

PERSONALIZED INDEXING OF MUSIC BY EMOTIONS

by

Amanda Twyla Cohen Mostafavi

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Computing and Information Systems

Charlotte

2014

Approved by:

Dr. Zbigniew Ras

Dr. Jing Yang

Dr. Wensheng Wu

Dr. Alicja Wiczorkowska

Dr. Randal Haldeman

©2014
Amanda Twyla Cohen Mostafavi
ALL RIGHTS RESERVED

ABSTRACT

AMANDA TWYLA COHEN MOSTAFAVI. Personalized indexing of music by emotions. (Under the direction of DR. ZBIGNIEW RAS)

How a person interprets music and what prompts a person to feel certain emotions are two very subjective things. This dissertation presents a method where a system can learn and track a user's listening habits with the purpose of recommending songs that fit the user's specific way of interpreting music and emotions. First a literature review is presented which shows an overview of the current state of recommender systems, as well as describing classifiers; then the process of collecting user data is discussed; then the process of training and testing personalized classifiers is described; finally a system combining the personalized classifiers with clustered data into a hierarchy of recommender systems is presented.

ACKNOWLEDGMENTS

I would first of all like to thank my advisor, Zbigniew Ras, for his support and guidance throughout my graduate career and for introducing me to music information retrieval research. I would also like to thank my committee members, Alicja Wieczorkowska, Wensheng Wu, Jing Yang, and Randal Haldeman, for all of their time and advice throughout this process.

Additionally I would like to thank everyone at the Music Technology Group at Universitat Pompeu Fabra, and in particular Perfecto Herrera and Cyril Laurier, for working with me, teaching me, and inspiring me as a researcher and a music lover. I would also like to acknowledge two previous KDD lab students, Dr. Rory Lewis and Dr. Mike Jiang, who took me under their respective wings and taught me the grad school basics.

I would not have been able to survive this dissertation without the patience and support of my friends and family

Finally I would like to thank Behrooz Mostafavi, the only reason I was able to finish this dissertation at all with any semblance of my heart, soul, and sanity in tact. He has been a far better partner in every sense of the word than I could have hoped for, a better friend than I sometimes deserved, and a constant source of love and support.

This material is based upon work supported by the National Science Foundation under Grant Nos. OISE-0730065 and IIS-0414815

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION AND MOTIVATION	1
CHAPTER 2: STATE OF THE ART	3
2.1. Music Emotion Detection and Indexing	3
2.2. Recommendation Systems	8
2.3. Classifiers	13
2.4. Hierarchical Classifiers	17
CHAPTER 3: PERSONALIZED CLASSIFIERS	21
3.1. Data Collection and Construction	21
3.2. Developing Personalized Classifiers	39
CHAPTER 4: CLUSTER DRIVEN CLASSIFIERS	44
4.1. Cluster Algorithms and Distance Measures	44
4.2. Building Cluster-Driven Classifiers	48
CHAPTER 5: HIERARCHY OF RECOMMENDER SYSTEMS	61
5.1. Data Storage	61
5.2. User Generalization	65
5.3. New User Placement	66
5.4. Evaluation	68
CHAPTER 6: CONCLUSION AND FUTURE WORK	76
6.1. Future Work	76
6.2. Conclusion	80
REFERENCES	82

CHAPTER 1: INTRODUCTION AND MOTIVATION

With the average size of a person's digital music collection expanding into the hundreds and thousands, there is a need for creative and efficient ways to search for and index songs. This problem shows up in several sub-areas of music information retrieval such as genre classification, automatic artist identification, and instrument detection. Here we focus on indexing music by emotion, as in how the song makes the listener feel. This way the user could select songs that make him/her happy, sad, excited, depressed, or angry depending on what mood the listener is in (or wishes to be in). However the way a song makes someone feel, or the emotions he associates with the music, varies from person to person for a variety of reasons ranging from personality and taste to upbringing and the music the listener was exposed to growing up. This means that any sort of effective emotion indexing system must be personal and/or adaptive to the user. This is so far a mostly unexplored area of Music Information Retrieval (MIR) research, as many researchers that attempt to personalize their music emotion recognition systems (see [64]) do so from the perspective of finding how likely the song is to be tagged with certain emotions rather than finding a way to create a system that can be personalized.

There are two possible approaches for creating a personalized system; one which groups users into clusters, and classifies songs differently for each cluster, and one which learns a user's particular tagging habits and creates a classifier unique to that

user. The clustering approach is much quicker and is personalized to the user almost immediately once he is put into the correct group, but there is a higher possibility of songs being classified incorrectly (according to the user). The unique classifier approach, on the other hand, would be more accurate but it would take more time for the user's own unique classifier to be created. This dissertation compares and contrasts the two approaches, and proposes a system that could combine the two.

CHAPTER 2: STATE OF THE ART

2.1 Music Emotion Detection and Indexing

The usual approach to creating a music emotion detection/indexing system is to start with a database of music, have one or more users annotate the music with emotions, extract audio features, and use those features to train classifiers to detect emotion. There is a large amount of variety in this process, namely in the construction of the audio dataset, the system of emotion annotation, the features extracted (and the method for extracting features), and the classifiers used. These variations are described in further detail below (with the exception of the classifier algorithms, which will be discussed in further detail in Section 2.3).

2.1.1 Datasets

When discussing the type of audio used in the initial training dataset for music emotion indexing systems, this could be referring to either the type of audio data (MIDI files, MP3s, or WAV files) or the predominant type of music in the audio (classical, popular, etc.). When referring to the type of audio data, MIDI has been used in some cases, principally by Grekow in [22] and [21], as well as Muyuan et al. in [51]). MIDI does offer several advantages as a data type that stem from its digital nature. Much of the information in the music regarding pitch and tempo are coded in the file information. This means less feature extraction is required, and

more effort can be put into fine-tuning the algorithm being tested. MIDI files can also be manipulated more easily than real audio, which can alter perceived emotion if needed as shown in [27] and [28]. However, while MIDI is easy to work with and manipulate, it is not the kind of audio most often listened to. MP3 and WAV audio are more common, and since they are both based in real audio recordings any viable music emotion indexing system has to be able to work with real audio and adapt to the unique challenges presented with that particular music format. That is why in most modern systems the initial dataset is based in real audio, [13], [48], and [18] to name only a few.

When referring to the predominant type of music used in the dataset there is some more flexibility. There are systems that choose to compose their datasets from film soundtracks. One of the most significant examples of this method is in [15]. The reasoning from the authors was that since film scores are simultaneously the least known (meaning the perceived emotion would be more likely to be based on the actual music rather than some memory or association from the user) and the most emotionally stimulating pieces of music (since film music is usually composed to evoke specific emotions depending on the context of the film at that point) a dataset composed of this music would be the best for emotion annotation. The dataset composed in this paper has since been utilized in several other systems such as [?]. Other systems use a mix of genres ranging from Pop/Rock in [48] and [53] to Classical [44].

2.1.2 Emotion Modeling

There are two distinct modeling methods commonly employed in music emotion indexing systems: dimensional, which views possible emotions on a multi-dimensional plane, and categorical which views possible emotions as a selection of one or more words (this is also described as discrete modeling). The authors in [15] did a study on the validity of dimensional and categorical methods for music emotion recognition and found that both models produced comparable results in terms of consistency (although the dimensional model was less consistent when dealing with pieces where the emotion was ambiguous).

Among the systems that use dimensional emotion modeling, the most common approach tends to be based on Thayer’s 2-dimensional plane of valence (how positive or negative the emotion is) and arousal (the intensity of the emotion, excitement or anger having higher arousal than depression or peacefulness). This plane was later extended to Russell’s circumplex model of affect, where words were added to the dimensional plane and placed in opposing places [56]. Dimensional emotion modeling is tempting to use in Music Emotion Retrieval (MER) studies, since it allows songs to be annotated in a more consistent way. The logic is that words that describe emotions are open to interpretation depending on the user, whereas with dimensional modeling all the user has to ask him or herself is how positive/negative the song is and how high/low is the song’s energy. From a visual standpoint, all a user has to do is select a point on the dimensional plane (or a trajectory of points, such as in [64]) that represents the mood that fits the song (or in the case of retrieval, the mood of

the song that they would like to listen to).

The basic assumption behind discrete modeling is that all human emotion falls under a set of basic emotions (or emotion categories) regardless of a person's temperament or background. This theory was put forward principally by Paul Ekman in [17] and [16] based on the existence of universal facial expressions (on a basic level, everyone reacts the same to similar stimuli). Throughout the course of music and emotion research the basic emotion theory has been expanded and adapted to fit with how people usually describe music. However no consistency has been achieved among MER researchers with regards to which emotions (or how many emotions) to use. In their review of current MER systems, Yang and Chen in [63] found categorical systems using anywhere from three to thirteen emotion labels. The authors also observe that there is a great deal of issue with granularity (how specific the emotion categories should be without resulting in a number of labels high enough that the user is overwhelmed) and ambiguity (different emotion words can be interpreted differently as individual emotions, as well as in relation to other terms). However, as mentioned earlier, despite these criticisms the discrete method of modeling emotions performed almost as well as dimensional models, and even worked better for songs with ambiguous emotions [15]. The discrete model also allows the user to think about the emotion of a song in ways they are used to. A user would not likely be thinking about the emotion of a song in terms of valence or arousal, but whether the song makes them happy, sad, excited, etc.

2.1.3 Feature Extraction

Feature extraction is an essential part of any MIR work and emotion detection in particular. The feature extraction process allows the various complex auditory properties of the music such as rhythm, key, and timbre to be translated into data, something a computer system can understand. The features used in this research will be discussed in greater detail in Section 3.1.2. However there has been extensive work done on the calculation and extraction of musical features and how they can relate to music emotion recognition. In particular, a study was done by Laurier, Lartillot, Eerola, and Toiviainen in [37] to find the relationship between particular musical features and a song's associated emotion. Using a dataset of 110 songs categorized into one of five basic emotion categories (fear, anger, happiness, sadness, tenderness), the authors found several correlations between audio features and these emotion categories. Features associated with perceived dissonance were highly correlated with fear and anger while features that detected a song's mode turned out to be a very good predictor of whether the song was perceived to have a positive or negative emotion (songs in a major mode tended to be happy or tender, while songs in a minor mode tended to be sad or fearful). Additionally features that calculate perceived speed and loudness could easily predict the level of arousal perceived in a song [37]. These results are not unexpected in and of themselves, of course songs that are loud, fast, consonant, and in a major mode would be viewed as "happy" while songs that are soft, slow, dissonant, and in a minor mode would be viewed as sad or fearful. What these results do show is the importance of feature extraction in music

emotion retrieval and indexing, as well as the fact that the things humans listen to when determining a song’s emotion can be extracted from raw audio and calculated.

Other researchers have made similar observations about the relationship between audio features and emotion in music. The authors in [44] divided all of their extracted features into groups based not only on what aspects of the audio the features pertained to (timbral features and rhythmic features) but also how the features related to the emotion(intensity features) [44]. An additional study described in [20] found similar results to [37] using four emotion categories (happy, sad, angry, tender) while also suggesting changes that could be made to music in a synthesized performance to evoke specific emotions. For example, to make a song happier the authors suggested speeding it up to a moderately fast tempo, raise or lower the sound level to a moderately loud level, and make the articulation more staccato (sharper).

2.2 Recommendation Systems

Recommender systems typically conform to two common approaches: collaborative filtering and content-based filtering. Collaborative filtering looks at various properties of the user and give recommendations based on what similar users (or users with similar preferences) liked. Conversely, content-based filtering gives recommendations based on which other items are similar to an item that the user liked. In other words, while collaborative filtering analyzes the user content-based filtering analyzes the items being recommended. Music recommendation systems more specifically tend to fall under one of these two categories, but several hybrid systems exist as well as emotion based systems.

Both approaches have their advantages and disadvantages. Collaborative filtering systems are considered very practical and easy to implement on large datasets. However collaborative systems are prone to a cold start problem, where new items that have not been rated by other users or detected cannot be properly analyzed in the context of the filtering system leading to inaccurate recommendations (see [65]). Content-based systems on the other hand do not have the cold start issue; since the recommender analyzes the content of the item independent of other users, a new item does not have to be previously rated in order to be accurately recommended (or not). However it does take time for the user to build up enough of a profile before a content-based system can accurately give recommendations [42].

2.2.1 Content Based Systems

The authors in [3] approached this problem in what could best be called a content-based way, in that their focus was on individual users (or simulated users, as was the case for this paper). However, they viewed it as a sequence problem with user-specified constraints, where the user would ask for a specified number of songs. This however was a very early study that, in contrast to other studies, did not actually involve users. A later system utilized C4.5 decision trees combined with artificial neural networks to create a content-based filtering system. Based on music the user had listened to and rated, the authors built individual artificial neural networks (or C4.5 decision trees, for another part of the study) for each user [41].

The authors in [4] created a personalized content-based user recommender system with a visualization element in the form of a "musical avatar". The authors deter-

mined user preference based on a set of example songs provided by the user which demonstrated their tastes. They then extracted low level features and assigned a set of semantic descriptors describing the song's genre, mood, instruments, rhythm, and tempo using support vector machines for each descriptor class. Finally the authors tested various measures for determining songs to recommend based on the user preferences [4]. This method of eliciting user preference based on content offers some advantages in that all the user has to do is provide examples of songs they like. However the authors note that the system can be enhanced by relying on listening statistics to determine user preference (thus taking some of the work away from the user, and allowing the system to adapt to the user) [4].

2.2.2 Collaborative Filtering Systems

With much of music listening now taking place online MIR researchers have begun to take advantage of the internet, and social networking in particular, to build collaborative filtering music recommender systems. Paul Lamere in [33] puts forward the idea of social tags, where tags attached to music by individuals are combined into a pool of tags created collaboratively. Last.fm uses collaboratively built tags principally in its online music system. However Lamere notes that consistency is a problem with social tags since there is no standard set of vocabulary to use (although many sites that utilize tagging attempt to suggest commonly used tags to users).

In 2000 an early collaborative filtering system was created based on data gathered from a web-crawler. The authors experimented with combining web data collected from music sites and blogs expressing opinions on music with the logs from a server

which contained a large amount of music available to download. This allowed the authors to create a set of simulated users. The amount of improvement gained by combining these data sources depended on the algorithm used to classify the music; while kNN did not show any significant improvement, the weighted majority algorithm was greatly improved [11]. Later the authors in [60] attempted to overcome the impersonal nature of social tags by incorporating semantic analysis of the tags into the song analysis. The authors determined that three entities existed in social tagging systems: users, tags, and items and modelled these entities using 3-order tensors. The higher-order singular value decomposition technique was then applied [60]. On the other hand the creators of MusiDB in [59] used collaborative filtering to perform two levels of prediction: whether a user would like a given album and what new album to recommend to a user. They then used a weighted kNN algorithm to predict user reactions and recommend music.

2.2.3 Hybrid Systems

Hybrid systems attempt to overcome the flaws of content-based and collaborative filtering systems by combining approaches. The way in which these two approaches are combined depend on the flaws the authors wish to fix. The authors in [43] for example wished to solve both the cold start issue inherent in collaborative systems and the lack of serendipity in content-based systems. The algorithm they developed computes the prediction of a song based on content based, collaborative, and emotion-based systems and weights the results of all three based on the user's preferences. The authors in [9] took a similar approach with MIDI files by implementing collaborative,

content-based, and general popularity measures into one music recommender system. The authors first classified music into music groups (based on similarity derived by feature extraction) and users into user groups based on their listening histories. However unlike the authors in [43] the authors in [9] do not combine the three methods in order to give one set of recommendations. Rather the user is given the option to view recommendations returned by each of the three recommendation methods. The authors additionally state that the main purpose of the collaborative filtering algorithm in their system is to overcome the lack of serendipity that would have happened with content-based filtering alone. This also allowed the authors to perform experiments and compare each method. The authors found that generally their content-based method had higher precision, achieving between 35-39% based on classification by a single feature and 51-62% based on classification with multiple features. However the authors concluded that for the sake of providing surprising recommendations (in addition to recommendations of immediate interest to the user) all recommendation methods should be utilized [9].

