Automatic Characterization of Music for Intuitive Retrieval

Abstract

In this work, two important approaches to automatically describe music are treated: content-based analysis of audio signals and information extraction from the Web. While the former is focussed on the physical audio signal that actually is sensed by the listener's ears, the latter is generally acknowledged to capture socio-cultural aspects of music that are not contained in the audio signal itself.

In the context of this work, the main interest of having characterizations based on the two media types (audio signal and text) is to use them as the basis for *similarity measures*. For both approaches, insights regarding existing similarity measures (of audio files and web-based artist descriptions, respectively) are gained, and methods are examined to abstract from the low level similarity measures in both areas. The aim of such abstractions is to allow high-level interactions with the content of music collections in retrieval scenarios. Elements that support these interactions are summarization and high-level description of the content of a music collection, and the possibility to conveniently formulate content-related queries.

In particular, major contributions discussed in this thesis are as follows: For audio-based similarity measures, a number of experiments is presented that lead to an improved audio similarity measure, as indicated by the presented automated classification experiments. Insights obtained in these experiments were used when building the audio similarity measure that ranked first in the 2009 MIREX Audio Music Similarity and Retrieval task. With respect to text-based artist similarity, in systematic experiments a part of the parameter space is explored to assess the optimisation potential. Furthermore, a method aiming to automatically describe artists by "topics" associated with them is discussed.

An approach to combine audio and text based similarity computations is discussed and evaluated. Finally, the use of the treated techniques is demonstrated in a number of example applications.

Zusammenfassung

In dieser Arbeit werden zwei wichtige Ansätze zur automatischen Beschreibung von Musik behandelt: inhaltsbasierte Analyse des Audiosignals und die Gewinnung von Informationen aus dem Internet. Während der Fokus bei ersterem auf dem physischen Audiosignal liegt, welches im Endeffekt von den Ohren des Hörers aufgenommen wird, zielt letzterer darauf ab, sozio-kulturelle Aspekte der Musik zu erfassen, die nicht im Audiosignal selbst enthalten sind.

Im Kontext dieser Arbeit werden derartige (aus dem Audiosignal bzw. Text) automatisch erstellte Charakterisierungen in erster Linie als Basis für Ähnlichkeitsmaße verwendet. Für beide Herangehensweisen werden Erkenntnisse hinsichtlich existierender Ähnlichkeitsmaße (von Audiodateien bzw. webbasierten Künstlerbeschreibungen) gewonnen, und es werden Methoden untersucht, um in beiden Bereichen von low-level Ähnlichkeitsmaßen zu abstrahieren. Ziel solcher Abstraktionen ist, in Retrieval-szenarios high-level Interaktionen mit dem Inhalt von Musiksammlungen zu ermöglichen. Elemente, die solche Interaktionen unterstützen, sind Zusammenfassung und Beschreibung des Inhalts einer Musiksammlung auf hoher Ebene, und die Möglichkeit, bequem inhaltsbezogene Anfragen zu stellen.

Insbesondere werden folgende Hauptbeiträge in dieser Arbeit behandelt: Im Bereich audiobasierte Ähnlichkeitsmaße wird eine Reihe von Experimenten vorgestellt, welche zu einem verbesserten Audioähnlichkeitsmaß führen, wie durch die angeführten automatisierten Klassifikationsexperimente gemessen bzw. angezeigt wird. Aufbauend auf den bei diesen Experimenten gewonnenen Erkenntnissen wurde das Ähnlichkeitsmaß konstruiert, das im MIREX 2009 Audio Music Similarity and Retrieval Task den ersten Rang belegte. Hinsichtlich der textbasierten Künstlerähnlichkeit wird in systematischen Experimenten ein Teil des Parameterraums exploriert, um das Optimierungspotential abzuschätzen. Des weiteren wird eine Methode diskutiert, die darauf abzielt, automatisch Künstler durch mit ihnen assoziierte “Themen” zu beschreiben.

Ein Ansatz zur Kombination von audio- und textbasierten Ähnlichkeitsmaßen wird diskutiert und evaluiert. Schließlich wird der Nutzen der behandelten Techniken in einer Reihe von Beispielanwendungen demonstriert.

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Dissertation selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Acknowledgments

The most important person to make this thesis possible is *Gerhard Widmer*. Offering both the freedom to choose the most interesting research topics, and giving support where needed, he is as close to a perfect supervisor as I can imagine. I am also grateful to *Andreas Rauber* for reviewing this thesis.

Thanks to my colleagues at Department of Computational Perception for making the work environment enjoyable by being part of a relaxed and friendly atmosphere, and for the insightful discussions. Additionally, many thanks to *Peter Knees* and *Markus Schedl* for their insightful comments. Also many thanks to *Dominik Schnitzer, Arthur Flexer* and *Martin Gasser* from the OFAI music group for the inspiring knowledge exchange.

Last but not least, I would like to thank *Stephan Baumann* for opening the doors to my engagement in Music Information Retrieval in the first place.

This dissertation has been funded by the Austrian Fonds zur Förderung der Wissenschaftlichen Forschung under project numbers L112-N04 and L511-N15.

