	42.00	2.38	11.90	21.43	20.00
1	42.00	7.14	14.29	23.81	20.36
2	42.00	0.00	7.14	19.05	21.31
3	43.00	2.33	11.63	18.60	22.40
4	43.00	4.65	18.60	27.91	17.79
Averages	**42.40**	**3.30**	**12.71**	**22.16**	**20.37**
Exp Val	**42.40**	**2.36**	**11.80**	**23.58**	**20.70**
% Imprv		***39.83***	***7.71***	***-6.02***	***1.59***

Table B.6: User C Test Set

Fold #	Set Size	% Correct	% in Top 5	% in Top 10	Ave Pos
0	42.00	2.38	14.28	23.81	20.60
1	42.00	2.38	9.52	23.81	20.00
2	42.00	2.38	14.28	23.81	21.33
3	42.00	2.38	14.28	26.20	19.98
4	42.00	2.38	9.52	19.05	21.14
Averages	**42.00**	**2.38**	**12.38**	**23.33**	**20.61**
Exp Val	**42.00**	**2.38**	**11.90**	**23.81**	**20.50**
% Imprv		***0.00***	***4.03***	***-2.02***	***-0.54***

Table B.7: User D Training Set

Fold #	**Set Size**	**% Correct**	**% in Top 5**	**% in Top 10**	**Ave Pos**
0	168.00	0.60	2.98	4.76	81.93
1	168.00	1.19	1.19	5.36	85.25
2	168.00	0.60	4.76	5.36	84.06
3	168.00	0.60	2.38	4.17	83.67
4	168.00	1.19	2.98	4.76	84.33
Averages	**168.00**	**0.83**	**2.86**	4.88	83.85
Exp Val	**168.00**	**0.60**	**2.98**	5.95	83.50
% Imprv		***38.33***	***-4.03***	***-17.98***	***-0.42***

Table B.8: User D Test Set

Fold #	**Set Size**	**% Correct**	**% in Top 5**	**% in Top 10**	**Ave Pos**
0	6.00	50.00	83.33	100.00	1.33
1	5.00	40.00	100.00	100.00	1.80
2	5.00	20.00	100.00	100.00	1.80
3	5.00	40.00	100.00	100.00	1.40
4	5.00	40.00	100.00	100.00	1.40
Averages	**5.20**	**38.00**	**96.67**	**100.00**	**1.54**
Exp Val	**5.20**	**19.23**	**96.15**	**100.00**	**2.10**
% Imprv		***97.61***	***0.54***	***0.00***	***26.67***

Table B.9: User E Training Set

Fold #	**Set Size**	**% Correct**	**% in Top 5**	**% in Top 10**	**Ave Pos**
0	20.00	10.00	35.00	55.00	8.70
1	21.00	0.00	23.81	42.86	10.71
2	21.00	0.00	9.52	47.62	10.43
3	21 0	23.81	47.62	47.62	10.14
4	21 0	4.76	19.05	61.90	9.14
Averages	**20.80**	**2.95**	**22.24**	**51.00**	**9.84**
Exp Val	**20.80**	**4.81**	**24.04**	**48.08**	**9.90**
% Imprv		***-38.67***	***-7.49***	***6.07***	***0.61***

Table B.10: User E Test Set

Fold #	**Set Size**	**% Correct**	**% in Top 5**	**% in Top 10**	**Ave Pos**
0	13.00	7.70	46.15	76.92	5.69
1	13.00	7.70	46.15	92.31	5.46
2	13.00	0.00	38.46	76.92	5.92
3	12.00	8.33	41.67	83.33	5.67
4	12.00	8.33	50.00	91.67	5.50
Averages	**12.60**	**6.41**	**44.49**	**84.23**	**5.65**
Exp Val	**12.60**	**7.94**	**39.68**	**79.36**	**5.80**
% Imprv		***-19.27***	***12.12***	***6.14***	***2.59***