The authors in [65] also sought to solve both the cold-start problem found in collaborative systems and the lack of serendipity in content based systems with their hybrid approach. They achieved this by modifying the three-way aspect model (a type of Bayesian network) to work with audio (rather than documents, which was the algorithm's original purpose) in order to calculate unobservable user preferences. They combined this with statistical estimations of user relations and item ratings in order to integrate the collaborative aspect. This method was found to outperform conventional content-based and collaborative filtering methods while still being able

to recommend some pieces with no ratings comparably with content-based filtering [65].

It should be observed that in both [65] and [9] the authors took the perspective of using collaborative filtering to introduce serendipity and novel recommendations rather than to improve overall recommendation accuracy. It can even be argued for [65] in particular that the authors focused far more on the content-based aspect of their system (which in and of itself is an excellent way to elicit latent user preferences in music). This shared perspective could be due in part to the fact that computationally, for audio in particular, item-based similarity is much more complex than user similarity. It may also show a bias toward content-based filtering in music recommendation in general.

2.3 Classifiers

There are five classifiers that will be trained and tested in this work. These classifiers were selected due to their common use and significance in Knowledge Discovery in Databases (KDD) and MIR:

- Decision Tree
- Support Vector Machines (SVM)
- Neural Net
- Random Forest
- Instance Based K-Nearest Neighbors Classifier (IBk)

The following sections describe these classification algorithms and their significance to MIR.

2.3.1 Decision Tree

For this classifier, we used the J48 classifier in Weka. This is an implementation of the C4.5 decision tree classifier, first outlined in [54]. C4.5 creates a decision tree based on splitting data on the most informative attributes (found by calculating information gain and splitting on the attribute with the highest gain). The resulting tree is then used to classify new instances by assigning them to the correct nodes in the tree (based on the value of the attributes used to split the tree).

C4.5, and its J48 implementation, has been used in several cases for musical instrument classification as well as genre classification. Castan, Ortega, and Lleida trained a C4.5 classifier on a selection of audio features and then used that classifier to classify audio frames based on whether they contained only speech or voices, only instruments, or a mix of both. Their results seemed very promising with an almost 100% success rate. However this probably had to do in part with the simplicity of the classification problem the authors chose, see [8].

This does not discount the usefulness of the C4.5 classifier for music classification. In fact in [66] in a comparison of the J48 implementation of C4.5 to Bayesian network, logistic regression, and locally weighted learning classification models for musical instrument classification, J48 was almost universally the most accurate classifier (regardless of the features used to train the classifier). The classification of musical instrument families (specifically string or woodwind) using J48 ranged in ac-

curacy from 90-92%, and the classification of actual instruments ranged from 60-75% for woodwinds and 60-67% for strings. This knowledge was used in continuing research in [31], [40], and [39] to develop classification techniques based on hierarchy and the results of signal separation techniques.

2.3.2 Support Vector Machines

Support Vector Machine (SVM) classification is one of the more common algorithms used in MIR. It is based on the construction of hyperplanes around the data points such that the space between the planes and the data points is maximized.

SVM is the principal classifier used in a variety of tasks, such as [38] for mood classification, [45] for artist identification (compared with k-nearest neighbors and Gaussian Mixture Models), and [53] for mood tracking. It has also been evaluated beside other classifiers in [58] for genre identification. These evaluations have shown the SVM classifier to be remarkably accurate, particularly in predicting mood. However in [24] SVM performed very poorly in comparison to Gaussian mixture model and support vector regression. This could have something to do with using a categorical approach to emotion annotation as opposed to a dimensional approach. Since annotations in a dimensional annotation space can be viewed as real numerical results a regression approach is more appropriate. SVM on the other hand seems to work very well with categorical emotion annotation.

2.3.3 Random Forest

The Random Forest algorithm was first proposed by Breiman in [5] and is based on creating a set of trees for prediction using different samples of the training dataset.

Once trained, new instances are classified based on the results of all the tree classifiers for that new object.

This algorithm was used principally in [32] for instrument classification in noisy audio. Sounds were created with one primary instrument and artificial noise of varying levels added in incrementally. The authors found that the percentage error was overall much lower than previous work done with SVM classifiers on the same sounds up until the noise level in the audio reached 50%.

2.3.4 IBk

IBk is an implementation of the k-nearest neighbors (kNN) algorithm, which was first proposed in [1]. In this algorithm, instances are classified based on the class values of the k closest data points.

kNN had been evaluated previously in [46] and [58] for genre classification. [46] achieved a 90%-98% classification accuracy by combining kNN and Neural Network classifiers and applying them to MIDI files using a 2-level genre hierarchical system. The authors of [58] on the other hand only achieved a 61% accuracy at the highest using real audio and k of 3.

The kNN algorithm was also used in [30] for instrument classification in polyphonic music. Rather than simply return the majority class, the kNN algorithm ranked instrument possibilities based on the class labeling of the nearest k frames. Then the top 2-6 instrument candidates were returned as the instruments present in the test audio with the expectation that the instruments returned would be all the instruments present at varying levels in the audio. With a high k (5) and no percussion instruments

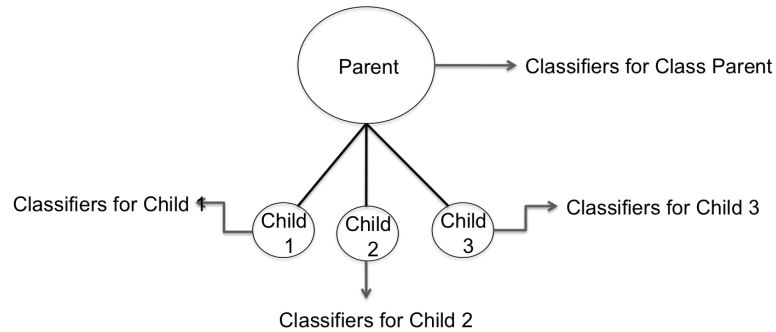


Figure 1: A general example of hierarchical classifiers

considered in the audio, the recognition rate was 87%.

2.4 Hierarchical Classifiers

The classifiers employed in this system are built around a hierarchical structure. At each level for each node of a tree there is an associated classifier (or in the case of this system a set of associated classifiers). A very general example of this idea is shown in Figure 1.

This idea has been employed in several places within the MIR community as well as the field of data mining at large. Freitas and de Carvalho in [19] outlined the general types and methods of hierarchical classification, as well as methods of evaluating them (with a focus on bioinformatics). In particular, the authors define a few key approaches to the hierarchical classification problem:

- Focusing the classification algorithm on only one level of the hierarchy (predicting only at the highest level, or the lowest leaf level, of the hierarchy).
- Applying the same classification algorithm at every level of the hierarchy independently

- The big bang approach, where one classifier is trained based on the entire hierarchy, where predictions are given for each level of the hierarchy at once
- The top down approach, where classifiers are trained at each level, resulting in at least one classifier per level (but can be several). This is closest to the approach taken in building classifiers for this recommendation system (details will be explained further on)

Many of the classification problems within MIR can be interpreted in a hierarchical fashion simply because of the way music is already interpreted and discussed. Instruments are already grouped hierarchically (the wind section, containing flute, clarinet, oboe, and saxophone, the brass section containing trumpet, horn, trombone, etc); genres can be viewed as broad genres containing smaller more specific genres. Additionally, as discussed in Section 2.1.2, emotions are very often viewed in a hierarchical fashion, whether the emotions are modeled discretely or dimensionally. These existing hierarchies have been utilized in [52] for upper level emotion classification, [21] for upper and lower level emotion classification, [57] for genre, and in [2] for extracting emotion from spoken words to varying degrees of success.

In [57] the authors implemented a hierarchical classification system for predicting genre. The authors proposed two approaches, based on combining common hierarchical classification approaches; combining a top down approach to training the classifiers with feature representation selection at each level and one classification algorithm; and combining the previous method with classifier selection at each node. The latter approach, with classifier and feature selection combined, resulted in the

highest improvement over the standard top down representation with single classification algorithms (69% at the top level, 78% at the middle level, and 73% at the leaf level).

The authors in [52] and [21] based their emotion hierarchies on the 2-dimensional model (also used as the emotion model for the recommender system presented in this dissertation, see Figure 7), where the quadrants are viewed as the upper level of the hierarchy and individual emotions were viewed as leaves. The authors in [52] focused more on tracking mood throughout the song and implementing a multi-label classification system. The authors therefore took the approach of classifying only on the upper level. The authors in [21] on the other hand applied classification to both levels of the hierarchy. They used two variations on the kNN algorithm and applied them in a similar top down fashion, with classifiers trained for each emotion on each level. The result was classifier accuracy ranging from 82-90% on the upper level, and 40-92% on the lower level. This result likely could have been improved with the testing of other classification algorithms on either level, but it is still encouraging for hierarchical classification.

The authors in [2] conversely used an emotion hierarchy based on a discrete emotion model based on the universal emotion described in [16] (with the exception of "boredom", which the authors used in place of "surprise"). These emotions were grouped by broad similarity (the authors tested two forms of grouping, the principal difference between the two groupings being the placement of "disgust"), and classified using either multilayer perceptron (MLP) or hidden Markov models(HMM). Overall, their best results came from using a 3-group hierarchy (two emotion groups, with

"disgust" placed as its own group), with a classification rate of 71%.

CHAPTER 3: PERSONALIZED CLASSIFIERS

3.1 Data Collection and Construction

3.1.1 User Data

We have created a questionnaire so that individuals can go through multiple times and annotate different sets of music based on their moods on a given day. This has given us almost 400 samples.

3.1.1.1 Questionnaire Structure

The Questionnaire is split into 5 sections

- Demographic Information (where the user is from, age, gender, ethnicity)
- General Interests (favorite books, movies, hobbies)
- Musical Tastes (what music the user generally likes, what he listens to in various moods)
- Mood Information (a list of questions based on the Profile of Mood States)
- Music Annotation (where the user annotates a selection of musical pieces based on mood)

The demographic information section is meant to compose a general picture of the user. The questions included ask for ethnicity (based on the NSF definitions), age,

what level of education the user has achieved, what field they work or study in, where the user was born, and where the user currently lives. Also included is whether the user has ever lived in a country other than where he/she was born or where he/she currently lives for more than three years. This question is included because living in another country for that long would expose the user to music from that country (see Figure 2 for a visual of this section).

The general interests section gathers information on the user's interests outside of music. It asks for the user's favorite genre of books, movies, and what kind of hobbies he/she enjoys. It also asks whether the user enjoyed math in school, whether he/she has a pet or would want one, whether he/she believes in an afterlife, and how he/she would handle an aged parent. These questions are all meant to build a more general picture of the user (see Figure 3).

The musical tastes section is meant to get a better picture of how the user relates to music. It asks how many years of formal musical training the user has had, as well as his/her level of proficiency in reading or playing music if any. It also asks what genre of music the user listens to when he/she is happy, sad, angry, and calm (see Figure 4).

The mood information section is a shortened version of the Profile of Mood States [47]. The Profile of Mood States asks users to rate how strongly he/she has been feeling a set of emotions over a period of time from the following list of possible responses:

- Not at all

- A little
- Moderately
- Quite a bit
- Extremely

The possible emotions are listed below:

- Tense
- Shaky
- Uneasy
- Sad
- Unworthy
- Discouraged
- Angry
- Grouchy
- Annoyed
- Lively
- Active
- Energetic

- Worn Out
- Fatigued
- Exhausted
- Confused
- Muddled
- Efficient

This is the section that is filled out every time the user returns to annotate music, since their mood would affect how they annotate music on a given day. These answers are later converted into a mood vector for each session, which describes the user's mood state at the time of the session (see Figure 5).

Finally, the music annotation section is where users go to annotate a selection of songs. 40 songs are selected randomly from a set of 100 songs. The user is then asked to check the checkbox for the emotion he/she feels in the music, along with a rating from 1-3 signifying how strongly the user feels that emotion (1 being very little, 3 being very strongly). The user has a choice of 16 possible emotion annotations, based on a 2-D hierarchical emotional plane (see Figure 6).

When the user goes through the questionnaire any time after the first time, he only has to fill out the mood profile and the annotations again. Each of these separate sections (along with the rest of the corresponding information) is treated as a separate user, so each individual session has classifiers trained for each emotion, resulting in 16 emotion classifiers for each user session to be clustered.

Emotion Indexing Questionare

Hello, and thank you for taking the time to fill out this test questionnaire. What will happen is first you will be asked about your general background, and then you be asked to assign an emotion to the pieces played for you. Enjoy!

Part 1-1: Demographic Information

Race/Ethnicity:

☐ Hispanic or Latino

☐ American Indian or Alaskan Native

☐ Asian

☐ African American

☐ Native Hawian or Pacific Islander

☐ White (non-Hispanic)

☐ Other

Age

Gender:

☐ Male

☐ Female

What country were you born in?

What country do you currently live in?

Please list any other countries you have lived in longer than three years, if there are any. Otherwise leave the following box blank.

What is your level of education?

What field are you studying/working in?

Figure 2: The demographic information page of the questionnaire

Emotion Indexing Questionare

Part 1-2: General Interests

Did you enjoy math in school

☐ Yes

☐ No

What genre of movies do you enjoy

What kind of books do you enjoy

Do you have, or do you currently want, a pet

☐ Yes

☐ No

What kind of activities do you enjoy

Do you believe in life after death

☐ Yes

☐ No

☐ Unsure

How will you help your parents when they grow old, assuming there are no financial or logistical constraints

Figure 3: The general interests page of the questionnaire

Emotion Indexing Questionare

Part 1-3: Musical Preferences

What formal musical training do you have (check all that apply):

- ☐ Little to None
- ☐ Basic (you can identify notes on a keyboard)
- ☐ Intermediate (you can read music in at least one clef, played an instrument or had vocal training)
- ☐ Advanced (you play or sing on a regular basis and/or took at least one college level music course)
- ☐ Recieved a Bachelors degree in music
- ☐ Recieved a Graduate degree in Music
- ☐ Foreign (studied, played, or had frequent exposure to music not in the western-classical style)

How many years of formal musical training have you had?:

What kind of music do you listen to when you are happy

What kind of music do you listen to when you are sad

What kind of music do you listen to when you are angry

What kind of music do you listen to when you are calm

Figure 4: The musical taste/background information page of the questionnaire

Emotion Indexing Questionare

Part 1-4: Mood Questions

Describe how you have been feeling the past week (including today) by selecting an appropriate box after each emotion

Tense

☐ Not at all ☐ A Little ☐ Moderately ☐ Quite a bit ☐ Extremely

Angry

☐ Not at all ☐ A Little ☐ Moderately ☐ Quite a bit ☐ Extremely

Worn Out

☐ Not at all ☐ A Little ☐ Moderately ☐ Quite a bit ☐ Extremely

Figure 5: Part of the mood state information page of the questionnaire. This section is filled out every time the user reenters the questionnaire (the user starts on this page once he/she has filled out the rest of the questionnaire once)

3.1.1.2 Emotion Model

This model was first presented in [22], and implements a hierarchy on the 2-dimensional emotion model, while also implementing discrete elements. The 12 possible emotions are derived from various areas of the 2-dimensional arousal-valence plane (based on Thayer's 2 dimensional model of arousal and valence [61]). However there are also generalizations for each area of the plane (energetic-positive, energetic-negative, calm-positive, and calm-negative) that the users can select as well. This compensates for songs that might be more ambiguous to the user; if a user generally knows that a song is high-energy and positive feeling but the words excited, happy, or pleased do not adequately describe it, they can select the generalization of energetic-positive.

Emotion Indexing Questionnaire Part 2

Thank you for providing your background. You will now be asked to assign a set of emotions to a given selection of music. Please feel free to check all emotions that the music makes you feel. If none of the specific emotions listed fits, please choose one of the broader emotional categories (Energetic-Positive, Energetic-Negative, Calm-Positive, Calm-Negative). In the box below the selected emotion please rate how strongly you feel the emotion based on the following scale.