Contents

1. Introduction

The way people use and interact with music has been changing vastly over the last decades. A major factor in this change is the digital storage and reproducibility of music, which facilitates online stores offering huge amounts of music for instant downloading, and mobile music players capable of holding increasing numbers of songs in their memory. These developments induce a need for specialised methods to handle larger amounts of music tracks. Typical problems arising from this are:

- With personal music collections growing beyond a certain size, users will likely have a growing need for assistance in organizing and accessing the music in their collection. Examples of such supporting functions are playlist generation algorithms, and the automated grouping of music into groups of similar music. The latter can be of use e.g. in mobile music players, where the user might want to select only a type of music instead of navigating to a particular piece of music. Limiting interactions to a high level may reduce the need for the user's attention, which is likely to be rare in mobile usage scenarios.

- Due to limitations of space, physical-world record stores typically offer the most popular music having the highest sales figures[1]. In contrast, online record stores could supply their customers also with less-known music that has lower sales figures. In general (i.e., not limited to the domain of music distribution), the multitude of less popular items, made available to customers, is referred to as Long Tail[1]. Obviously, providing well matching and interesting recommendations from the long tail can both increase customer satisfaction and sales figures. While methods discussed in this thesis can be used to create content-based music recommender systems, they also may be used to complement recommender systems based on collaborative filtering [Celma, 2008] that suffer from an issue known as cold-start problem. The system can not recommend items for which no user-generated data, such as ratings, is available yet. For example, this is the case for items that are newly added to the catalogue.

Being able to automatically estimate the similarity of music pieces, or music artists, can be an important building block for algorithms that aim to remedy such issues. For music recommendation, automated similarity estimation methods have the advantage that they may take into consideration all music in the current collection, or catalogue, (almost) regardless of the collection

[1] http://www.wired.com/wired/archive/12.10/tail.html, retrieved 15. 11. 2009

size, and (almost) regardless of the popularity of the music. When using the calculated similarities for music collection structuring, a structure intrinsic to the data might be revealed that was not previously apparent to the user.

This work contributes to the research field of music similarity estimation by examining approaches to potentially improve similarity measures between music-related items based on two sources of information: the audio content of pieces of music, and textual descriptions of music artists derived from the Internet. In both areas, insights regarding existing similarity measures are gained, and methods are examined to abstract from the low-level similarity measures. The aim of such abstractions is to allow high-level interactions with the content of music collections in retrieval scenarios. Elements that support these interactions are summarization and high-level description of the content of a music collection, and the possibility to conveniently formulate content-related queries.

1.1. Audio Content

By applying Digital Signal Processing (DSP) techniques, the audio signal of music recordings is analyzed. The present work contributes to our understanding of audio similarity measures in two ways.

First, in Chapter 2, in extensive experiments the effect of modifications of existing similarity measures is evaluated. Similarity measures are modified by changing the distance functions between audio features, by using modified features and by using new feature combinations. In particular, indication is found that complementing frame-based audio features with two features we call *Harmonicness* and *Attackness* contributes to an improved audio similarity measure. Measured by genre classification experiments, incorporating the suggested modifications is shown to improve the results obtained by an existing audio similarity measure by about 5.8 percentage points, for instance (see Figure 25, page 80). Insights gained in these experiments were used when building the audio similarity measure that ranked first in the 2009 MIREX Audio Music Similarity and Retrieval task[2].

Second, in Chapter 4 a method to extract higher-level sound building blocks from the raw audio data is discussed and evaluated. Describing audio by such building blocks may be useful for building future similarity measures.

[2] http://www.music-ir.org/mirex/2009/index.php/Audio_Music_Similarity_and_Retrieval_Results

Presented results indicate that this method yields comparable accuracies as the widely-used MFCCs (computed frame-wise) in the used evaluation setting.

1.2. Textual Information

Using text information retrieval techniques, the web pages related to music artists are analyzed. A variety of methods to calculate similarity functions based on this data is evaluated in Chapter 3. From these experiments, a number of conclusions is drawn considering the relative importance of the algorithm's building blocks.

To bring artist descriptions to a higher level, a previously suggested method to describe document topics is transferred to describe aspects of music artists. The automatically found aspects can be used to describe and retrieve artists in a compressed and more human-understandable form (Chapter 5).

1.3. Combining Audio and Web Based Similarity Measures

A straightforward method to combine audio and web based similarity measures is discussed and evaluated, showing that results obtained by the similarity measure based only on audio can be improved by adding web-based information. This contribution is the topic of Chapter 6.

1.4. Applications

Finally, some of the methods discussed and evaluated in this work are used to create example applications. For example, it is demonstrated that the discussed techniques can be used to enhance user interfaces of music players, or even create a novel interface concept for music players. Also, an application to browse music artists is described, and further possible application scenarios are drafted.

In summary, the work presented in this thesis gives a number of impulses to the domain of music similarity search. For audio-based similarity measures, a number of experiments is presented that lead to an improved audio similarity measure, as indicated by the presented automated classification experiments. Insights obtained in these experiments also were used when building the audio

similarity measure that ranked first in the 2009 MIREX Audio Music Similarity and Retrieval task. With respect to text-based artist similarity, in systematic experiments a part of the parameter space is explored to assess the optimisation potential. Furthermore, a method aiming to automatically describe artists by "topics" associated with them is discussed. A method to combine audio and text based similarity computations is discussed and evaluated. Finally, the use of the treated techniques is demonstrated in a number of example applications that maybe even give a first taste of the way music is consumed in the future.