Table B.11: User F Training Set

Fold #	**Set Size**	**% Correct**	**% in Top 5**	**% in Top 10**	**Ave Pos**
0	50.00	4.00	12.00	18.00	24.78
1	50.00	2.00	10.00	18.00	23.80
2	50.00	0.00	8.00	16.00	25.18
3	51.00	1.96	9.80	21.57	24.72
4	51.00	1.96	7.84	15.68	26.16
Averages	**50.40**	**1.98**	**9.53**	**17.85**	**24.93**
Exp Val	**50.40**	**1.98**	**9.92**	**19.84**	**24.70**
% Imprv		***0.00***	***-3.93***	***-10.03***	***-0.93***

Table B.12: User F Test Set

Fold #	**Set Size**	**% Correct**	**% in Top 5**	**% in Top 10**	**Ave Pos**
0	23.00	4.35	21.74	39.13	11.04
1	23.00	4.35	17.39	56.52	10.13
2	23.00	8.70	26.09	34.78	10.52
3	23.00	8.70	21.74	43.48	10.91
4	23.00	0.00	21.74	43.48	11.22
Averages	**23.00**	**5.22**	**21.74**	**43.48**	**10.76**
Exp Val	**23.00**	**4.35**	**21.74**	**43.48**	**11.00**
% Imprv		***20.00***	***0.00***	***0.00***	***2.18***

Table B.13: User G Training Set

Fold #	Set Size	% Correct	% in Top 5	% in Top 10	Ave Pos
0	92.00	1.09	10.87	13.04	44.60
1	92.00	0.00	6.52	11.96	43.62
2	92.00	5.43	10.87	14.13	44.38
3	92.00	0.00	4.35	6.52	47.62
4	92.00	1.09	4.35	11.96	45.49
Averages	**92.00**	**1.52**	**7.39**	**11.52**	**45.14**
Exp Val	**92.00**	**1.09**	**5.43**	**10.87**	**45.50**
% Imprv		***39.45***	***36.10***	***5.98***	***0.79***

Table B.14: User G Test Set

Fold #	Set Size	% Correct	% in Top 5	% in Top 10	Ave Pos
0	23.00	4.35	21.74	43.48	10.56
1	23.00	8.70	26.09	43.48	10.52
2	22.00	4.54	10.91	54.54	8.95
3	22.00	4.54	22.72	36.36	10.54
4	22.00	9.09	27.27	54.54	9.23
Averages	**22.40**	**6.24**	**27.75**	**46.48**	**9.97**
Exp Val	**22.40**	**4.46**	**22.32**	**44.64**	**10.70**
% Imprv		***39.91***	***24.33***	***4.12***	***6.82***

Table B.15: User H Training Set

Fold #	Set Size	% Correct	% in Top 5	% in Top 10	Ave Pos
0	89.00	0.00	6.74	16.85	42.65
1	89.00	2.25	6.74	10.11	44.34
2	90.00	2.22	8.89	16.67	40.40
3	90.00	1.11	5.56	13.33	42.77
4	90.00	0.00	4.44	12.22	39.90
Averages	**89.60**	**1.12**	**6.47**	**13.84**	**42.00**
Exp Val	**89.60**	**1.12**	**5.58**	**11.16**	**44.30**
% Imprv		***0.00***	***15.95***	***24.01***	***5.19***

Table B.16: User H Test Set

Appendix C

Detailed Results: Cross User, Single Style, MFCCs

In this section we present the results for the cross user, single style datasets using only the Mel-Frequency Cepstral Coefficients as features. The metrics for each of the five folds are shown accompanied by the average values of the metrics across the folds, the expected values of the metrics given random orderings, and the percent improvement of the average value of each metric over the expected value. Each query style has two tables, one for the training sets and one for the test sets.