- 1 - You barely feel this emotion
- 2 - You feel this emotion a moderate amount
- 3 - You feel this emotion very strongly

NOTE: Google Chrome users may have some problems playing the audio in the annotation section. If this happens to you, please switch to another browser

Part 2: Emotion Assignment



Song #	Clip	Emotion
1		<div style="display: flex; justify-content: space-between; align-items: flex-start;"> <div style="display: flex; justify-content: space-around; width: 100%;"> <input type="checkbox"/> Pleased <input type="checkbox"/> Happy <input type="checkbox"/> Excited <input type="checkbox"/> Sad <input type="checkbox"/> Bored <input type="checkbox"/> Depressed <input type="checkbox"/> Nervous <input type="checkbox"/> Annoyed <input type="checkbox"/> Angry <input type="checkbox"/> Calm <input type="checkbox"/> Relaxed <input type="checkbox"/> Peaceful <input type="checkbox"/> Energetic-Positive <input type="checkbox"/> Energetic-Negative <input type="checkbox"/> Calm-Positive <input type="checkbox"/> Calm-Negative </div> </div> <div style="display: flex; justify-content: space-between;"> <div style="display: flex; flex-direction: column; align-items: center;"> <div>Pleased Rating</div> <input style="width: 40px; height: 20px;" type="text"/> </div> <div style="display: flex; flex-direction: column; align-items: center;"> <div>Happy Rating</div> <input style="width: 40px; height: 20px;" type="text"/> </div> <div style="display: flex; flex-direction: column; align-items: center;"> <div>Excited Rating</div> <input style="width: 40px; height: 20px;" type="text"/> </div> <div style="display: flex; flex-direction: column; align-items: center;"> <div>Sad Rating</div> <input style="width: 40px; height: 20px;" type="text"/> </div> <div style="display: flex; flex-direction: column; align-items: center;"> <div>Bored Rating</div> <input style="width: 40px; height: 20px;" type="text"/> </div> <div style="display: flex; flex-direction: column; align-items: center;"> <div>Depressed Rating</div> <input style="width: 40px; height: 20px;" type="text"/> </div> <div style="display: flex; flex-direction: column; align-items: center;"> <div>Nervous Rating</div> <input style="width: 40px; height: 20px;" type="text"/> </div> <div style="display: flex; flex-direction: column; align-items: center;"> <div>Annoyed Rating</div> <input style="width: 40px; height: 20px;" type="text"/> </div> <div style="display: flex; flex-direction: column; align-items: center;"> <div>Angry Rating</div> <input style="width: 40px; height: 20px;" type="text"/> </div> <div style="display: flex; flex-direction: column; align-items: center;"> <div>Calm Rating</div> <input style="width: 40px; height: 20px;" type="text"/> </div> <div style="display: flex; flex-direction: column; align-items: center;"> <div>Relaxed Rating</div> <input style="width: 40px; height: 20px;" type="text"/> </div> <div style="display: flex; flex-direction: column; align-items: center;"> <div>Peaceful Rating</div> <input style="width: 40px; height: 20px;" type="text"/> </div> <div style="display: flex; flex-direction: column; align-items: center;"> <div>Energetic-Positive Rating</div> <input style="width: 40px; height: 20px;" type="text"/> </div> <div style="display: flex; flex-direction: column; align-items: center;"> <div>Energetic-Negative Rating</div> <input style="width: 40px; height: 20px;" type="text"/> </div> <div style="display: flex; flex-direction: column; align-items: center;"> <div>Calm-Positive Rating</div> <input style="width: 40px; height: 20px;" type="text"/> </div> <div style="display: flex; flex-direction: column; align-items: center;"> <div>Calm-Negative Rating</div> <input style="width: 40px; height: 20px;" type="text"/> </div> </div>
2		<div style="display: flex; justify-content: space-between; align-items: flex-start;"> <div style="display: flex; justify-content: space-around; width: 100%;"> <input type="checkbox"/> Pleased <input type="checkbox"/> Happy <input type="checkbox"/> Excited <input type="checkbox"/> Sad <input type="checkbox"/> Bored <input type="checkbox"/> Depressed <input type="checkbox"/> Nervous <input type="checkbox"/> Annoyed <input type="checkbox"/> Angry <input type="checkbox"/> Calm <input type="checkbox"/> Relaxed <input type="checkbox"/> Peaceful <input type="checkbox"/> Energetic-Positive <input type="checkbox"/> Energetic-Negative <input type="checkbox"/> Calm-Positive <input type="checkbox"/> Calm-Negative </div> </div> <div style="display: flex; justify-content: space-between;"> <div style="display: flex; flex-direction: column; align-items: center;"> <div>Pleased Rating</div> <input style="width: 40px; height: 20px;" type="text"/> </div> <div style="display: flex; flex-direction: column; align-items: center;"> <div>Happy Rating</div> <input style="width: 40px; height: 20px;" type="text"/> </div> <div style="display: flex; flex-direction: column; align-items: center;"> <div>Excited Rating</div> <input style="width: 40px; height: 20px;" type="text"/> </div> <div style="display: flex; flex-direction: column; align-items: center;"> <div>Sad Rating</div> <input style="width: 40px; height: 20px;" type="text"/> </div> <div style="display: flex; flex-direction: column; align-items: center;"> <div>Bored Rating</div> <input style="width: 40px; height: 20px;" type="text"/> </div> <div style="display: flex; flex-direction: column; align-items: center;"> <div>Depressed Rating</div> <input style="width: 40px; height: 20px;" type="text"/> </div> <div style="display: flex; flex-direction: column; align-items: center;"> <div>Nervous Rating</div> <input style="width: 40px; height: 20px;" type="text"/> </div> <div style="display: flex; flex-direction: column; align-items: center;"> <div>Annoyed Rating</div> <input style="width: 40px; height: 20px;" type="text"/> </div> <div style="display: flex; flex-direction: column; align-items: center;"> <div>Angry Rating</div> <input style="width: 40px; height: 20px;" type="text"/> </div> <div style="display: flex; flex-direction: column; align-items: center;"> <div>Calm Rating</div> <input style="width: 40px; height: 20px;" type="text"/> </div> <div style="display: flex; flex-direction: column; align-items: center;"> <div>Relaxed Rating</div> <input style="width: 40px; height: 20px;" type="text"/> </div> <div style="display: flex; flex-direction: column; align-items: center;"> <div>Peaceful Rating</div> <input style="width: 40px; height: 20px;" type="text"/> </div> <div style="display: flex; flex-direction: column; align-items: center;"> <div>Energetic-Positive Rating</div> <input style="width: 40px; height: 20px;" type="text"/> </div> <div style="display: flex; flex-direction: column; align-items: center;"> <div>Energetic-Negative Rating</div> <input style="width: 40px; height: 20px;" type="text"/> </div> <div style="display: flex; flex-direction: column; align-items: center;"> <div>Calm-Positive Rating</div> <input style="width: 40px; height: 20px;" type="text"/> </div> <div style="display: flex; flex-direction: column; align-items: center;"> <div>Calm-Negative Rating</div> <input style="width: 40px; height: 20px;" type="text"/> </div> </div>

Figure 6: The music emotion annotation section, also filled out every time the user goes through the questionnaire. The user clicks on a speaker to hear a music clip, then checks an emotion and supplies a rating 1 to 3

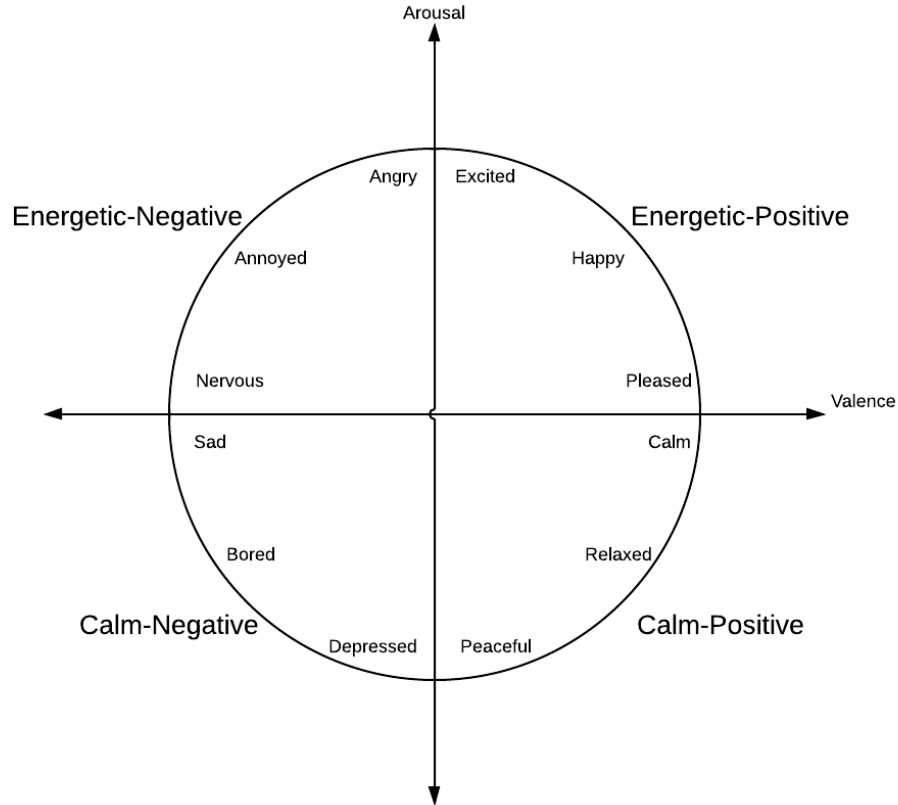


Figure 7: A diagram of the emotional model to be used for classifier clustering

3.1.2 Music Data

There are currently several feature extraction packages in common use in MIR research, such as MARSYAS [62], Psysound [7], and MIRToolbox [36]. These systems read audio files and extract various different features, depending on the perspective adopted. Psysound, for example, focuses more on psychoacoustic features [7] whereas MARSYAS and MIRToolbox focus more on broader MIR tasks [62] [36]. Because of this, MARSYAS and MIRToolbox are more commonly used in music emotion indexing tasks in particular such as in [53] and [58]. For this research, MIRToolbox is the package used.

Music data was collected from 100 audio clips split into 12-15 sequential 0.8 second segments (with a 0.2 second overlap between each segment) in order to allow for changes in features and labelling as the music progresses. These clips originated from several film and video game sound tracks in order to achieve a similar effect to the dataset composed in [15] (namely a set composed of songs that are less known and more emotionally evocative). The MIRToolbox [36] collection was then used to extract musical features. MIRToolbox is a set of functions developed for use in MATLAB which uses, among others, MATLAB’s Signal Processing toolbox. It reads .wav files at a sample rate of 44100 Hz. The following features were extracted using this toolbox.

3.1.2.1 Rhythmic Features

Rhythmic features refer to the set of audio features that describe a song’s rhythm and tempo, or how fast the song is.

Fluctuation Summary: The fluctuation summary is the calculation of rhythmic periodicity over the audio signal. First the estimated power spectrogram is calculated on frames of 23ms with a Bark-band decomposition (with an estimation of the masking effects), which results in a set of bands (i.e. amplitudes representing the energy distribution). The spectrogram is then modified by calculating a fast-Fourier transform (FFT) along each resulting band. The spectrogram is then added together across the bands, and the following features are obtained:

- Fluctuation Peak: Returns the peak of the audio signal’s fluctuation summary.
- Fluctuation Centroid: Returns the centroid of the audio signal’s fluctuation

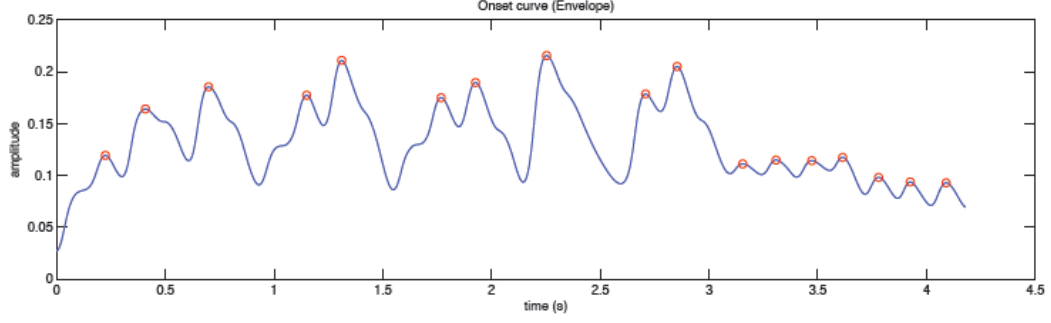


Figure 8: The onset curve extracted (via [36])

summary.

Tempo Estimation:

- Frame-based Tempo Estimation: An onset detection curve is calculated (Fig 8), which shows the rhythmic pulses in the song in the form of amplitude peaks for each frame. Tempo is determined by detecting peaks and when they occur.
- Autocorrelation: Detects tempo by computing an autocorrelation function of the onset detection curve. Given a signal x , and a lag of j , the autocorrelation function for time frame n is calculated as follows (via [36])

$$R_{xx}(j) = \sum_n x_n \bar{x}_{n-j} \quad (1)$$

Attack Properties: From the onset detection curve (shown in Figure 8), the following features can be extracted:

- Attack Time: The temporal duration of an onset from the beginning of an onset curve to the highest peak, referring to how quickly a beat occurred and indicating the rhythmic style of the piece.

- Attack Slope: The slope of an onset occurrence, based on the ratio between the magnitude distance during the attack period and the time difference.

$$S = \frac{amp(start) - amp(end)}{time_{start} - time_{end}} \quad (2)$$

3.1.2.2 Timbral Features

Timbral features describe a piece's sound quality, or the sonic texture of a piece of audio. The timbre of a song can change based on instrument composition as well as play style.

Spectral Features: The audio signal can be decomposed into an audio spectrum using the following equation (via [36]) for signal x and N samples (for $k = 0, \dots, N-1$).

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn} \quad (3)$$

The following features are calculated on the audio spectrum of a signal:

- Spectral Centroid: Returns the geometric center of the audio spectrum. This is found by calculating the first moment (mean) of the spectral distribution, as follows (via [36]):

$$\mu_1 = \int x f(x) dx \quad (4)$$

- Spectral Spread: The standard deviation of the audio spectrum, found by calculating the second central moment of the spectral distribution and taking the square root:

$$\sigma = \sqrt[2]{\mu_2} = \sqrt[2]{\int (x - \mu_1)^2 f(x) dx} \quad (5)$$

- Coefficient of Spectral Skewness: Refers to how symmetrical the spectrum is.

Found by first calculating the third central moment of the spectral distribution, as shown below:

$$\mu_3 = \int (x - \mu_1)^3 f(x) dx \quad (6)$$

The coefficient of skewness is then calculated using the cubed standard deviation:

$$\frac{\mu_3}{\sigma^3} \quad (7)$$

- Kurtosis: Describes the variance in peaks in the audio spectrum. This found by calculating the fourth standardized moment on the audio spectrum (with a subtraction of 3 applied in order to standardize the Kurtosis such that the kurtosis of a normal distribution is equal to 0):

$$\frac{\mu_4}{\sigma_4} - 3 \quad (8)$$

- Spectral Flux: Determines the distance between the spectra of each frame, found as follows for each frame i , where $X(k)$ is the spectrum of the given signal:

$$F_i = \sum_{k=1}^{\frac{N}{2}} (|X_i(k)| - |X_{i-1}(k)|)^2 \quad (9)$$

- Spectral Flatness: Indicates how smooth or spiky the spectrum is according to the ratio between the geometric mean and arithmetic mean, calculated as follows on the spectrum k of each frame i , given N as the total number of

partials in the spectrum:

$$\frac{\sqrt[N]{\prod_{i=0}^{N-1} X(k)}}{\left(\frac{\sum_{i=0}^{N-1} X(k)}{N} \right)} \quad (10)$$

- Irregularity: The degree of variation in the spectral peaks. This is calculated based on the approach proposed in [29] where a_k is the amplitude at partial k and N is the total number of partials in the signal:

$$\left(\sum_{k=1}^N (a_k - a_{k+1})^2 \right) / \sum_{k=1}^N a_k^2 \quad (11)$$

Mel-Frequency Cepstral Coefficients (MFCC) features: MFCC refers to a set of coefficients that describe the timbral shape of the sound using the Mel scale, which replicates how the human ear processes sound. They are calculated here by positioning the audio frequency bands logarithmically based on the Mel scale and calculating a Discrete Cosine Transform on them. Based on this calculation, we can retrieve:

- Delta-MFCC and Delta-Delta-MFCC: MFCC with temporal differentiations of the 1st order(at the 1st Mel-Frequency Cepstral Coefficient, called Delta-MFCC) and 2nd order(at the 2nd Mel-Frequency Cepstral Coefficient, called Delta-delta-MFCC).