2. Similarity of Audio Signals

In this chapter the topic of music similarity measures that are based solely on the audio signal is discussed (so-called *content-based* music similarity measures). One main use of such similarity measures is to automatically find similar sounding music in large collections of music. This can be useful in cases where the audio signal of the music is the only information available for music from the long tail. However, applications of similarity measures are not limited to this scenario. Examples of other scenarios include the following.

- Collection clustering. Automatically grouping music that is likely to sound similar can be used to create an overview of a given music collection. Examples of this are the *Islands of Music* [Pampalk, 2001] and nepTune [Knees et al., 2006b].
- Playlist generation. Using a music similarity measure, an application for automatic track selection based on skipping behavior is presented in [Pampalk et al., 2005c].
- Classification. By using a kNN classifier, a given music similarity measure can be used to classify music. This way of classification is also a possible way to evaluate music similarity measures (e.g., [Pampalk, 2006b]).
- Detection of duplicate tracks in large collections. As tracks with a similarity above a certain threshold may be considered as being identical, music similarity measures also may be used to detect duplicate tracks.

This chapter starts with a general view on the algorithm outline (Section 2.1), followed by a brief discussion on how to compare and evaluate different approaches for music similarity computation (Section 2.2). Then, the two algorithms to compute music similarity that ranked top in the MIREX Audio Similarity Tasks held in 2006 and 2007 are described, which are used as a starting point for the experiments in this chapter (Sections 2.3 and 2.4).

The biggest part of this chapter is dedicated to systematically evaluating how these algorithms can be improved. Two approaches are being followed: modified or additional audio features and alternative ways to compute the distance from the features. These experiments are divided into three sections. Two sections are dedicated to evaluate modifications to the two major parts of the algorithm used as starting point (Section 2.5 and 2.6). One of the most interesting aspects discussed is the use of an audio feature that describes the amount of harmonic and percussive elements in the spectrogram, called *Harmonicness*

and *Attackness*. A third section describes experiments how to combine the parts of the algorithm into a final similarity measure (Section 2.7). An algorithm that incorporates the suggested changes is shown to outperform the two algorithms that are used as a starting point (Section 2.8).

2.1. Algorithm Outline and Common Audio Features

In this section, an overview of the commonly used algorithm structure is given, and a number of commonly used audio features is described.

2.1.1. Coarse Structure of Algorithm

Many audio analysis algorithms follow a comparable coarse structure, which can be represented as two major steps.

1. *Feature Data Extraction.* In most cases, the audio data to analyze is a track of music, but also it could be an excerpt of an audio stream. In the first step, feature data is extracted from this audio data to represent the relevant aspects in a compressed form.

2. *Usage of Feature Data.* The second step is to use the extracted feature data. If the application at hand requires to classify (or label) the data, the extracted feature data is used to classify the current instance into one of several previously learned classes. Depending on the format of the feature data, many different classification algorithms can be used in this step.

   If the application requires an estimation of how *similar* two instances are (e.g., for retrieving similar items), then the feature data is used to estimate this similarity by defining a similarity (or distance) function on top of it. If distances can be computed, then also classification can be done (e.g., by simple kNN-classification, or Support Vector Machines, e.g. [Mandel and Ellis, 2005a]).

The feature extraction part of this process contains much of the complexity of this process. The more robust the features (and the more relevant the aspects of the signal they describe) the more robust the whole algorithm can be. Feature extraction is illustrated in Figure 1. The steps are discussed in the following paragraphs. The realisation of each of these steps can have an influence on the overall performance of the algorithm.

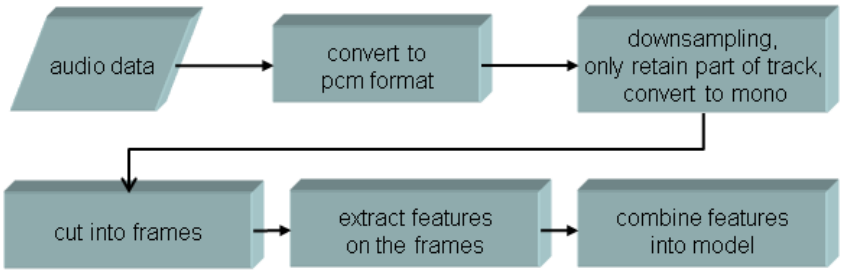

Figure 1: Typical feature extraction phase. In BOF approaches, the temporal order of frames is discarded.

2.1.1.1. Conversion to time series signal. In most cases, the audio data is given in a compressed from, e.g., in mp3 format. While in some cases, it is possible to analyze this compressed form directly (e.g., using a toolkit such as Maaate[3]), in most cases it leads to more straightforward algorithms to convert the data into time series with equitemporal sampling.

2.1.1.2. Data reduction. The purpose of the next step (downsampling, conversion to mono, only retaining a part of the original signal) is mainly to reduce the amount of data that is processed, which reduces computation time. However, one has to be aware that important information may be contained in the discarded parts of the signal, such as stereo information [Tzanetakis et al., 2007], or high-frequency content above the Nyquist frequency of the downsampled signal. For example, when the signal is converted to a sampling rate of 11 kHz, then the frequencies above 5.5 kHz are lost. Depending on sound engineering that was used during recording, in this case high percussive sounds (e.g., Hi-Hat sounds) may be suppressed to a large extent. The influence of sample rate has been examined in [Aucouturier and Pachet, 2004].