Fold #	**Set Size**	**% Correct**	**% in Top 5**	**% in Top 10**	**Ave Pos**
0	36.00	13.89	30.56	50.00	13.47
1	36.00	5.56	13.89	36.11	17.06
2	36.00	8.33	22.22	41.67	13.33
3	35.00	11.43	25.71	40.00	13.57
4	35.00	8.57	34.28	42.86	14.26
Averages	**35.60**	**9.56**	**25.33**	**42.13**	**14.34**
Exp Val	**35.60**	**2.81**	**14.04**	**28.09**	**17.30**
% Imprv		***240.21***	***80.41***	***49.98***	***17.11***

Table C.1: Whistling Training Set

Fold #	**Set Size**	**% Correct**	**% in Top 5**	**% in Top 10**	**Ave Pos**
0	142.00	0.70	4.22	6.34	69.96
1	142.00	0.70	5.63	7.04	68.91
2	142.00	0.70	3.52	5.63	70.28
3	143.00	0.70	2.10	5.59	75.28
4	143.00	0.70	3.50	7.70	73.34
Averages	**142.40**	**0.70**	**3.79**	**6.46**	**71.56**
Exp Val	**142.40**	**0.70**	**3.51**	**7.02**	**70.70**
% Imprv		***0.00***	***7.98***	***-7.98***	***-1.22***

Table C.2: Whistling Test Set

Fold #	**Set Size**	**% Correct**	**% in Top 5**	**% in Top 10**	**Ave Pos**
0	41.00	9.76	29.27	46.34	12.81
1	41.00	17.07	36.58	48.78	11.88
2	40.00	22.50	40.00	52.50	11.45
3	39.00	10.26	35.90	48.72	12.08
4	39.00	7.69	17.95	35.90	14.44
Averages	**40.00**	**13.46**	**31.94**	**46.45**	**12.52**
Exp Val	**40.00**	**2.50**	**12.50**	**25.00**	**19.50**
% Imprv		***438.40***	***155.52***	***85.80***	***35.79***

Table C.3: Singing Words Training Set

Fold #	**Set Size**	**% Correct**	**% in Top 5**	**% in Top 10**	**Ave Pos**
0	159.00	0.00	3.14	5.03	79.26
1	159.00	0.63	3.77	6.92	75.83
2	160.00	0.62	2.50	5.00	82.03
3	161.00	0.62	2.48	5.59	81.57
4	161.00	0.92	3.10	5.59	81.25
Averages	**160.00**	**0.50**	**3.00**	**5.52**	**80.00**
Exp Val	**160.00**	**0.63**	**3.13**	**6.25**	**79.50**
% Imprv		***-20.00***	***-4.00***	***-11.68***	***-0.63***

Table C.4: Singing Words Test Set

Fold #	**Set Size**	**% Perfect**	**% in Top 5**	**% in Top 10**	**Ave Pos**
0	72.00	1.39	5.56	11.11	34.22
1	72.00	2.78	9.72	19.44	31.21
2	72.00	0.00	4.47	13.89	33.04
3	71.00	1.41	8.45	18.31	34.76
4	71.00	1.41	7.04	16.90	32.11
Averages	**71.60**	**1.40**	**6.99**	**15.93**	**33.07**
Exp Val	**71.60**	**1.40**	**6.99**	**13.97**	**35.30**
% Imprv		***0.00***	***0.00***	***14.03***	***6.32***

Table C.5: Humming Training Set

Fold #	**Set Size**	**% Perfect**	**% in Top 5**	**% in Top 10**	**Ave Pos**
0	286.00	0.00	2.10	5.59	143.16
1	286.00	0.70	1.75	3.85	135.75
2	286.00	0.00	2.10	3.50	134.18
3	287.00	0.00	1.74	3.14	143.94
4	287.00	0.00	1.39	4.53	139.10
Averages	**286.40**	**0.14**	**1.82**	**4.12**	**139.23**
Exp Val	**286.40**	**0.35**	**1.74**	**3.49**	**142.70**
% Imprv		***-60.00***	***4.60***	***18.05***	***2.43***

Table C.6: Humming Test Set

Appendix D

Detailed Results: Cross User, Cross Style, MFCCs

In this section we present the results for the cross user, cross style datasets using only the Mel-Frequency Cepstral Coefficients as features. The metrics for each of the five folds are shown accompanied by the average values of the metrics across the folds, the expected values of the metrics given random orderings, and the percent improvement of the average value of each metric over the expected value.