Other Waveform features: The following features are found based on simple calculations on the audio signal

- Zero Crossings: Zero crossings refers to the number of times a waveform signal changes sign (crosses the x-axis). This is a commonly used feature used to indicate noisiness. The zero crossings are computed for each frame of the songs.

- Brightness: Returns the ratio of the amount of energy above a given frequency (in this case 1500 Hz) to the total amount of energy in the audio at each frame.

3.1.2.3 Tonal Features

Tonal features describe the tonal aspects of a song such as key, dissonance, and pitch.

Pitch: One of the more basic features extracted is the pitches present in the audio (in Hz). The audio is first decomposed into frames of .464ms with a hop length of 10ms. The pitches are then found by calculating the autocorrelation function on the audio waveform. This is the same function used to detect tempo, except applied on the audio waveform rather than the onset detection curve.

Chromagram: The pitch chromagram is also known as a Harmonic Class Profile and shows the distribution of energy across pitches. This can show a dominant pitch or set of pitches in the given audio, which can be used to determine key and tonality. The spectrogram is first calculated from the audio, and then the calculated energy is redistributed according to pitch rather than frequency. This redistribution can be done by converting the spectrum frequency in Hz to its corresponding pitch. For example, A4 is 440 Hz while C4 ("middle C" in musical terms) is 261.63 Hz. An example of a pitch chromagram can be seen in Figure 9.

Using this chromagram, the following features can be calculated:

- Pitch Chromagram Peak: The highest peak of the unwrapped pitch chromagram, in other words the pitch with the highest energy distribution. This can indicate dominant pitch

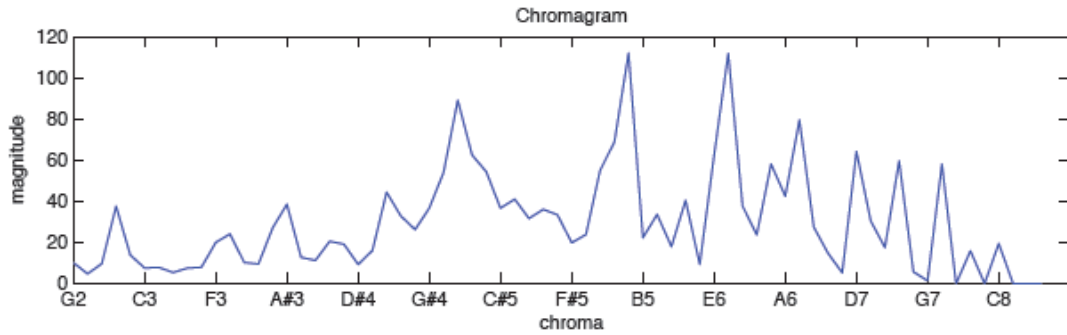


Figure 9: An example of a pitch chromagram, (via [36])

- Pitch Chromagram Centroid: The center of the pitch chromagram. This is found by calculating the first central moment, similar to finding the spectral centroid.

Key Strength Curve: In order to estimate key and mode, a key strength curve is first calculated based on a cross-correlation between the chromagram and possible tonality candidates (the possible keys and modes the piece could be in). Once this curve is calculated, the following features are found:

- Key Clarity: The key strength associated with the best key, namely the one with the highest strength. This is found by picking the peaks of the key strength curve (as calculated above) and finding the associated magnitude
- Mode: Estimates whether a piece is major or minor, based on a calculated value between -1 and 1. A positive value indicates a major key, while a negative value indicates a more minor key. This is found as follows:

$$\max(keyStrength) - \min(keyStrength) \quad (12)$$

The $\max(keyStrength)$ value represents the best major key candidate, while the $\min(keyStrength)$ value represents the best minor key candidate

- HCDF: The harmonic change detection function finds how often and how much the harmony changes in a song (the flux of the tonal centroid) based on the method proposed in [26]. This is calculated as follows (via [26]): First the tonal centroid vector ζ is found based on the chromagram on frame n , given d as one of the given dimensions (between 1 and 5), Φ as a transformation matrix based on the tonal space, and l as the chroma vector pitch class index (for indexes 0 through 11).

$$\zeta_n(d) = \frac{1}{\|c_n\|_1} \sum_{l=0}^{11} \Phi(d, l) c_n(l) \quad (13)$$

This results in a 6-dimensional tonal centroid vector. The HCDF is then found as follows:

$$\xi_n = \sqrt{\sum_{d=0}^5 [\zeta_{n+1}(d) - \zeta_{n-1}(d)]^2} \quad (14)$$

3.1.3 Data Compilation

The user would annotate 40 music pieces ($M_1, M_2, M_3, \dots, M_{40}$) in a single session. The user's answers to the questions from the demographic, general interests, and music questionnaire sections (described in Section 3.1.1 and shown in Figures 2, 3 and 4) are used as the identifier of a decision table. We assume that q1, q11, q12, q13, q2, q21, q22, q23, g3, q31, q32, q33, q4, q41, q42, and q43 are the mood questions corresponding to the words listed in the wheel in Figure 7 [q11=excited, q12=happy, q13=pleased, q1 is the generalization of q11,q12,q13, etc.]. This decision

table is then extended using the values of music features described in Section 3.1.2 (F_1, F_2, \dots, F_k where k is the total number of features, in the case of this work 340 features were calculated) found for each segment of each song. Recall that each of these songs are split into 12-15 overlapping segments of 0.8 seconds (with a 0.2 second overlap), so each song M_i has features (F_1, F_2, \dots, F_k) calculated for each segment $(M_{i1}, M_{i2}, \dots, M_{in})$ where n is equal to the total number of segments in that song. In addition, each table has one of the possible emotion annotations as a decision attribute, the values of which are based on the ratings the user gave for that emotion and that song. An example of an extended decision table, with annotations and features, for a user is pictured in Table 1.

For each user session there are 16 decision tables, one for each possible decision attribute (where each of the 16 emotion annotation columns is a possible decision attribute while the rest are removed). Each user can have up to 8 sessions resulting in 8 sets of annotations. The questionnaire answers, 8 groups of annotations, and 16 decision tables represent the total data for each user. This data is used to create the personalized classifiers for each user.

3.2 Developing Personalized Classifiers

Before cluster driven classification could be done, personalized classifiers had to be trained and tested using the classification algorithms listed previously (J48, SVM, Random Forest, IBk). The user annotation data was first converted so that each annotation for each song was represented as a vector of 16 numbers with each number representing the emotion labeling. The numbers ranged from 0 to 3, with 0 represent-

Table 1: A generalized version of the extended decision table

	F_1	F_2	F_k	q1	q11	q12	q13	q4	q41	q42	q43
$M_{1.1}$					1	2	1	0			2	2	3	0
$M_{1.2}$					1	2	1	0			2	2	3	0
$M_{1.3}$					1	2	1	0			2	2	3	0
.....														
.....														
$M_{1.15}$					1	2	1	0			2	2	3	0
$M_{2.1}$					1	0	0	2			0	0	0	0
$M_{2.2}$					1	0	0	2			0	0	0	0
$M_{2.3}$					1	0	0	2			0	0	0	0
.....														
.....														
$M_{2.15}$					1	0	0	2			0	0	0	0
$M_{3.1}$					2	2	0	2			2	1	2	2
$M_{3.2}$					2	2	0	2			2	1	2	2
$M_{3.3}$					2	2	0	2			2	1	2	2
$M_{3.15}$					2	2	0	2			2	1	2	2
.....														
.....														
$M_{40.1}$					2	2	2	0			1	1	0	0
$M_{40.2}$					2	2	2	0			1	1	0	0
$M_{40.3}$					2	2	2	0			1	1	0	0
.....														
.....														
$M_{40.15}$					2	2	2	0			1	1	0	0

ing an emotion that was not selected by the user and the remaining numbers being the strength the user entered with the annotation. These vectors for all the users were then linked with the feature data extracted from the corresponding music clips. From this resulting table all the annotations and music data linked with individual user IDs were used to train and test personalized classifiers for each emotion (depending on whether the user used a given emotion during the course of annotating). This resulted in each user having at most 16 personalized classifiers. The classifiers were all evaluated via Weka [23] using 10-fold cross validation. Analysis of these results indicates which classifier algorithm is most effective for personalized classification and, therefore, the most effective cluster-driven classifier.

3.2.1 Results

All four classifiers achieved a relatively high average accuracy, all above 80%. SVM achieved the lowest accuracy, 82.35%, while J48 trees achieved the highest accuracy, 86.62%. However, SVM as well as Random Forest achieved the highest average F-score (a combined measure of precision and recall). IBk on the other hand had the lowest F-score of 0.92. SVM was expected to have a higher accuracy since it works so well with music data, but our previous success with J48 means the high accuracies and F-scores are not surprising.

Looking at the average Kappa statistic reveals further insights into the effectiveness of each classifier. The Kappa statistic measures the agreement between a true class and the prediction, and the closer to 1 the statistic is the more agreement (1 represents complete agreement). None of the classifiers reaches higher than 0.15, although again

Table 2: Table of classifier accuracies and F-scores for Personalized Classifiers

Classifier	Average Accuracy	Average F-Score	Average Kappa
SVM	82.35%	0.90	0.137658
IBk	85.7%	0.87	0.153468
J48	86.62%	0.89	0.076869
Random Forest	84.25%	0.90	0.133027

Table 3: Comparison of classifier accuracies and F-scores between personalized and non-personalized classifiers

Classifier	Average Accuracy	Average F-Score	Average Kappa
Personalized J48	86.62%	0.89	0.076869
Non-personalized J48	87.29%	0.82	0.00015

IBk has the highest average Kappa (J48, again, the lowest). This all suggests that while J48 is overall very accurate it is more inconsistent in terms of this particular set of data, while SVM is moderately accurate and consistent, although none of the classifiers have any sort of high agreement. See Table 2

3.2.2 Comparison with Non-Personalized Classifiers

As a point of comparison, non-personalized classifiers were trained based on this dataset as well. Since it proved to be the most accurate classifier, J48 was chosen as the algorithm to use to build the non-personalized classifiers for comparison. Again 16 emotion classifiers were built, this time using all the user annotations to train and test rather than individual user annotations. The results compared to the personalized J48 classifiers are shown in Table 3.

The average accuracy does not change too much between personalized and non-personalized classifiers, however this was mainly due to the fact that several of the emotions were not used to the same extent as others when tagging, and in that case all the classifier did was predict '0' (for emotions that were not selected). This was

the case for the classifiers built on generalized emotions in particular, since those emotions would only have a rating in a very specific circumstance (i.e. the user selected a generalized emotion instead of any of the specific emotions). Future work would likely involve automatically assigning a corresponding value to generalized emotions. In the meantime while this raised the accuracy for those classifiers, it is not nearly as indicative as to the quality of the classifier as the F-Score and Kappa, which showed a great deal of improvement in the personalized classifier. The average F-score for the non-personalized classifiers is 0.07 less than the average F-score for personalized classifiers, and the average Kappa for the non-personalized classifiers is far less than the personalized classifiers. These both signify a significant loss in classifier consistency once the classifiers are no longer personalized.

CHAPTER 4: CLUSTER DRIVEN CLASSIFIERS

4.1 Cluster Algorithms and Distance Measures

Variations on agglomerative clustering were used as possible clustering methods. Agglomerative clustering begins with the dataset as individual instances (objects), then determines the distance either between two individual instances or between an instance and a cluster and merges the two objects with the closest distance into a cluster. The result is one large cluster with smaller sub-clusters within built based on the distances between objects.

To determine the best clustering for cluster-driven classifiers several inter-instance distance metrics (distances between individual data points) and inter-cluster distance metrics (distance between two clusters, or clusters and individual points) were combined to create clusters. Below is an explanation of each distance measure and how it is calculated

4.1.1 Inter-Instance Distance Measures

The distance between two items x and y , where x and y are two points in a dataset with n attributes (so $x = [x_1, \dots, x_n]$ and $y = [y_1, \dots, y_n]$).

- Manhattan Distance: One of the simpler ways to calculate distance, found by subtracting the values of x_i and y_i , calculating the absolute value of the result,

and adding the differences together.

$$ManhattanDist_{xy} = \sum_{i=1}^n |x_i - y_i| \quad (15)$$

- Euclidean Distance: This is the square root of the sum of the squared distance between x_i and y_i for every attribute i .

$$EuclideanDist_{xy} = \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (16)$$

- Maximum Distance: This is the maximum distance between the values of components x_i and y_i (based on their absolute values)

$$MaxDist_{xy} = \max(|x_i - y_i|) \quad (17)$$

- Canberra Distance: Used most often for data scattered around the origin, Canberra distance calculates distance by finding the sum of the fraction distances between items (the ratio between the difference of the values and their sum)

$$CanberraDist_{xy} = \sum_{i=1}^n \frac{|x_i - y_i|}{|x_i| + |y_i|} \quad (18)$$

- Pearson Distance: Based on the squared Pearson correlation coefficient, which is found as follows (given $C(x, y)$ as the covariance between x and y and $V(x)$ and $V(y)$ as the variances of each variable):

$$PCC_{xy} = \frac{(C(x, y))^2}{V(x)V(y)} \quad (19)$$

The distance is then found using the following equation:

$$PearsonDist_{xy} = 1 - PCC_{xy} \quad (20)$$

- Spearman's Distance: Spearman's distance is also correlation based, using Spearman's rank correlation coefficient instead of the Pearson coefficient. The distance, d_i is first calculated by taking the difference between x and y at index i . The Spearman rank correlation (SC_{xy}) is then calculated as shown below (given n as the number of values in x and y):

$$SC_{xy} = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)} \quad (21)$$

The distance is then found

$$SpearmanDist_{xy} = 1 - SC_{xy} \quad (22)$$

4.1.2 Cluster Formation

Given a distance matrix compiled using the distance measures listed in 4.1.1, clusters are formed using the Lance-Williams dissimilarity update formula to determine which clusters to merge at given points in the algorithm. This formula determines distance based on the following formula (where A is a cluster that holds instances x and y , and z is an unassigned instance in the dataset) [34]:

$$d_{Az} = \alpha_{A_x} dist(x, z) + \alpha_{A_y} dist(y, z) + \beta dist(xy) + \gamma |dist(x, z) - dist(y, z)| \quad (23)$$

The values of α , β and γ determine how and when clusters are merged together,

in that clusters with the smallest resulting d are merged as the clustering algorithm makes its way up the hierarchy. In their survey of hierarchical clustering algorithms, the authors in [49] list the corresponding parameter values for each of the methods listed below:

- Ward Minimum Variance: The purpose of Ward linkage is to form compact and spherical clusters by linking clusters based on finding the smallest minimum variance between clusters according to a given variance equation, although this method is sensitive to outliers.
- Single Linkage: Single linkage, also known as nearest neighbor clustering, is found by taking the minimum distance between any two objects in each cluster (or, in the case of a single object c and a cluster, the minimum distance between that object and any object in the cluster).
- Complete Linkage: Unlike single linkage complete linkage, or furthest neighbor clustering, takes the maximum distance between any two objects in the different clusters.
- Average Linkage: In this case, the distance between clusters is the average distance between all objects in both clusters.
- McQuitty: The distance here is also found by calculating the average distance between all objects in both clusters, except that the sizes of each cluster are added as weights. So for clusters A and B and instance c :

- Median: For this linkage, the median point of each cluster is found and the distance is determined to be the distance between each cluster's median point.
- Centroid: After finding the centroid of each cluster, the centroid being the average point in the space of a cluster or the "center", the distance between two clusters is the distance between their centroids.