2.1.1.3. Frame-wise processing. Audio feature extraction is typically based on audio frames that are short enough so that the contained signal can be considered as stationary. For example, a typical frame length is 46 ms. A common operation on each audio frame is the transformation from time domain to frequency domain, as discussed in Section 2.1.2. During this transformation, the samples in the frame are windowed to avoid edge effects. As windowing reduces the weights of the samples located at the beginning and end of the frame, consecutive frames are chosen to overlap, e.g., by 50%.

[3] http://maaate.sourceforge.net/

2.1.1.4. Feature Extraction. Probably the most important aspect of feature extraction is which audio features to choose. A variety of audio features has been suggested, of which some important ones are described in Section 2.1.3. Many of these are calculated on only one frame, but there are also features that are computed on a number of consecutive audio frames.

2.1.1.5. Combination of Feature Data Into Model. The features extracted in the previous step usually are not in a format suited to serve as feature data for the whole song. For example, they usually are given on a per-frame basis. To combine this data into a more compressed form that can be used for classification or similarity computation, typically statistics are calculated on them. Common choices are mean and variance of a feature value over all frames, but also more advanced ways of building a common model are available, most notably Gaussian Mixture Models, as discussed in Section 2.1.4.

This concludes the discussion of the main aspects of feature extraction on a coarse level. The next sections give information about crucial steps in more detail.

2.1.2. Transformation Into Frequency Domain

While much information can be extracted from the time domain signal directly, many features rely on a representation in the frequency domain instead of the time domain. This is motivated by human hearing in which frequencies play an important role. The most widely used algorithm for accomplishing this is the Fast Fourier Transformation (FFT). By taking the absolute value of the FFT transformation of a frame, the amplitudes of the signal at frequency bands with center frequencies of $n \cdot f_o$ are obtained, where f_0 is the framesize divided by the sample rate of the signal.

2.1.2.1. Cent Scale. A drawback when using FFT to model musical signals is that FFT bands are linearly scaled, while music follows a logarithmic scale. For example, if the frequency of a tone is doubled, this results in the tone being one octave higher, and the equal tempered scale is obtained by dividing an octave equally by 12 on the logarithmic scale. It is possible to transform the FFT representation into a cent-scale representation by summing and weighting FFT bands. This is mainly used to extract chroma features, to detect the activation of frequencies associated with each pitch class in the different octaves folded together (e.g. [Goto, 2006]).

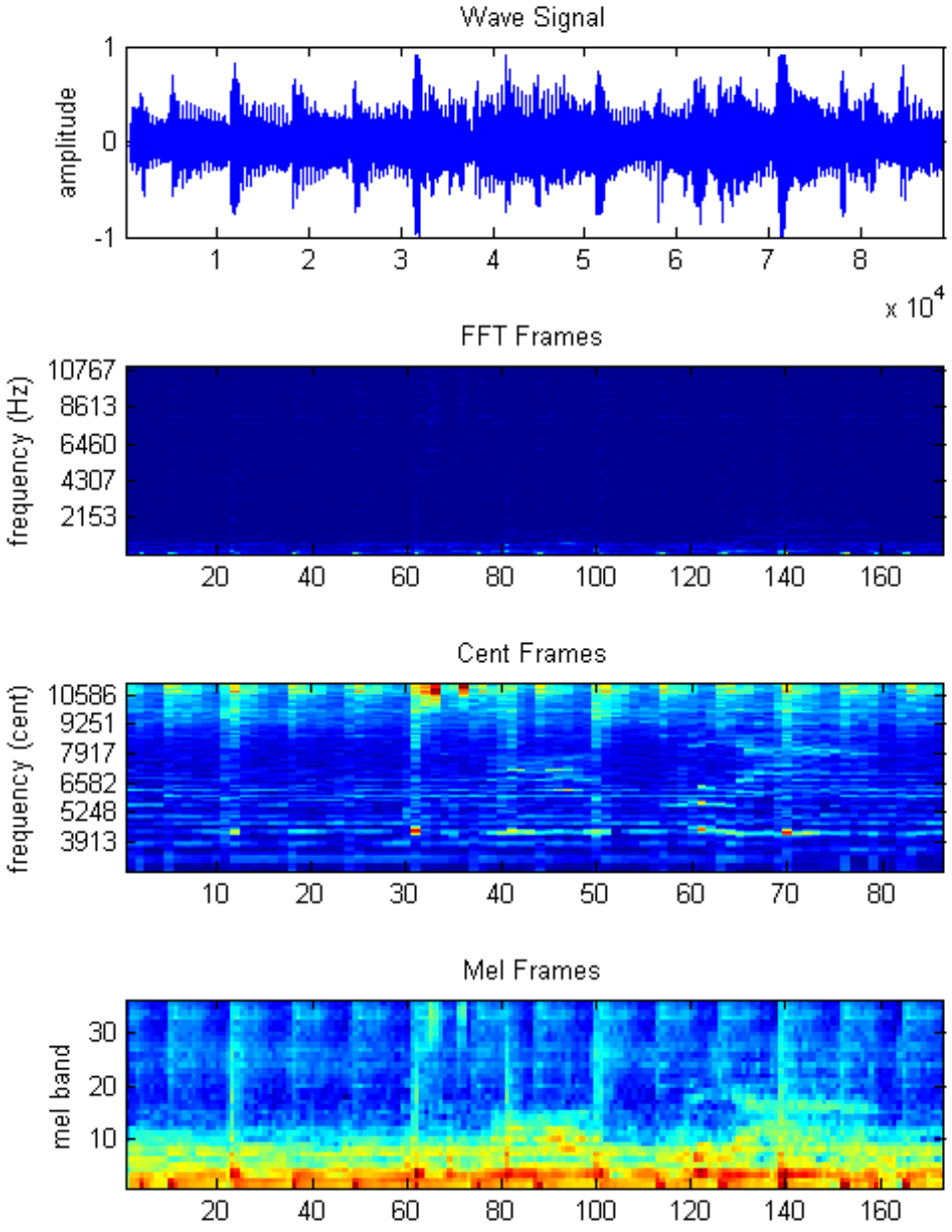

Figure 2: The same audio excerpt in different representations.