Fold #	**Set Size**	**% Perfect**	**% in Top 5**	**% in Top 10**	**Ave Pos**
0	155.00	0.64	3.87	9.03	72.16
1	154.00	0.65	3.25	7.14	74.12
2	153.00	0.65	2.61	6.53	74.79
3	150.00	3.33	6.00	12.67	66.01
4	150.00	2.67	7.33	11.33	69.31
Averages	**152.40**	**1.59**	**4.61**	**9.34**	**71.31**
Exp Val	**152.40**	**0.66**	**3.28**	**6.56**	**75.70**
% Imprv		***140.91***	***40.55***	***42.38***	***5.80***

Table D.1: Training Set

Fold #	**Set Size**	**% Perfect**	**% in Top 5**	**% in Top 10**	**Ave Pos**
0	607.00	0.16	0.33	1.48	296.49
1	608.00	0.16	0.66	1.48	302.63
2	609.00	0.00	0.82	1.48	316.00
3	612.00	0.32	0.65	1.96	276.63
4	612.00	0.65	0.98	1.63	299.76
Averages	**609.60**	**0.26**	**0.69**	**1.61**	**298.29**
Exp Val	**609.60**	**0.16**	**0.82**	**1.64**	**304.30**
% Imprv		***62.50***	***-15.85***	***-1.83***	***1.98***

Table D.2: Test Set

Bibliography

[1] P. Ahrendt, A. Meng, and J. Larsen. Decision time horizon for music genre classification using short time features. In *Proceedings of European Signal Processing Conference*, pages 1293–1296, Vienna, Austria, September 2004.

[2] E. Allamanche, Jürgen Herre, Oliver Hellmuth, T. Kastner, and C. Ertel. A multiple feature model for musical similarity retrieval. In *Proceedings of International Conference on Music Information Retrieval*, Baltimore, MD, USA, October 2003.

[3] William Birmingham, Roger Dannenberg, and Bryan Pardo. Query by humming with the vocalsearch system. *Communications of the ACM*, 49(8):49–52, 2006.

[4] Kyle B. Dickerson and Dan Ventura. Music recommendation and query-by-content using self-organizing maps. In *Proceedings of the International Joint Conference on Neural Networks*, Atlanta, GA, USA, June 2009.

[5] Michael Dittenbach, Dieter Merkl, and Adreas Rauber. The growing hierarchical self-organizing map. In *Proceedings of International Joint Conference on Neural Networks*, volume 6, page 6015, Washington, DC, USA, July 2000. IEEE Computer Society.

[6] B. Feiten and S. Günzel. Automatic indexing of a sound database using self-organizing neural nets. *Computer Music Journal*, 18(3):53–65, 1994.

[7] J. Foote. A similarity measure for automatic audio classification. In *Proceedings of AAAI Symposium on Intelligent Integration and Use of Text, Image, Video and Audio Corpora*, Stanford, CA, USA, March 1997.

[8] J. Foote, M. Cooper, and U. Nam. Audio retrieval by rhythmic similarity. In *Proceedings of International Conference on Music Information Retrieval*, Paris, France, October 2002.

[9] Asif Ghias, Jonathan Logan, David Chamberlin, and Brian C. Smith. Query by humming: Musical information retrieval in an audio database. In *Proceedings of*

ACM International Conference on Multimedia, pages 231–236, New York, NY, USA, 1995. ACM.

[10] S. Harford. Automatic segmentation, learning and retrieval of melodies using a self-organizing neural network. In *Proceedings of International Conference on Music Information Retrieval*, Baltimore, MD, USA, 2003.

[11] Kurt Jacobson. A multifaceted approach to music similarity. In *Proceedings of International Conference on Music Information Retrieval*, pages 346–348, Victoria, Canada, October 2006.