4.2 Building Cluster-Driven Classifiers

4.2.1 Data Preprocessing

Before clustering could occur, the data from the first part of the questionnaire had to be preprocessed. First the numerical data (age, the amount of music training, and all of the mood scores) had to be standardized using the following measurement:

Given the mean absolute deviation s_f calculated below

$$s_f = (|x_{1f} - m_f| + |x_{2f} - m_f| + \dots + |x_{nf} - m_f|)/n \quad (24)$$

Where for each row in the raw questionnaire dataset (out of n rows), x is the value at a given row of attribute f and m is the average of attribute f 's values.

This value is then used to calculate the standardized measurement z_{if} for each value i of the attribute.

$$z_{if} = [x_{if} - m_f]/s_f \quad (25)$$

Afterwards, the remaining text-based data was converted to a set of numbers based on the data recorded for the user. Similar to the way the emotion annotations were

converted, each recorded attribute was converted to a vector of numbers representing what possible answer had (or had not) been recorded for each user. 0 represented possible answers that had not been recorded for the user, whereas 0.5 represented the answers that had been recorded for the user. 0.5 was selected as the numerical representation for selected answers for the sake of finding distance between two text attributes in a binary way, with the result of 1 being the distance between two users who selected different answers, and 0 between two users who had selected identical answers.

4.2.2 Tree-cutting methods

Once the hierarchical clustering is completed using the possible combinations of inter-instance and inter-cluster distance measures, the next step would be to "cut" the resulting dendrogram at a given level, resulting in a set of k clusters. Previous work on hierarchical clustering of music data tended to focus on instrument classification, such as in [31]. However the authors in that case had the benefit of having a k to use in cutting the resulting dendrogram (namely the number of instruments to be classified). Since we did not have a specific number of clusters we were looking for we needed a method to cut the resulting dendrogram that was organic to the data and the resulting clusters. For this purpose we used the Dynamic Tree Cut package in R, first introduced in [35] to work with bioinformatics data. This package provides methods that detect the clusters in each level of a dendrogram based on the shape of the resulting clusters at the base level of the dendrogram (which is often the result of an agglomerative clustering algorithm). This results in clusters that are organic

to the data, rather than relying on a specific height cutoff or a specific number of resulting clusters. There are two variants to the method in this package, both of which were tested in conjunction with the clustering distances described in Sections 4.1.1 and 4.1.2:

- **Dynamic Tree:** The dynamic tree variant of the dynamic tree cut algorithm analyzes the dendrogram resulting from hierarchical clustering to find clusters. The clusters from an initial cut of the dendrogram (usually a small number of large clusters) are analyzed for patterns in the height changes. This allows the algorithm to learn the structure of the tree, and split based on that structure. Clusters are then split based on the subclusters contained within. The algorithm stops once the number of clusters is stabilized.
- **Dynamic Hybrid:** The dynamic hybrid variant works from the bottom of the dendrogram upwards, clustering individual items based on certain base conditions (minimum size, distance from cluster, etc.) and iteratively joining branches.

4.2.3 Building Classifiers

Similar to the previous Section, classifiers were built for each emotion. However rather than create classifiers for each individual, classifiers were created for all the individuals in each cluster. Due to the hierarchical nature of agglomerative clustering, classifiers were also created for each node going up the dendrogram. This was achieved through the following process:

Merge Step: The result of the dynamic tree cut algorithms mentioned in Section 4.1.2 is a set of clusters dynamically created based on the clustering algorithm's resulting dendrogram. While this was something we wanted to utilize, it fails to preserve the hierarchical nature of the dendrogram. What is necessary for this system is not just a set of cluster assignments for each object, but cluster assignments for each object at each level of the resulting tree. Therefore, we first take the clusters resulting from the original dynamic tree cutting algorithm and create a new distance matrix based on the distance between the clusters individual centroids. The distance between the centroids is the same as the distance metric used to originally cluster the individual instances of the original data (as well as the cluster linkage), preserving the overall structure of the original dendrogram. Then starting from the bottom of the tree we travel up by iterating through the possible heights where subtrees are drawn and remerge the clusters in the subtree at each level based on this distance matrix. This results in a dendrogram that clusters the original individual clusters into larger clusters at each level.

Creating Classifiers based on Remerged Data: At each merge, the data is split based on the clusters and classifiers are trained and tested. Although the IBk algorithm has been proven to work the best with this data we train classifiers based on each of the four algorithms (SVM, J48, Random Forest, IBk). This allows for the possibility of employing a cascade classification strategy similar to [31].

At each level, the system selects the appropriate classifier to predict the next level down. In the case of [31], this meant the next instrument group. For this system this process is somewhat modified. Since the progressive levels of this tree are based

on the range of users covered (more general user information above to more specific user information below), and are therefore more dynamic in granularity, the resulting clustering structures have to be evaluated as a whole in addition to the classification algorithms for each level.

For all the clusters, the best average F-Score found for each classifier is saved in order to evaluate the clustering structure. This process of evaluation is explained in Section 4.2.5.

4.2.4 Elimination based on Cluster Validity Measures

Before evaluating the cluster driven classifiers, the clustered structures were first evaluated based on connectivity, silhouette width, and Dunn index. This was done for a couple of reasons: first of all, creating 64 classifiers (16 emotions * 4 classifier types) for every merge (the number of which range from 4 to over 200) for 61 total combinations of clustering measures, metrics, and cutting algorithms would not have been computationally feasible; second of all, if any of the clustering combinations resulted in poorly clustered data in the first place, it is very unlikely that the cluster driven classifier would have been effective and would likely even result in misclassification once new users were added. The measures used for initial evaluation of the clusterings are outlined below (implemented in the R package `clValid` [6], with the exception of silhouette width).

4.2.4.1 Connectivity

As outlined in [25] (using the implementation described in [6]), connectivity tests the extent to which instances are in the same clusters as their closest neighbors in

the rest of the dataset. Given a dataset with M instances and N attributes, for a given instance i in a cluster $nn_{i(k)}$ is the k th nearest neighbor to i . If i and $nn_{i(k)}$ are in the same cluster then $x_{i,nn_{i(k)}}$ is equal to zero, otherwise $x_{i,nn_{i(k)}}$ is equal to $1/k$. Finally, given L as the number of nearest neighbors to calculate, the connectivity for the set of clusters C is found as follows:

$$Connectivity(C) = \sum_{i=1}^M \sum_{k=1}^L x_{i,nn_{i(k)}} \quad (26)$$

4.2.4.2 Silhouette

Silhouette width was first proposed in [55] initially as a graphical validation measure. It has been adapted as a measure of clustering confidence with a range between -1 and 1 (with higher values indicating a well clustered instance).

Given a_i as the average dissimilarity (i.e. distance) between instance i and all other objects in its cluster, and b_i as the next nearest neighbor to instance i , the silhouette value is found by the following equation:

$$SilhouetteValue(i) = \frac{b_i - a_i}{\max(b_i, a_i)} \quad (27)$$

The average of these values is then taken for all instances in the dataset.

4.2.4.3 Dunn Index

The Dunn index was defined in [14] as the ratio between the minimum distance between instances in different clusters and the maximum distance between instances in the same cluster. Given the set of clusters C where $c_k, c_l, c_m \in C$ and instances i

and j as members of clusters c_k and c_l respectively the Dunn index is found as follows (as it is implemented in [6]):

$$Dunn(C) = \frac{\min_{c_k, c_l}(\min dist(i, j))}{\max diam(c_m)} \quad (28)$$

4.2.4.4 Result of Validity Measures

The highest values for these measures are considered more ideal, with the exception of connectivity which should be close to or at zero. With these requirements in mind, clusterings were eliminated as follows: first clusterings with a connectivity greater than zero were removed from consideration. From here, the interesting thing to observe is that there were no single inter-instance measures, inter-cluster measures, or branch cutting methods that resulted in an ideal Silhouette width or Dunn index. In fact, clusterings with the highest Dunn index had the lowest Silhouette width, and vice versa. Since it was not yet known which of these measures was more important for clustering this data in particular, in the interest of selecting measures with the best overall internal validity the clustering with the highest Dunn index was selected, as well as the clustering with highest Silhouette width. More general observations are below:

- Inter instance measures: Taking the average of each of these measures, the measures with the largest average connectivity were Canberra, Pearson, and Spearman. Spearman unfortunately also had the highest average Dunn index, and in general the clustering combinations with the highest connectivity also had the highest Silhouette and Dunn indexes, implying an opposition between

Table 4: Clusterings selected for classifier evaluation and corresponding silhouette and Dunn index (note: Connectivity has been omitted, since the connectivity for all selected clusterings is 0)

Inter Instance	Inter Cluster	Branch cutting	Silhouette	Dunn
Canberra	Single	Tree	0.2801	0.7092
Maximum	Average	Hybrid	0.7962	0.2313

cluster connectivity and Silhouette and Dunn. See Figure 10 for overall comparison

- Inter cluster measures: Ward had the lowest average connectivity, although the average connectivity of each of the inter cluster measures were low overall, with a much smaller range than the average connectivity of each of the inter instance measures. Ward also had a lower silhouette (by very little), and a very low Dunn index, again showing the opposition between connectivity and the other internal validity measures. Overall though, with the exception of a very high average connectivity for clusterings using the single cluster distance measure, the validity measures by inter cluster distance are overall very even. See Figure 11.
- Branch cutting measures: Clusterings with branches cut using the tree method had the lowest connectivity measure, while again also having the lowest Dunn index and silhouette. See Figure 12

Based on the criteria above, the clusterings selected for classifier testing and evaluation are listed in Table 4.

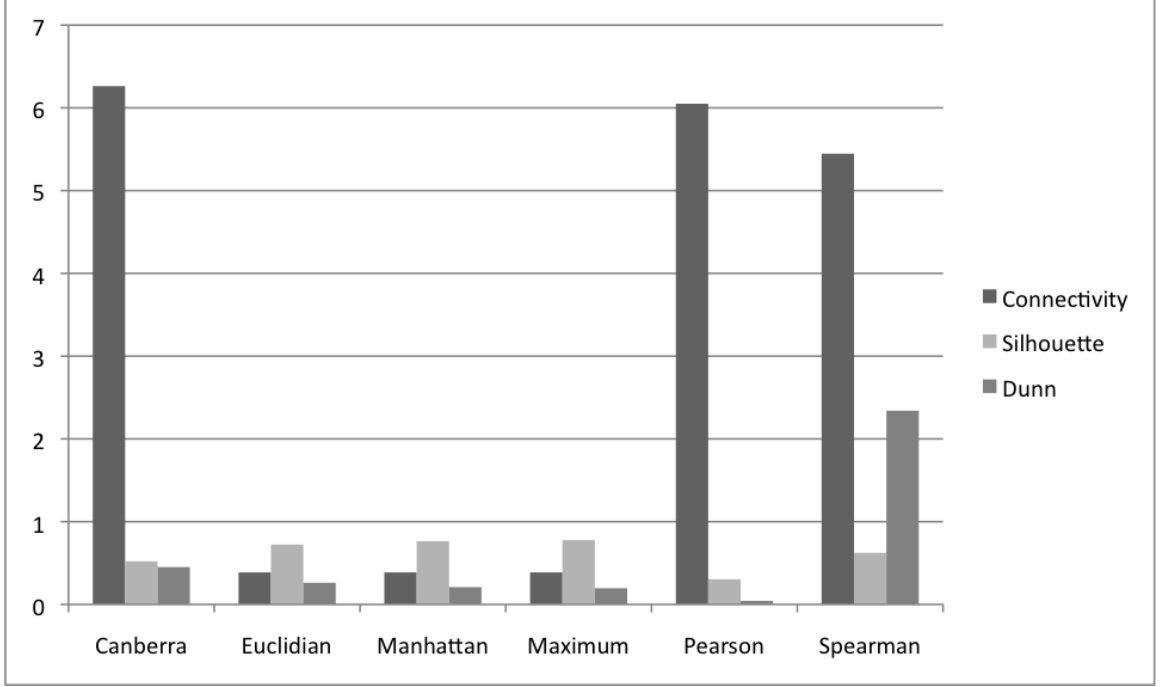


Figure 10: Average Cluster validity measure results, calculated by inter instance distance measures

4.2.5 Clustering Algorithm Evaluation

Once the classifiers were created, the next step is to use the classifiers to evaluate which combination of inter-instance measures and inter-cluster linking metrics resulted in clusters with the best potential classifiers. This was decided by looking at the F-Scores for the classifiers made at each merge in relation to the number of merges and the number of instances in each cluster. The following equation, which has been named *CCS* (Cluster-driven Classifier Score) was utilized to accomplish this:

$$CCS = (\sum_{i=1}^m (\sum_{j=1}^c iF_i * \frac{k_{ij}}{k_{all}})) / m \quad (29)$$

Where m is the total number of levels in the dendrogram, c_i is the number of clusters at level m_i , and k is the number of instances either in the given cluster c_{ij}

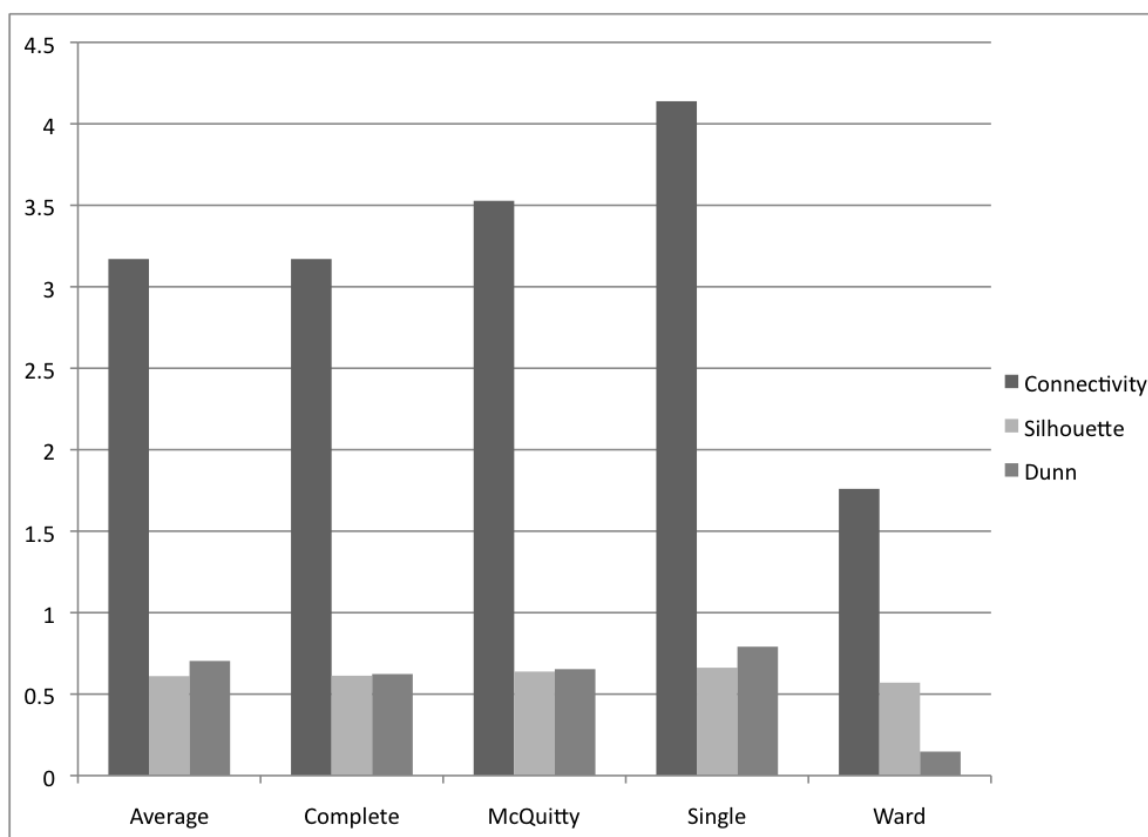


Figure 11: Average Cluster validity measure results, calculated by inter cluster distance measures