2.1.2.2. Mel Scale. For a model that is more closely associated with perception than the cent scale, commonly the mel-scale is used. The mel-scale is designed to reflect the perceived pitch differences of pure sinusoidals. These perceived differences differ from the cent scale. The mel-scale, like the cent scale, in higher frequencies is a logarithmic scale. The mapping from mel scale to Hz scale is determined empirically by listening experiments [Fastl and Zwicker, 2007]. A sufficiently large number of subjects is asked to indicate what ratios lie between pure tones (or narrowband noises) that they perceive as having half, or double, pitch. It turns out that up to a frequency of about 1000 Hz for the upper tone, the lower tone has about half the value in Hz. Above 1 kHz, the lower tone gradually gets assigned a clearly lower Hz value. Conversion of values larger than 1000 Hz to mel can be done by [O'Shaughnessy, 1987]:

$$m = 2585 \log_{10}(1 + f/700) \tag{1}$$

The mel scale is associated with critical bands. 1 Bark corresponds to 100 mel [Fastl and Zwicker, 2007]. Based on this observation, the spectrum can be represented by means of critical bands. For example, in the G1C algorithm [Pampalk, 2006b], the spectrum is initially represented as 36 bands, of which each has a width of approximately 85 mel. Typically, the amplitudes in each band then are transferred into the log domain to better model human loudness perception. The various representations are illustrated in Figure 2.

2.1.2.3. Additional Processing. Sometimes, the audio signal is further processed. For example, a weighting of the different frequencies depending on the loudness sensation in different frequency bands can be superimposed on the spectrum, or spectral masking effects can be simulated (e.g. [Pampalk et al., 2004]).

2.1.3. Typical Audio Features

A great variety of audio features has been proposed in the literature. An overview of some audio features that have been used for music classification and similarity computation can be found in [Pohle, 2005, Pohle et al., 2005a]. Here, some audio features are drafted that may be considered as "typical" audio features. For a number of audio features, software is available to compute them. One of the first frameworks for music feature extraction was MARSYAS [Tzanetakis and Cook, 1999a]. A Matlab toolbox to extract audio features is presented in [Pampalk, 2004]. Further implementations for audio feature

extraction are contained, for example, in YALE [Fischer et al., 2002], ACE [McKay et al., 2005] and COMIRVA [Schedl et al., 2007].

2.1.3.1. Examples of Time Domain Features Time domain features are extracted from the waveform of the audio frame. Probably the simplest time domain feature is the *Zero Crossing Rate*, which simply is the number of times the amplitude of the audio signal crosses the zero point in the current frame, which is linked both to pitch and noise, as noted in [Pampalk, 2006b]. Time domain features such as *Root Mean Square* have been used to extract information about the volume (e.g. [Tzanetakis and Cook, 1999b]). Linear Prediction Coefficients (e.g., [Li et al., 2001]) can for example give information about clicks in the signal.

2.1.3.2. Statistics on Spectrum of Frame. A number of audio features can be calculated from the spectrum of a single audio frame. Many of them are related to aspects of *timbre* sensation, which refers to sound quality, independent from pitch and loudness (cf. [Fastl and Zwicker, 2007, Aucouturier, 2006]). For example, the center of the magnitude distribution, called *Spectral Centroid*, is related to brightness. A somewhat related measure is the *Spectral Rolloff*, which is the frequency band index below which a given portion (e.g., 85%) of the signal energy is accumulated (e.g. [Tzanetakis and Cook, 2002]). Another feature based on the time scale of individual frames is *Spectral Flux*, which is a measure of the rate of change between consecutive frames. The spectrum such features are calculated from either is the FFT spectrum, or it can be a logarithmically scaled spectrum ([mpe, a]).

2.1.3.3. MFCCs Perhaps the most important single feature in music information retrieval are Mel Frequency Cepstral Coefficients (MFCCs). They have been adapted for music signal analysis from speech processing [Logan, 2000, Logan and Salomon, 2001]. MFCCs are calculated from a frame that is transformed into a mel-scaled frequency representation with amplitudes given on the logarithmic scale. For instance, this representation has 40 frequency bands. Typically, the activations of these bands are highly correlated. Decorrelation is done by Principal Component Analysis, or with comparable results ([Logan, 2000]) by Discrete Cosine Transform (DCT). The result of this step are MFCCs. The 0^{th} MFCC reflects the average amplitude of the spectrum, and lower-order MFCCs are a coarse representation of the spectrum's shape. Higher-order MFCCs contain information about the more fine-grained shape

of the spectrum, which usually is discarded. An illustration of MFCCs is given in Figure 5.