[12] T. Kohonen. *Self-Organizing Maps*. Springer, 2001.

[13] Naoko Kosugi, Yuichi Nishihara, Tetsuo Sakata, Masashi Yamamuro, and Kazuhiko Kushima. A practical query-by-humming system for a large music database. In *Proceedings of ACM International Conference on Multimedia*, pages 333–342, New York, NY, USA, 2000. ACM Press.

[14] Jorma Laaksonen, Markus Koskela, and Erkki Oja. Content-based image retrieval using self-organizing maps. In *Proceedings of International Conference on Visual Information and Information Systems*, pages 541–548, London, UK, 1999. Springer-Verlag.

[15] Chih-Chin Liu and Po-Jun Tsai. Content-based retrieval of mp3 music objects. In *Proceedings of International Conference on Information and Knowledge Management*, pages 506–511, Atlanta, GA, USA, 2001. ACM.

[16] B. Logan and A. Salomon. A music similarity function based on signal analysis. In *Proceedings of IEEE International Conference on Multimedia and Expo*, pages 745–748, Tokyo, Japan, August 2001.

[17] Beth Logan. Mel frequency cepstral coefficients for music modeling. In *In International Symposium on Music Information Retrieval*, pages 23–25, Plymouth, MA, USA, October 2000.

[18] Lie Lu, Hong You, and Hong-Jiang Zhang. A new approach to query by humming in music retrieval. In *Proceedings of IEEE International Conference on Multimedia and Expo*, pages 595–598, Tokyo, Japan, August 2001.

[19] Michael I. Mandel and Daniel P. W. Ellis. Song-level features and support vector machines for music classification. In *Proceedings of International Conference on Music Information Retrieval*, pages 594–599, September 2005.

[20] Cory McKay and Ichiro Fujinaga. Automatic music classification and the importance of instrument identification. In *Proceedings of Conference on Interdisciplinary Musicology*, pages 1–10, Montreal, Canada, March 2005.

[21] A. Meng and J. Shawe-Taylor. An investigation of feature models for music genre classification using the support vector classifier. In *Proceedings of International Conference on Music Information Retrieval*, pages 604–609, London, UK, September 2005.

[22] Jouni Paulus and Anssi Klapuri. Measuring the similarity of rhythmic patterns. In Michael Fingerhut, editor, *Proceedings of International Conference on Music Information Retrieval*, pages 150–156, Paris, France, Oct 2002.

[23] Steffen Pauws. Cubyhum: A fully operational "query by humming" system. In *Proceedings of International Conference on Music Information Retrieval*, Paris, France, October 2002.

[24] Tim Pohle, Elias Pampalk, and Gerhard Widmer. Evaluation of frequently used audio features for classification of music into perceptual categories. In *Proceedings of International Workshop on Content-Based Multimedia Indexing*, Riga, Latvia, June 2005.

[25] M. Anand Raju, Bharat Sundaram, and Preeti Rao. Tansen: A query-by-humming based music retrieval system. In *Proceedings of Indian Institute of Technology National Conference on Communications*, Chennai, India, February 2003.

[26] A. Rauber, E. Pampalk, and D. Merkl. Using psycho-acoustic models and self-organizing maps to create a hierarchical structuring of music by sound similarities. In *Proceedings of International Conference on Music Information Retrieval*, Paris, France, October 2002.

[27] Davide Rocchesso. *Introduction to Sound Processing*. 2003.

[28] Nicolas Scaringella, Giorgio Zoia, and Daniel Mlynek. Automatic genre classification of music content: a survey. *Signal Processing Magazine, IEEE*, 23(2):133–141, 2006.

[29] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293– 302, July 2002.

[30] Kris West, Stephen Cox, and Paul Lamere. Incorporating machine-learning into music similarity estimation. In *Proceedings of ACM Workshop on Audio and Music Computing Multimedia*, pages 89–96, New York, NY, USA, 2006. ACM.