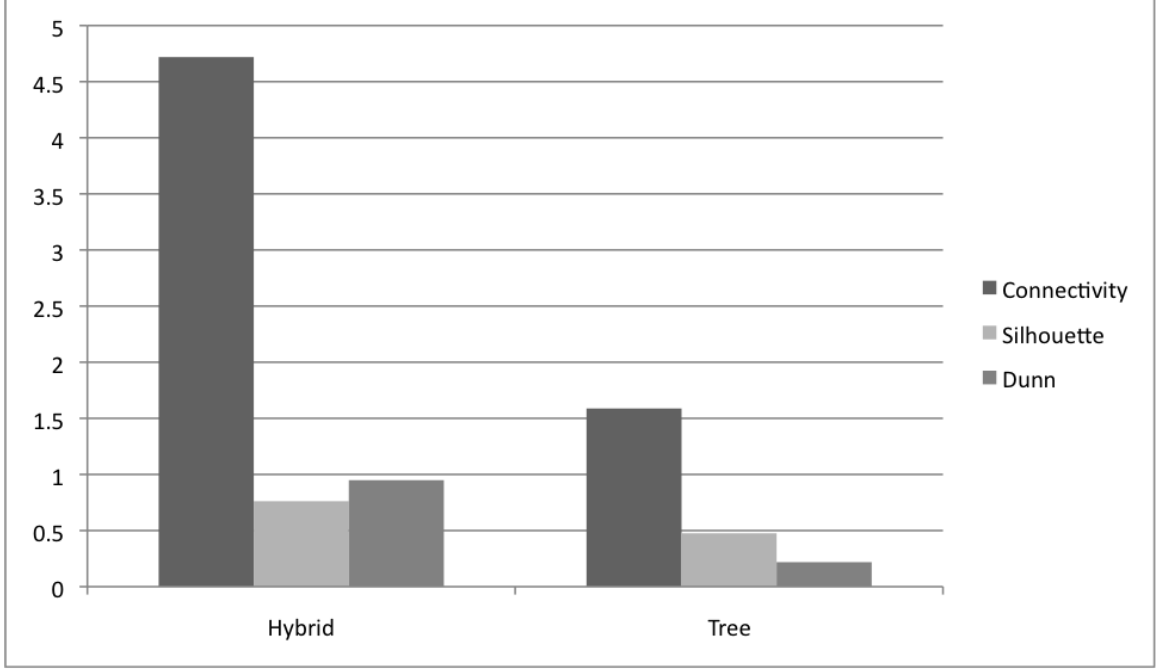


Figure 12: Average Cluster validity measures results, calculated by branch cutting method

(in the case of k_{ij}) or all the instances in the dataset (in the case of k_{all}). F_i in this case refers to the best F-Score of all the possible classifiers at merge m_i . So if for example at the given merge m_i the IBk algorithm yielded the highest F-Score, then F_i would refer to the F-Score of the IBk classifier. This allows us to find the best structure while normalizing the score based on the number of clusters and the number of instances.

4.2.6 Results

As shown in Table 4, the two cluster structures selected for evaluation were (in order of Inter cluster method-Inter instance method-Branch cutting method) the Single-Canberra-tree structure and the Average-Maximum-Hybrid structure. An interesting observation made as evaluation was going on was that the Single-Canberra-Tree struc-

ture only had one level (aside from the individual user level). The Average-Maximum-Hybrid structure, conversely, had 10. This would explain the larger Silhouette width for Single-Canberra-Tree, as the clusters in that structure would be very large.

Regarding the classifiers (the results of which are listed in Table 5), the best F-Scores seemed to come mostly from either Random Forest or SVM classifiers for each level, which mostly confirms the results in Section 3.2.1 about the consistency of these two classifier algorithms in regards to this dataset (recall that these algorithms both resulted in the highest F-Scores for fully personalized classification as well). It is interesting to observe for both structures that the F-Scores for both algorithms were generally higher than the F-scores for the fully personalized classifiers (recall that the average F-Scores for the personalized SVM and Random Forest classifiers were 0.90 for both algorithms), particularly in Average-Maximum-Hybrid levels 8-11 where the F-Scores start to get closer to 1. It is possible that having the input of other, closely related users increases the consistency of the classifiers and counteracts any improper tagging that may have occurred when gathering data in the first place. The F-Scores do dip somewhat at level 12 of the Average-Maximum-Hybrid structure (as well as Single-Canberra-Tree’s only highest level), suggesting a point of diminishing returns once the individual clusters become larger and the individual users within become more diverse.

The resulting *CCS* for each clustering structure evaluated is shown in Table 6. From this data, and from a cluster validity standpoint, it is clear that the Average-Maximum-Hybrid clustering structure is the best fit for this data. Considering that higher F-Scores were able to be achieved when creating classifiers for each level, this

Table 5: The best F-Scores, and the algorithm used, for each level of the clustering structures

Cluster Structure and Level	F-Score	Algorithm
Average-Maximum-Hybrid Level 1	0.9475493	Random Forest
Average-Maximum-Hybrid Level 2	0.9532502	Random Forest
Average-Maximum-Hybrid Level 3	0.9427221	Random Forest
Average-Maximum-Hybrid Level 4	0.9419194	SVM
Average-Maximum-Hybrid Level 5	0.9351064	Random Forest
Average-Maximum-Hybrid Level 6	0.9363794	Random Forest
Average-Maximum-Hybrid Level 7	0.9341183	SVM
Average-Maximum-Hybrid Level 8	0.9744212	SVM
Average-Maximum-Hybrid Level 9	0.9744212	SVM
Average-Maximum-Hybrid Level 10	0.9781771	SVM
Average-Maximum-Hybrid Level 11	0.9783481	SVM
Average-Maximum-Hybrid Level 12	0.9314592	Random Forest
Single-Canberra-Tree Level 1	0.9348288	SVM

Table 6: The clustering structures evaluated and the resulting *CCS*

Clustering Structure	<i>CCS</i>
Average-Maximum-Hybrid	0.816276564
Single-Canberra-Tree	0.467414403

is not unexpected. The Single-Canberra-Tree structure, conversely, was never able to achieve such a high F-Score (although its SVM classifiers were still able to achieve a higher F-Score on average than their fully personalized counterparts). As such, the Average-Maximum-Hybrid structure is the structure that will ultimately be used for the hierarchical recommendation system.

CHAPTER 5: HIERARCHY OF RECOMMENDER SYSTEMS

In the previous chapters we have found an optimized combination of classifier algorithms and clustering metrics for our music data. We can now build a hierarchy of individualized recommender systems based on this optimal set of algorithms, where the leaf nodes are individual users and each subtree is a grouping of those users. Each node of the hierarchy has its own set of 16 emotion classifiers, one for each possible emotion, based on the annotation data from all the users in each node. The lower the node on the tree the more specialized each of the cluster driven classifiers are, down to the leaves (which are individual users). This allows us to insert new users at the correct level and apply the classifiers associated with that cluster to music for the user. See Figure 13 for a visual representation of this structure.

5.1 Data Storage

Each user has two sets of data: a vector resulting from the questions the user answered in the first part of the questionnaire which is used to cluster the users in the first place (which will hereafter be referred to as D), the set of mood vectors built for each session based on the profile of mood states questions M , and a set of classifiers built on decision tables associated with each mood vector. See Figure 14.

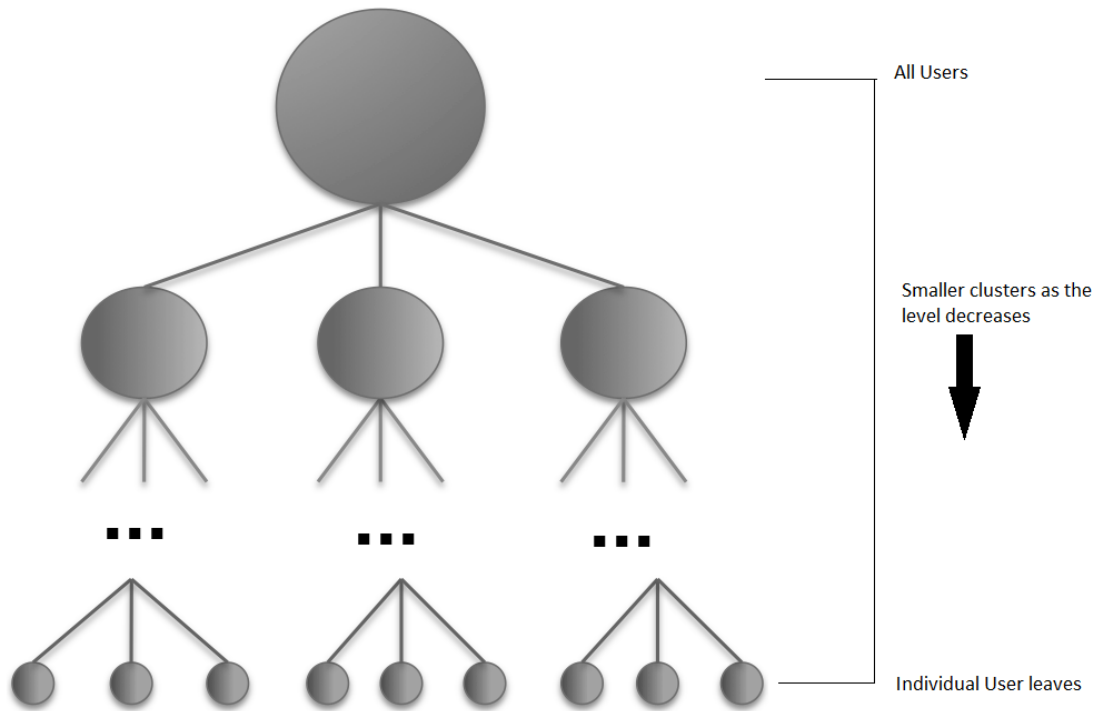


Figure 13: A visual of the proposed classifier hierarchy



Figure 14: A visualization of the data stored for a user a , where D is the set of answers the user submitted for the first part of the questionnaire, $M_{a1}-M_{an}$ is the set of mood vectors for that user, and C is the set of classifiers for each mood vector (M_{a1} , in the case of the above figure)

5.1.1 Mood Vector Creation

As explained in Section 3.1.1, for each session the user is asked to select an answer describing how much he/she has been feeling a selection of emotions. Once this is finished, his/her answers are then converted to a numerical mood vector as follows: each answer is given a score based on the response, with 0 representing "Not at all", 1 representing "A little", 2 representing "moderately", 3 representing "Quite a bit", and 4 representing "Extremely". From here, sets of mood scores corresponding to different emotions are added together into a set of scores:

$$TA = Tense + Shaky + Uneasy \quad (30)$$

Where TA stands for Tension/Anxiety

$$DD = Sad + Unworthy + Discouraged \quad (31)$$

Where DD stands for Depression/Dejection

$$AH = Angry + Grouchy + Annoyed \quad (32)$$

Where AH stands for Anger/Hostility

$$VA = Lively + Active + Energetic \quad (33)$$

Where VA stands for Vigor/Activity

$$FI = WornOut + Fatigued + Exhausted \quad (34)$$

Where FI stands for Fatigue/Inertia

$$CB = (Confused + Muddled) - Efficient \quad (35)$$

Where CB stands for Confusion/Bewilderment

These scores are recorded, along with a total score calculated as follows:

$$Total = (TA + DD + AH + FI + CB) - VA \quad (36)$$

The mood vector is then defined for user u and session s as

$$m[u, s] = (TA_{[u,s]}, DD_{[u,s]}, AH_{[u,s]}, FI_{[u,s]}, CB_{[u,s]}, VI_{[u,s]}, Total_{[u,s]}) \quad (37)$$

This vector is then linked to a set of decision tables for each emotion annotated by the user. For example, assume that we have three users ($u1, u2, u3$) with eight sessions. Each user builds 8 sets, each one containing 16 associated decision tables.

- Vector $m[u1,1]$ representing answers from user $u1$ in session 1 to the questionnaire.
- Vector $m[u2,1]$ representing answers from user $u2$ in session 1 to the questionnaire.
- Vector $m[u3,1]$ representing answers from user $u3$ in session 1 to the questionnaire.
- Vector $m[u1,2]$ representing answers from user $u1$ in session 2 to the questionnaire.

- Vector $m[u2,2]$ representing answers from user $u2$ in session 2 to the questionnaire.
- Vector $m[u3,2]$ representing answers from user $u3$ in session 2 to the questionnaire.

Therefore for user $u1$, M_{u1} is the set of mood vectors $m[u1,n]$ where n is the number of sessions:

$$M_{u1} = \{m[u1, n] : 1 \leq n \leq 8\} \quad (38)$$

Each of the vectors $m[ui,j]$, ($i=1,2,3$; $j=1,2,...,8$), is associated with a class containing 16 decision tables for each of the 16 possible moods. Assuming that vector $m[ui,j]$ is associated with a class $G[i,j]$ containing 16 decision tables, we get the following set of representative pairs $\{(m[ui,j], G[i,j]) : i = 1, 2, 3 \& j = 1, 2, 3, ..., 8\}$. These decision tables are used to train 16 classifiers for each emotion.

5.2 User Generalization

For every cluster at each level, we first create the smallest generalized description of all the users in that cluster. This way new users are assigned based on whether they fit in a given generalization. This does mean it is possible for a new user to be so unlike any other user that he/she is assigned to a node of the tree structure which is no longer close to its leaves.

Assume, for example, a cluster C with two users a and b . Cluster D_C is created as the smallest generalization of vectors D_a and D_b such that both of them are included in D_C . In other words, for each coordinate i in D_a and D_b :

$$D_{Ci} = \{k : \min(D_{ai}, D_{bi}) \leq k \leq \max(D_{ai}, D_{bi})\} \quad (39)$$

These larger clusters also have a set of mood vectors and decision tables. For instance, let us assume that cluster C contains $m[a1,8]$, $m[a2, 5]$, $m[b1, 2]$; C 's mood vector representation is defined as the smallest generalization of $m[a1,8]$, $m[a2, 5]$, $m[b1, 2]$ and it can be interpreted as a gate to a new set of 16 decision tables associated with them representing our sixteen different moods and constructed from semantically similar tables in $G[a1,8]$, $G[a2,5]$, and $G[b1,2]$. To be more precise, we take the union of 3 decision tables (one from each $G[a1,8]$, $G[a2,5]$, and $G[b1,2]$) with the same mood as their decision attribute, then build a classifier for this new unified decision table. This step has to be repeated for all 16 moods. See Figure 15.

5.3 New User Placement

When a new user, x , fills out the first part of the questionnaire his/her representative vector D_x is created. Then it is checked to see if this vector is equal to one of the representative vectors representing leaves of the tree structure. If this is not the case, then the representative sets of vectors belonging to the parents of these leaves are checked to see if one of them contains D_x . This is accomplished by comparing the individual values of each column i , D_{xi} , to each column range D_{Ci} . If every D_{xi} fits in the range of each D_{Ci} , then the user is assigned to that cluster. If D_x does not fit within any node on a given level, then it is compared to the parent of the cluster that D_x most closely matches (based on the number of column ranges in D_C that D_x does fit in). For example, if on the bottom level D_x cannot be assigned

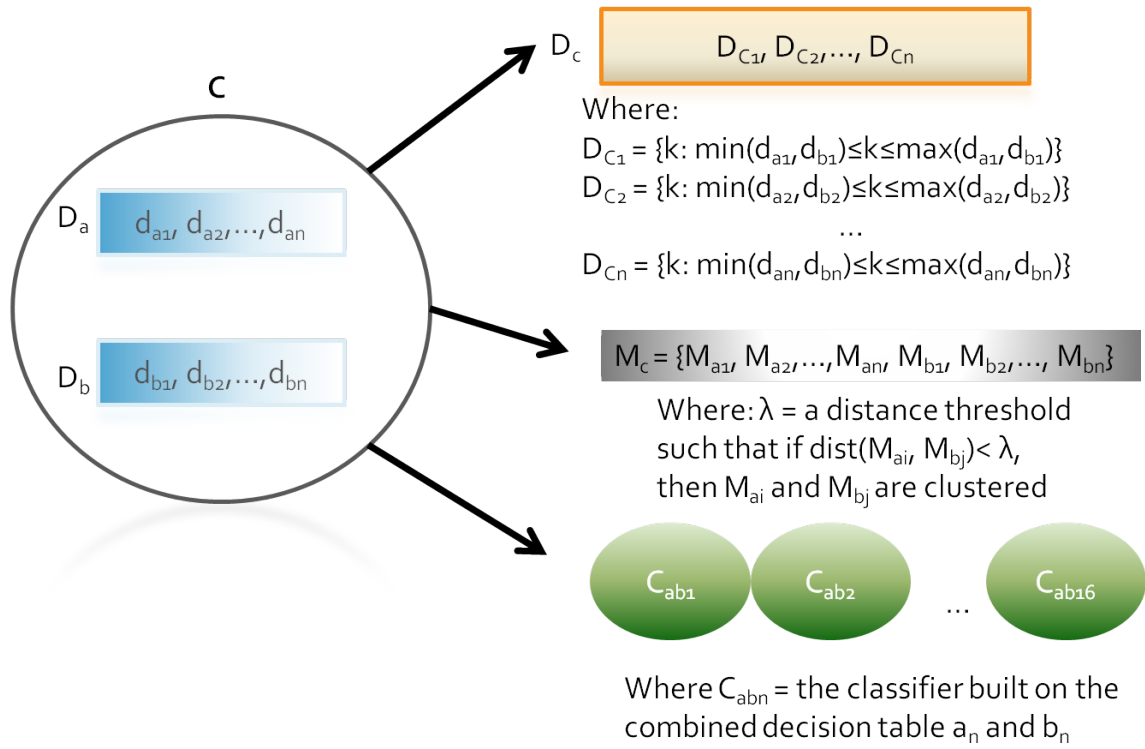


Figure 15: A visualization of the data stored for each cluster

to any cluster, but all i in D_{xi} fit in the ranges for D_{Ci} except for one, this would make it the closest matched cluster, and then D_x would be compared to the parent of D_C . This would occur for the first cluster found with the most matching values, future work would need to be done to determine an intelligent way to determine which cluster to compare next in the case of a tie. The goal is to assign the user to the cluster on the lowest level they can fit in because the classifiers built for the clusters on the lowest levels are more specialized, since they are built on data from a more homogenous group of users (and therefore a more unified group of annotations more than likely). These classifiers will therefore be more accurate to the new user, solving the "cold-start" issue inherent in collaborative recommender systems.