2.1.3.4. Features Related to Rhythm. As development in time is one of the key aspects of music, time-dependent features are of particular interest. Techniques such features can be based on are inter-onset analysis, or periodicity estimation functions such as autocorrelation, FFT and comb filterbanks. The topic is discussed in depth in [Gouyon, 2005].

A possible first step in calculating rhythm features is to estimate onset strengths at different frequency bands, and then combining these to obtain an overall function. In [Ellis, 2006] onset strengths are estimated on 40 mel bands, which are summed up and periodicities are then extracted by applying an autocorrelation function. The *Beat Histogram* [Tzanetakis et al., 2001, Tzanetakis and Cook, 2002], cf. [Scheirer, 1998] is the autocorrelation function of the sum of the time-domain amplitude envelopes in several frequency bands. Various features are obtained from a Beat Histogram, such as the periodicity, measured in bpm, of the highest peak. A *Periodicity Histogram* (PH) [Pampalk et al., 2004] indicates how often within a music track different strength levels occur at different periodicities. The similarity of two PHs can be calculated by interpreting them as vectors and taking the Euclidean distance. For summarizing the periodicity estimation function, MFCC-like descriptors have been proposed [Gouyon and Dixon, 2004].

Fluctuation Patterns (FPs) [Pampalk et al., 2002, Pampalk et al., 2003] measure both rhythmic and frequency aspects of the sound. They are designed to indicate perceived periodicities (fluctuation strengths) at various time scales and frequency bands. FPs are calculated from audio chunks of approximately 3 s length. The computation of FPs is described in more detail in Section 2.3.1.

2.1.4. Combining Feature Data for a Track

Usually, MIR applications are based on statistics of feature values over a chunk of audio (or a whole audio track). There are at least two reasons for doing so. First, storing all feature data (e.g., feature vectors of all frames of a track) typically consumes magnitudes more storage than only storing statistics calculated from them. Second, statistics of the feature values is in many cases better suited as input for the algorithmic steps that follow the feature extraction phase. For example, a time series of features typically can not be used

as useful attribute for a classification algorithm, while the mean and variance of the given feature are better-suited attributes. Such an approach of using mean and variance of features as attributes is used in [Tzanetakis and Cook, 2002]. Within the same lines, characteristics of peak positions (e.g., within a spectrum or autocorrelation function) are of interest and can be used as the basis for attributes [Mierswa and Morik, 2005].

Due to the temporal structure of music, mid-term characteristics of the signal can be of interest. For example, [Tzanetakis and Cook, 2002] use a fixed-size *Texture Window* to calculate mid-term statistics over segments of length of e.g. one second. These statistics (mean and variance) then are combined over all texture windows in a track. In this case, the texture window has a fixed length. Also, it has been proposed to use an onset detection algorithm to cut the track into segments [West and Cox, 2005].

2.1.4.1. Gaussian Mixture Models. Instead of simple statistics of individual features, *Gaussian Mixture Models* (GMMs) can be used to model dependencies between features to a certain extent. GMMs model a set of observations by explaining each observation as being produced by one of K multivariate Gaussians (e.g., [Aucouturier and Pachet, 2004, Pampalk, 2006b, Jensen et al., 2007]).

$$p(x) = \sum_{k=1}^{K} c_k \frac{1}{\sqrt{|2\pi\Sigma_k|}} \exp(-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)) \quad (2)$$

The process of generating points with this distribution can be imagined by first choosing one of the K Gaussians (Gaussian k is chosen with probability c_k), and then one point is generated according to $\mathcal{N}(\mu_k, \Sigma_k)$. A common way to estimate the parameters c_k, μ_k and Σ_k from a set of given points is by applying an expectation maximization (EM) procedure.

Two GMMs can be compared by estimating the probability that a set of points generated by one GMM is generated by the other. For example, the first GMM is used to generate a set of random points, and the probability of these points being generated by the second GMM is estimated, and vice versa [Aucouturier and Pachet, 2004].

Using a single Gaussian to model the MFCC frames of a song ([Mandel and Ellis, 2005a]) turned out to perform roughly comparably to using a GMM in a number of experiments ([Mandel and Ellis, 2005a, Pampalk, 2006b, Levy and Sandler, 2006, Jensen et al., 2007]), while being computationally more effective.

2.1.5. Examples of MIR Algorithms Based on Audio Features.

The area of classification based on audio features has been extensively studied. Genre classification algorithms are presented in [Tzanetakis and Cook, 2002]. Learning tags associated with audio excerpts has been proposed in [Aucouturier et al., 2007, Eck et al., 2007, Torres et al., 2007]. Approaches to using such labels that are assigned to a track by the machine learning algorithm to improve similarity estimation of audio tracks are presented in [West and Lamere, 2007, Barrington et al., 2007].

2.2. Evaluation Techniques

After in the previous section, a coarse overview of some techniques commonly used in the field of music signal analysis was given, in this section, issues associated with evaluating music retrieval algorithms are discussed. This section also covers the evaluation procedure that will be used throughout this thesis, and the data used for the evaluations presented in this chapter.

In retrieval tasks, it is of interest to find (a) particular item(s) in a – usually large – database. In the case of music audio similarity, usually the scenario is a query-by-example scenario. The query takes the form of one (or more) music tracks, or excerpts, and the task is to retrieve tracks from a music collection that match the query. In the case of music similarity algorithms "match" means to sound similar. While still being an imprecise definition, it turns out that human subjects are quite consistent in rating which tracks sound similar (cf. [Pampalk, 2006b, Pampalk, 2006a, Novello et al., 2006]).

Usually, during development of audio based music similarity algorithms, these algorithms are tested on one or more music collections with associated meta data giving information about similarities of the music pieces. In the optimum case, this meta data would consist of reliable (i.e., verified) pairwise similarity ratings between all pairs of tracks. The method generally accepted to be most reliable for this validation is by human assessment. As associating each pair of tracks with a human rating is costly both in terms of time and money, this method would be feasible only for small music collections. However, as differences in various variants of an algorithms may be rather small, and to test for robustness, it is generally preferable to have tests run on larger music collections.

A number of ways have been proposed to circumvent this problem [Berenzweig et al., 2003]. One kind of solution is to analyze user data. *Playlist (or music*

collection) co-occurrence [Logan et al., 2003] is based on the idea to collect a large number of human-generated playlists. Assuming that similar music appears more frequently in the same playlist (or user collection), a (normalised) co-occurrence of items is used as ground truth for similarity. Potential problems with this approach include the problem of data availability. As large amounts of data may be necessary to obtain sufficiently stable similarity scorings, distances to tracks for which only little data is available are likely to be noisy, and for very unknown artists it is likely that data is not available at all. On the other hand, unknown artists are of particular interest, as one of the strengths of audio based approaches is that tracks are suggested regardless of their popularity.

Last.fm[4] collects usage data, and makes data such as lists of similar artists available over a web service[5]. However, it is not obvious how this data is generated. For example, data collected from the users might be complemented by data collected from the web, or by audio similarity measures. Using such complemented data as ground truth obviously would induce a bias.

Probably the most common approach to automated similarity evaluation is to obtain genre labels for the tracks, and use these as a ground truth. It is assumed that tracks in the same genre are generally similar in some relevant ways. A suitable similarity algorithm would rank such very similar items high. Based on this notion, k-NN genre classification accuracy is frequently used as an indication of the algorithm's quality. As in this work, evaluation is also based on k-NN genre classification accuracy, this issue is discussed in detail in the next section.

2.2.1. Evaluation by Genre Classification

While the assumption that very similar tracks are in the same genre is an important grounding of using k-NN genre classification experiments as a basis for evaluation, it is not the only aspect. Another argument for using genre classification accuracy for evaluation of music recommendation systems is based on common sense. Asking people what kind of music they like, it is reasonable to expect an answer based on genre, e.g., "I like Goth", "I listen to a lot of classical piano music", etc. Such an argumentation is of importance when taking into account that one important application scenario of music similarity algorithms is music *recommendation*.

[4] http://last.fm

[5] http://www.audioscrobbler.net/
http://www.lastfm.com/api

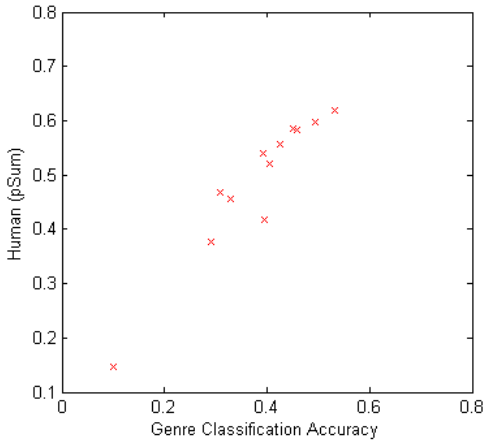

Figure 3: MIREX'07 results: Average (probabilistic) genre classification accuracy versus human judgment. Each data point represents the values of one participating algorithm. The overall impression is that genre classification accuracy and human judgments are highly related.

In the literature, further support for the evaluation of audio similarity algorithms by k-NN genre classification is given. [Novello et al., 2006] present a user study based on 19 tracks from 8 genres, finding that different subjects' similarity ratings are rather concordant, and gives support for genre as a dimension of human similarity ranking. In [Pampalk, 2006b], two algorithms are evaluated by genre classification and human listening tests, and in both cases the same algorithm scored higher. However, in this experiment, it also turned out that overall, "more tracks from different genres are rated as perfect matches than songs from the same genre". It seems noteworthy that this effect is even desirable, as it points out the algorithm's strength to suggest matching tracks that are not in the same genre. At the same time, it indicates that the algorithm might actually perform better than indicated by genre classification figures. This should be kept in mind when interpreting genre classification accuracies in the context of music similarity. It might be even impossible to improve accuracy above a certain value.

To gain further insight into the relationship between genre classification accuracy and human ratings, a visualization based on data collected in the MIREX'07 Audio Similarity and Retrieval Task[6] is given in Figure 3. The figure shows the average genre classification accuracy of each algorithm in relation to the average human-assigned broad similarity score $pSum$[6] of each

[6]available at http://www.music-ir.org/mirex/2007/index.php/MIREX2007_Results

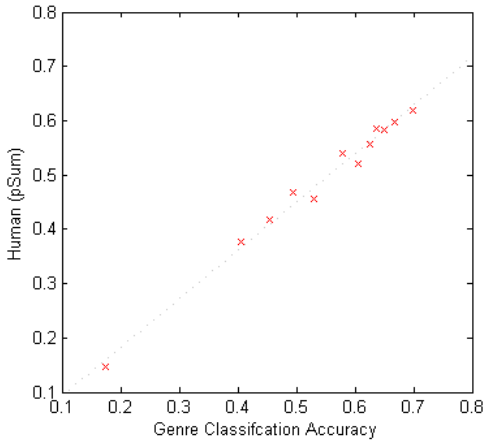

Figure 4: Same figure as in Figure 3, but accuracy calculated with the classes *Romantic*, *Classical* and *Baroque* merged. The dotted line indicates a linear fit.