5.4 Evaluation

There are several methods for evaluating hierarchical classifiers for performance (a summary of method types can be found in [12]). However, since these hierarchical classifiers are meant to be personalized, it makes the most sense to test the classifier against the individual user's reaction. In other words, the user's opinion is the best test of accuracy in this situation. Therefore in order to evaluate this system and how well it addresses the "cold-start" problem, a dataset consisting of new users and new songs was used to test the classifiers. The users were assigned to a cluster using the process described in 5.3. The classifier for the user's assigned cluster was applied and predictions were made for each emotion. These predictions were compared to the user's original annotations to test the classifier's accuracy, precision, and recall. These measures were then averaged for each user, and these averages are used to judge

Table 7: Accuracies, F-Scores, and Kappa Statistics for the fully personalized and hierarchical cluster driven classifiers

Classifier	Average Accuracy	Average F-Score	Average Kappa
Personalized SVM	82.35%	0.90	0.137658
Personalized IBk	85.7%	0.87	0.153468
Personalized J48	86.62%	0.89	0.076869
Personalized Random Forest	84.25%	0.90	0.133027
Hierarchical Classifier	88.5%	0.34	0.519426874

how well the assigned classifiers perform with new users. The higher the accuracy, precision, and recall measures of the system, the better the system addresses the "cold-start" problem.

The test set consisted of 56 unique users, each of whom completed an equivalent of one session (meaning in this case there was only one mood vector per user). As in previous tests, a classifier was created for each user and each emotion, resulting in 480 total classifiers. These users annotated a separate set of 16 songs, ranging in both tonality (major/minor) and genre (classical, rock, jazz, country).

5.4.1 Overall Results

Table 7 shows a comparison between the fully personalized classifiers developed in chapter 3 and the hierarchical clusters assigned to completely new users as described in the preceding Section. Overall, these results are very promising. Not only did it achieve a higher accuracy than any of the classifiers built on datasets of user annotations alone, but the Kappa statistic is much higher than any of those classifiers were able to achieve, suggesting a higher agreement between the true classes and the predictions. This means the classifiers assigned to a first time user of this system would already be much more accurate than classifiers developed by users who had

annotated a set of songs. The one concerning thing about these results is the lowered F-score for these classifiers. This would imply that these classifiers may have lower precision and/or recall than the fully personalized classifiers. Whether this is something that can be improved through continued use of the system, perhaps by retraining the classifier in the case of inaccurate predictions, or by simply recording additional user annotations is currently unknown. However, the high accuracy and kappa statistics alone suggest that the developed classifiers are very effective in a "cold start" situation.

5.4.2 Individual User Test Case

To determine how accurate these observations are at the individual level, below is the analysis of a new user using this system for the first time. This user has filled out the same questionnaire questions, however they only have one mood vector which (along with their other questionnaire answers) is used to assign this user to a cluster. This user also annotated a different set of songs, which had the same feature data extracted from them as the initial training set.

Once this user is assigned to a cluster, he/she is given the set of emotion classifiers made for the cluster in that part of the tree. The user's new annotations were then used to evaluate the classifiers' effectiveness. The resulting statistics are shown in Table 8.

The accuracy and F-Scores are able to stay relatively high for each classifier, although it is not consistent. This could be explained by the user not annotating songs with certain emotions, which would make the accuracies very difficult to judge in this

Table 8: Statistics for the classifiers built for a new user

Emotion	Accuracy	Avg. Precision	Avg. Recall	Avg. F-Score
Pleased	87.5	0.875	0.875	0.87
Happy	50	0.493	0.500	0.493
Excited	70.8333	0.675	0.708	0.691
Sad	100	1.000	1.000	1.000
Bored	91.6667	0.917	0.917	0.917
Depressed	95.8333	0.918	0.958	0.938
Nervous	79.1667	0.756	0.792	0.773
Annoyed	91.6667	0.917	0.917	0.917
Angry	100	1.000	1.000	1.000
Calm	83.3333	0.761	0.833	0.795
Relaxed	75	0.714	0.750	0.729
Peaceful	62.5	0.594	0.625	0.609
Energetic-Positive	87.5	0.911	0.875	0.892
Energetic-Negative	87.5	0.875	0.875	0.875
Calm-Positive	54.1667	0.574	0.542	0.557
Calm-Negative	87.5	0.915	0.875	0.894

case. This could indicate also that certain emotions are easier to classify for songs than others. Observe that the lower accuracies (aside from Happy) are for emotion classifiers from the calm-positive quadrant of the arousal-valence plane (Calm, Relaxed, Peaceful), and the least accurate quadrant classifier (Energetic-Positive, Energetic-Negative, Calm-Positive, Calm-Negative) is the Calm-Positive classifier. This could imply that emotions with a high valence and low arousal are particularly difficult to detect in music, although this would require further investigation (and likely a value automatically assigned to generalized emotions based on the selection of a specific emotion).

5.4.3 Comparison of All Classifiers

What follows is a comparison between the hierarchical classifiers, the personalized classifiers described in chapter 3, and the cluster driven classifiers built in chapter 4. A full table of the average accuracies, f-scores, and kappa statistics can be seen in Table 9. Graphical representations can be seen in Figures 16 and 17.

The most accurate classifiers were the cluster based classifiers toward the top of the hierarchy, although the accuracy drops sharply at level 12 (likely due to the size and diversity of the data at that level). It seems there is an ideal cluster size to use as classification data for users; clusters that are too large result in data that is too diverse to build accurate classifiers, clusters that are too small will yield classifiers that are only accurate for a limited set of music for a similarly limited set of users. It is likely, therefore, that the hierarchical clusters manage to assign users to their ideal point in the cluster, where the data used to train the cluster is similar enough to the

Table 9: Accuracies, F-Scores, and Kappa Statistics for the fully personalized and hierarchical cluster driven classifiers

Classifier	Mean Accuracy	Mean F-Score	Mean Kappa
Personalized SVM	82.35	0.90	0.137658
Personalized IBk	85.7	0.87	0.153468
Personalized J48	86.62	0.89	0.076869
Personalized Random Forest	84.25	0.90	0.133027
Average-Maximum-Hybrid Level 1	86.82	0.95	0.097877
Average-Maximum-Hybrid Level 2	88.82	0.95	0.12894
Average-Maximum-Hybrid Level 3	86.90	0.94	0.10496
Average-Maximum-Hybrid Level 4	87.71	0.94	0.070993
Average-Maximum-Hybrid Level 5	85.86	0.94	0.003190
Average-Maximum-Hybrid Level 6	86.17	0.94	0.002089
Average-Maximum-Hybrid Level 7	86.03	0.93	0.008013
Average-Maximum-Hybrid Level 8	91.49	0.97	-0.000803
Average-Maximum-Hybrid Level 9	91.50	0.97	-0.000803
Average-Maximum-Hybrid Level 10	92.16	0.98	-0.001110
Average-Maximum-Hybrid Level 11	93.18	0.98	0.000955
Average-Maximum-Hybrid Level 12	81.91	0.93	-0.007563
Hierarchical Classifier	88.5	0.34	0.519426874

user to be relevant, but diverse enough to cover emotion and musical types that the user has not encountered.

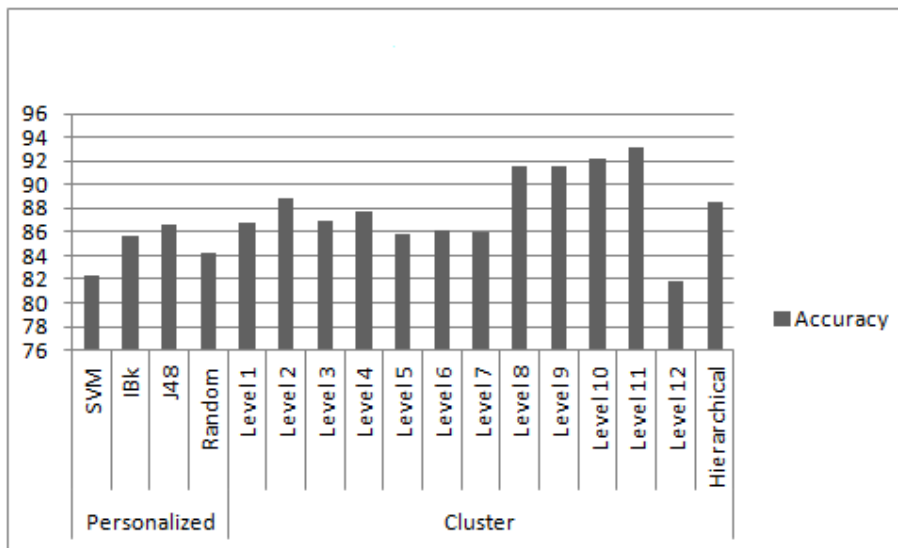


Figure 16: The average accuracies for the personalized, cluster based, and hierarchical classifiers

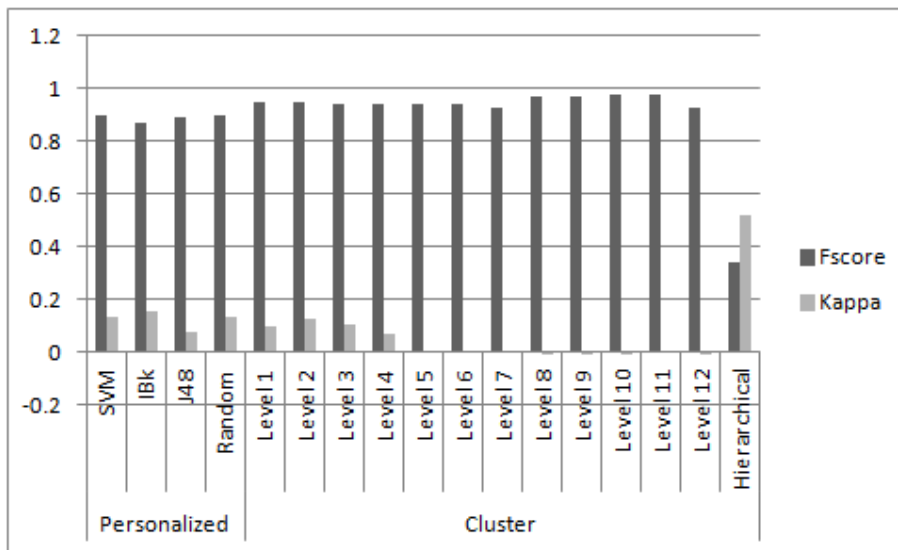


Figure 17: The F-Scores and Kappa statistics for the personalized, cluster based, and hierarchical classifiers

CHAPTER 6: CONCLUSION AND FUTURE WORK

6.1 Future Work

In examining the results a couple general observations can be drawn about the overall effectiveness of the system, and there are clearly some areas that can be improved. First of all, one of the unforeseen consequences of using a straightforward hierarchical scheme and later reconstituting it is that some of the clusters are of a very uneven size. While at the upper levels this is not as much of a concern, at the lower levels larger clusters mean a less personalized cluster. Additionally, users could be assigned to clusters they do not entirely match with despite fitting in the range of the generalized vector (since the vector would be too generalized). Clearly a little more control over the cluster formation is necessary, whether in size or otherwise. User assignment could also be based more on distance from the cluster centroid or calculating the distance (in the way that was done during cluster formation) and allowing for a set distance tolerance rather than using the generalized vector. It would stand to reason in this case that the tolerance would have to be higher in the upper levels of the tree.

In regards to testing, in this case the focus was on hierarchical classifier effectiveness in terms of the "cold start" problem. In this case that meant each user had only one session and, therefore, one mood vector. While sufficient for testing the "cold start"

issue for new users, future evaluation would involve starting a new user in the system and having them use it continually, building new mood vectors and in turn testing how the system reacts and learns the user's musical and emotional preferences.

6.1.1 System Improvements

6.1.2 Flexible Query-Answering System

Future work with this system will also involve a flexible query answering system, which will be built to handle user queries for music of different moods. The user will be able to ask for songs which fit a set of emotions. However, there are songs where the query may not fit, so the query needs to be able to be changed or extended in order to find songs. For example, a user can ask for a song that makes them feel happy and calm. In which case, the classifiers for "Happy" and "Calm" at the node the user is assigned to are used to find which songs are happy and calm. However, if there is no song that can be classified as both happy and calm, then the query is extended using the emotion generalizations. "Happy" can be extended to "Energetic-Positive" while "Calm" can be extended to "Calm-Positive".

6.1.3 Overview of Query Relaxation Strategies

There are several approaches to flexible query answering and handling failing queries. The typical approach is to relax the query constraints when a result is not returned. What is returned instead is a result (or set of results) that are close to what the user asked for, if not exactly what their query specified. A simple example would be querying a table of customer information by a specific age and having the query changed to a range of ages if there are no customers of that specific age in the

table (if a user is querying a database for customers that are 26 years old and there are none, the query may be changed to search for customers between the ages of 25 and 30).

Queries cannot be relaxed arbitrarily however. While a result that fits an exact query may not be able to be returned, the goal of a flexible query answering system is still to return a result that is as close as possible given the constraints on the table and the data present. In order to return the closest results, the system must consider the database structure and the existing relationships between attributes that can be found from the current data. Muslea and Park proposed an algorithm in [50] known as LOQR, which modifies a query based on decision rules found in the database. Based on a randomly selected subset of data, rules describing the implicit relationship between the data attributes related to the query were found. The query was then modified based on the rule which was found to be closest to the user's original query. This ensures that the new query submitted to the system returns the closest result that fulfils the user's original query based on the data itself.

Assume the case of music data annotated with n emotions e_1, e_2, \dots, e_n with k music features F_1, F_2, \dots, F_k . Given a failing query $Q = e_1 \wedge e_5$ this algorithm would be implemented by first finding rules that would satisfy the emotions listed in the query based on the extracted musical features. These rules would consist of two parts; the first part is a set of conditions expressing the values (or range of values) of a set of r features; the second part is the condition of a given query attribute based on the condition of the first part, such as $emotion == e_1$. So the overall format of the rules would be:

$$F_1 = [val1, val2] \wedge F_2 = [val3, val4] \wedge \dots \wedge F_r = [val_x, val_y] \implies emotion = e_x \quad (40)$$

This could be read as *"If Feature 1 is between value1 and value 2, and Feature 2 is between value3 and value4, and (...), and Feature r is between value x and value y, then the emotion is e_x .* This set of rules would then be parsed to find which rules are closest to the original query (in this example, it would be the rules that satisfied the emotion conditions listed in the query, so for query Q the most useful rules would be the rules that implied e_1 or e_5). Finally, the original query would be changed. Based on the most useful rule determined by the algorithm, the resulting emotion condition would be replaced by the feature conditions in the first part of the rule. So given the original query Q and assuming the most useful rule is determined to be $F_1 = [value1, value2] \wedge F_6 = [value3, value4] \implies e_5$, the new query would be:

$$Q_{new} = e_1 \wedge (F_1 = [value1, value2] \wedge F_6 = [value3, value4]) \quad (41)$$

While the LOQR algorithm adds new attributes to the query based on decision rules, there are also systems where the constraints of the query are simply generalized rather than altered. In a system where the data is organized in a hierarchical system, the hierarchical relationship between attributes can be utilized to relax a failing query as well. Chu and Chen referred to this kind of hierarchy as a type abstraction hierarchy [10], where higher levels of the hierarchy are viewed as abstractions of their lower level children and where queries submitted to this hierarchy can

be relaxed by replacing one part of a failing query with its higher level abstraction. So if given a system where $a[1,1]$ and $a[1,2]$ are subclasses of $a[1]$, then a query looking for $a[1,1]$ or $a[1,2]$ can be generalized by replacing either term with $a[1]$.

6.1.4 Application to Music Recommender

As previously noted, the emotion modeling system used here is also hierarchical in nature, where an emotion $e11$ can be written as $Emotion[e11, e1]$. This means it can be applied as an abstraction hierarchy for query relaxation. For example, take the class $Emotion[Excited-Positive]$ and the subclasses $Emotion[Pleased, Excited-Positive]$, $Emotion[Happy, Excited-Positive]$, and $Emotion[Excited, Excited-Positive]$. A query Q for song S that is Pleased and Peaceful can be presented as

$$Emotion[Pleased, Excited - Positive] \wedge Emotion[Peaceful, Calm - Positive] \quad (42)$$

If that query fails, it can be relaxed to the emotion's given abstraction, in this case $Emotion[Excited-Positive]$ and $Emotion[Calm-Positive]$ respectively. So the original query can be changed to

$$Emotion[Excited - Positive] \wedge Emotion[Peaceful, Calm - Positive] \quad (43)$$

or

$$Emotion[Pleased, Excited - Positive] \wedge Emotion[Calm - Positive] \quad (44)$$

6.2 Conclusion

This dissertation presents a hierarchical recommendation system that is both collaborative and content-based. It presents an accurate set of music recommendations

for a new user based on his/her mood from the outset, solving the cold-start problem inherent in more collaborative systems. This is based on the optimal combination of cluster-driven classifiers, instance distances, and cluster linkages. The system has been evaluated to see how well it combats the cold-start problem, and has been proven to be very effective at this task. Classifiers assigned to new users are about as consistent, and more accurate, than classifiers built from individual user data (and about as accurate as the best cluster-based classifiers). This system can be incorporated into personal music players to recommend music based on mood, used for music therapy, and eventually even incorporate classifier retraining to make the classifiers even more personalized to the user. Finally, this system will also serve as a significant contribution to the field of music information retrieval and music recommender systems since it directly addresses two significant research problems: music mood retrieval and the cold-start problem.

REFERENCES

- [1] D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66, 1991.
- [2] E. M. Albornoz, D. H. Milone, and H. L. Rufiner. Spoken emotion recognition using hierarchical classifiers. *Computer Speech Language*, 25(3):556–570, 2011.
- [3] M. Alghoniemy and A. H. Tewfik. Personalized music distribution. *2000 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings Cat No00CH37100*, pages 2433–2436, 2000.
- [4] D. Bogdanov, M. Haro, F. Fuhrmann, A. Xambó, E. Gómez, and P. Herrera. Semantic audio content-based music recommendation and visualization based on user preference examples. *Information Processing & Management*, 49(1):13–33, Jan. 2013.
- [5] L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001.
- [6] G. Brock, V. Pihur, S. Datta, and S. Datta. clValid: An R Package for Cluster Validation. *Journal of Statistical Software*, 25(4), 2008.
- [7] D. Cabrera and Others. PsySound : A Computer Program for Psychoacoustical Analysis. In *Australian Acoustical Society Conference*, volume 24, pages 47–54, 1999.
- [8] D. Castán, A. Ortega, and E. Lleida. Speech/Music classification by using the C4.5 decision tree algorithm. In *FALA 2010*, number M, pages 197–200, 2010.
- [9] H.-C. Chen and A. L. P. Chen. A Music Recommendation System Based on Music and User Grouping. *Journal of Intelligent Information Systems*, 24(2-3):113–132, Mar. 2005.
- [10] W. W. Chu and Q. C. Q. Chen. A structured approach for cooperative query answering. *IEEE Transactions on Knowledge and Data Engineering*, 6(5):738–749, 1994.
- [11] W. W. Cohen and W. Fan. Web-collaborative filtering: recommending music by crawling the Web. *Computer Networks*, 33(1-6):685–698, June 2000.
- [12] E. P. Costa, C. Postal, A. C. Lorena, and R. S. Ad. A Review of Performance Evaluation Measures for Hierarchical Classifiers. In *Evaluation Methods for Machine Learning II: papers from the AAAI-2007 Workshop*, pages 1–6. AAAI, 2007.
- [13] L. Cyril. Estimating tonal tension from audio content. *Frontiers in Human Neuroscience*, 3, 2009.

- [14] J. Dunn. Well-separated clusters and optimal fuzzy partitions. *Journal Of Cybernetics*, 4(1):95–104, 1974.
- [15] T. Eerola and J. K. Vuoskoski. A comparison of the discrete and dimensional models of emotion in music. *Psychology of Music*, 39(1):18–49, Aug. 2011.
- [16] P. Ekman. An argument for basic emotions. *Cognition & Emotion*, 6(3):169–200, May 1992.
- [17] P. Ekman. Are there basic emotions? *Psychological review*, 99(3):550–553, 1992.
- [18] Y. Feng, Y. Zhuang, and Y. Pan. Music information retrieval by detecting mood via computational media aesthetics. pages 235–241, 2003.
- [19] A. A. Freitas and A. de Carvalho. A Tutorial on Hierarchical Classification with Applications in Bioinformatics. In D. Taniar, editor, *Research and Trends in Data Mining Technologies and Applications*, number ML, pages 175–208. Idea Group, 2007.
- [20] A. Friberg. Digital Audio Emotions - An Overview Of Computer Analysis And Synthesis Of Emotional Expression In Music. *Proc. Int. Conf. on Digital Audio Effects*, 2008.
- [21] J. Grekow and Z. Ras. Emotion Based MIDI Files Retrieval System. *Advances in Music Information Retrieval*, 274:261–284, 2010.
- [22] J. Grekow and Z. W. Raś. Detecting Emotions in Classical Music from MIDI Files. In J. Rauch, editor, *Foundations of Intelligent Systems, Proceedings of ISMIS’09*, pages 261–270, Prague, Czech Republic, 2009. Springer.
- [23] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software. *ACM SIGKDD Explorations Newsletter*, 11(1):10, Nov. 2009.
- [24] S. Han, B. Rho, R. Dannenberg, and E. Hwang. SMERS: Music Emotion Recognition using Support Vector Regression. In *Proceedings of the 10th International Society for Music Information Retrieval Conference*, pages 651–656, Kobe, Japan, Oct. 2009. International Society for Music Information Retrieval.
- [25] J. Handl, J. Knowles, and D. B. Kell. Computational cluster validation in post-genomic data analysis. *Bioinformatics (Oxford, England)*, 21(15):3201–12, Aug. 2005.
- [26] C. Harte and M. Sandler. Detecting harmonic change in musical audio. *Proceedings of the 1st ACM workshop on Audio and music computing multimedia AMCMM 06 (2006)*, C(06):21, 2006.
- [27] P. G. Hunter, E. G. Schellenberg, and U. Schimmack. Mixed affective responses to music with conflicting cues. *Cognition & Emotion*, 2(22):327–352, 2008.

- [28] P. G. Hunter, E. G. Schellenberg, and U. Schimmack. Feelings and perceptions of happiness and sadness induced by music: Similarities, differences, and mixed emotions. *Psychology of Aesthetics, Creativity, and the Arts*, 4(1):47–56, 2010.
- [29] K. Jensen. *Timbre models of musical sounds*. 1999.
- [30] W. Jiang, A. Wieczorkowska, and Z. W. Ras. Music Instrument Estimation in Polyphonic Sound Based on Short-Term Spectrum Match. *Foundations of Computational Intelligence Volume 2*, pages 259–273, 2009.
- [31] W. Jiang, X. Zhang, A. Cohen, and Z. Raś. Multiple classifiers for different features in timbre estimation. *Advances in Intelligent Information Systems*, pages 335–356, 2010.
- [32] M. Kursa, W. Rudnicki, A. Wieczorkowska, E. Kubera, and A. Kubik-Komar. Musical Instruments in Random Forest. *Foundations of Intelligent Systems*, 5722:281–290, 2009.
- [33] P. Lamere. Social Tagging and Music Information Retrieval. *Journal of New Music Research*, 37(2):101–114, June 2008.
- [34] G. N. Lance and W. T. Williams. A General Theory of Classificatory Sorting Strategies: 1. Hierarchical Systems. *The Computer Journal*, 9(4):373–380, Feb. 1967.
- [35] P. Langfelder, B. Zhang, and S. Horvath. Defining clusters from a hierarchical cluster tree: the Dynamic Tree Cut package for R. *Bioinformatics (Oxford, England)*, 24(5):719–20, Mar. 2008.
- [36] O. Lartillot, P. Toivainen, and T. Eerola. *MIRtoolbox*. University of Jyväskylä, 2008.
- [37] C. Laurier, O. Lartillot, T. Eerola, and P. Toivainen. Exploring Relationships between Audio Features and Emotion in Music. In *ESCOM, Conference of European Society for the Cognitive Sciences of Music*, Jyväskylä, Finland, 2009.
- [38] C. Laurier, O. Meyers, R. Marxer, D. Bogdanov, J. Serrà, E. Gómez, P. Herrera, and N. Wack. Music classification using high-level models. *ISMIR 2009*, 2010.
- [39] R. Lewis, A. Cohen, W. Jiang, and Z. Raś. Hierarchical tree for dissemination of polyphonic noise. In *Rough Sets and Current Trends in Computing*, pages 448–456. Springer, Springer, 2008.
- [40] R. Lewis, X. Zhang, and Z. Ras. A knowledge discovery model of identifying musical pitches and instrumentations in polyphonic sounds. *Special Issue, International Journal of Engineering Applications of Artificial Intelligence*, 2007.
- [41] N. H. Liu and S. J. Hsieh. Intelligent Music Playlist Recommendation Based on User Daily Behavior and Music Content. *Advances in Multimedia Information ProcessingPCM 2009*, pages 671–683, 2009.

- [42] P. Lops, M. de Gemmis, and G. Semeraro. Content-based Recommender Systems: State of the Art and Trends. In *Recommender Systems Handbook*, chapter 3, pages 73–105. Springer, 2011.
- [43] C. C. Lu and V. S. Tseng. A novel method for personalized music recommendation. *Expert Systems with Applications*, 36(6):10035–10044, Aug. 2009.
- [44] L. Lu, D. Liu, and H.-J. Zhang. Automatic mood detection and tracking of music audio signals. *IEEE Transactions on Audio, Speech and Language Processing*, 14(1):5–18, 2006.
- [45] M. I. Mandel and D. P. W. Ellis. Song-level features and support vector machines for music classification. In J. D. Reiss and G. A. Wiggins, editors, *Proceedings of the 6th International Conference on Music Information Retrieval - ISMIR 2005*, volume 6, pages 594–599, London, U.K., 2005.
- [46] C. Mckay and I. Fujinaga. Automatic genre classification using large high-level musical feature sets. In *Proceedings of the International Society of Music Information Retrieval Conference, ISMIR 2004*, pages 525 – 530, 2004.
- [47] D. M. McNair, M. Lorr, and L. F. Droppleman. Profile of Mood States (POMS), 1971.
- [48] O. C. Meyers. *A mood-based music classification and exploration system*. PhD thesis, Massachusetts Institute of Technology, 2007.
- [49] F. Murtagh and P. Contreras. Methods of Hierarchical Clustering. *arXiv preprint arXiv:1105.0121*, (2):1–21, 2011.
- [50] I. Muslea and M. Park. Machine Learning for Online Query Relaxation Categories and Subject Descriptors. In *KDD '04 Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 246–255, Seattle, WA, 2004. ACM.
- [51] W. Muyuan, Z. Naiyao, Z. Hancheng, M. Wang, N. Zhang, and H. Zhu. User-adaptive music emotion recognition. In *Proceedings of the 7th International Conference on Signal Processing, 2004. Proceedings. ICSP '04. 2004.*, volume 2, pages 1352–1355. IEEE, 2004.
- [52] E. E. P. Myint and M. Pwint. An approach for mulit-label music mood classification. In *Proceedings of the 2nd International Conference on Signal Processing Systems*, volume 1, pages V1–290–V1–294. IEEE, July 2010.
- [53] R. Panda and R. P. Paiva. Using Support Vector Machines for Automatic Mood Tracking in Audio Music. In *130th Audio Engineering Society Convention*, 2011.
- [54] J. R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc, San Francisco, Mar. 1993.

- [55] P. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20(1):53–65, 1987.
- [56] J. A. Russell. A circumplex model of affect. *Journal of Personality and Social Psychology*, 39(6):1161–1178, 1980.
- [57] C. N. Silla and A. A. Freitas. Novel Top-Down Approaches for Hierarchical Classification and Their Application to Automatic Music Genre Classification. In *2009 IEEE International Conference on Systems, Man and Cybernetics*, number October in IEEE International Conference on Systems Man and Cybernetics Conference Proceedings, pages 3499–3504. IEEE, Oct. 2009.
- [58] C. N. Silla Jr., A. L. Koerich, and C. A. A. Kaestner. A Machine Learning Approach to Automatic Music Genre Classification. *Journal of the Brazilian Computer Society*, 14(3):7–18, 2008.
- [59] R. Stegers, P. Fekkes, and H. Stuckenschmidt. MusiDBA personalized search engine for music. *Web Semantics: Science, Services and Agents on the World Wide Web*, 4(4):267–275, Dec. 2006.
- [60] P. Symeonidis, M. Ruxanda, A. Nanopoulos, and Y. Manolopoulos. Ternary Semantic Analysis of Social Tags for Personalized Music Recommendation. In *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR 2008)*, pages 219–224, Philadelphia, Pennsylvania, 2008.
- [61] R. E. Thayer. *The biopsychology of mood and arousal*. Oxford University Press, 1989.
- [62] G. Tzanetakis. Music analysis, retrieval and synthesis of audio signals MARSYAS. In *Proceedings of the seventeen ACM international conference on Multimedia*, pages 931–932. ACM, 2009.
- [63] Y.-H. Yang and H. H. Chen. Machine Recognition of Music Emotion: A Review. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(3):40, May 2012.
- [64] Y.-H. Yang, Y.-F. Su, Y.-C. Lin, and H. H. Chen. Music Emotion Recognition: The Role of Individuality. In *Proc. International Workshop on Human-centered Multimedia 2007*, Bavaria, Germany, Sept. 2007. ACM.
- [65] K. Yoshii, M. Goto, K. Komatani, T. Ogata, and H. G. Okuno. Hybrid Collaborative and Content-based Music Recommendation Using Probabilistic Model with Latent User Preferences. In *Proceedings of the 7th international conference on Music Information Retrieval (ISMIR)*, volume 2006, pages 296 – 301, 2006.

- [66] X. Zhang and Z. Ras. Differentiated Harmonic Feature Analysis on Music Information Retrieval for Instrument Recognition. In *Proceedings of IEEE International Conference on Granular Computing (IEEE GrC 2006)*, pages 578–581, Atlanta, Georgia, 2006.