algorithm. This score was obtained by assigning each suggested track that was ranked *not similar* a score of 0, tracks that were ranked as *somewhat similar* a score of 1, and *very similar* tracks a score of 2. Scores are normalized in the range 0..1, i.e., an algorithm for which all candidates are scored "not similar" would be assigned a pSum score of zero, while an algorithm for that all candidates are scored "very similar" would be assigned a pSum score of 1.0. It can be seen that there is a certain relationship between pSum (i.e., human judgment) and the genre membership of the suggested tracks[7]. However, the relationship seems not fully straight, i.e., there seem to be some deviations in the middle part of the figure. This phenomenon can be examined by analyzing the algorithms' genre confusion matrices. It turns out that most algorithms have noticeable confusions between the genres *Romantic*, *Classical* and *Baroque*. Without knowing for sure, it may be the case that the found pieces actually sound quite similar (e.g., due to similar instrumentation) although they were categorised into different genres. Without access to the actual tracks used in the evaluation, it is not possible to know if this is the case.

Based on these observations and thoughts, a second figure (Figure 4) is given with the corresponding results when the classes *Romantic*, *Classical* and *Baroque* are merged. It can be seen that there is a much closer relationship between genre classification accuracy and human judgments in this case.

[7]Figures based on other human judgment measures than pSum are roughly comparable. pSum was chosen because it seems to show the assumed relationship between genre classification accuracy and human scoring most clearly.

As a conclusion, it seems that with an appropriate genre labeling, it is possible to model the human view with sufficient precision to conduct automated evaluations using genre classification accuracy. However, dividing a set of similar sounding tracks into various genres (e.g., due to socio-cultural aspects such as a historic period) is counterproductive to this evaluation approach, as it may induce additional randomness into the classification result. It seems likely that when comparing algorithms over a number of collections, these noise effects level out – in addition to minimizing the danger of overfitting to a particular collection.

2.2.2. Used Evaluation Score

In this thesis, automated evaluation of audio similarity measures based on genre labels is conducted in a leave-one-out k-NN genre classification manner. Each evaluated algorithm is applied to obtain the k closest (most similar) tracks for each track t in the music collection in turn. The *percentage* of the k closest tracks that have the same genre as the seed track t is taken as the quality score for the recommendations associated with track t, denoted p_t. This is equivalent to calculating precision at k. We refer to this measure as the (probabilistic) genre classification accuracy of track t. Obviously, choosing a higher number for k yields scores with higher resolution (and usually lower overall figures), which can have an impact on the results of the Friedman test (see below). In most experiments in this thesis, k was chosen to be 20.

The score p_t is used in two ways. First, its average for algorithm A $\overline{\mathrm{acc}_\mathrm{A}}$ over all tracks is assumed to be a reasonable indication of the algorithm's performance. It is calculated as

$$\overline{\mathrm{acc}_\mathrm{A}} = \frac{1}{T}\sum_{t=1}^{T} p_t \tag{3}$$

where T is the number of tracks in the collection. The second way the p_t are used is to rank different algorithms by their performance for track t when the Friedman test is calculated, as discussed in Section 2.2.3.

2.2.2.1. Artist Filter It has been noted that the audio signals of tracks by the same artist may have aspects in common that are perceptually not so relevant, such as production effects [Pampalk, 2006b]. To reduce the risk that the algorithm is optimised towards such perceptually less relevant aspects, following [Pampalk, 2006b], an *artist filter* is applied in some experiments discussed in this thesis. An artist filter prevents tracks by the same artist

as the seed track's artist to appear in the set of k closest tracks, i.e., the k closest tracks with different artist than the seed track's artist are used as the evaluation basis.

2.2.3. Friedman Test

In several tasks of the second and third runnings of the Music Information Retrieval eXchange (MIREX2006 and MIREX2007), the Friedman test ([Friedman, 1937]) was applied to test participating algorithms for significant differences in their performances [Downie, 2006]. The evaluation setting in this work is basically comparable to the one in the MIREX Audio Music Similarity and Retrieval task. Human-assigned similarity scores are replaced by measuring how many of the k closest tracks are in the same genre as the current query track. Thus, the Friedman test can be applied in a similar manner (cf. [West, 2008]). In the human evaluation, 60 or 100 query tracks are used, while the automated experiments are run on all tracks. One should keep in mind that with a larger number of queries, smaller differences can be shown to be significant than with a smaller number of queries (i.e., there is a higher statistical power).

Using the Friedman test, only the relative performance of algorithms with respect to each track to be classified are compared, not how much they differ as measured by overall average classification accuracy. I.e., in the scenario used here, when comparing two algorithms A and B, if for Algorithm A all k nearest neighbors of a track t are in the correct genre, then it does not matter if for Algorithm B only $k-1$ tracks are in the correct genre, or none – both cases are treated the same by the Friedman test (cf. [Pampalk, 2006a]).

In this manner, all algorithms A_i under test are ranked for each seed track t, resulting in a rank $r_{A_i,t}$ for the given seed track t and algorithm A_i. If an algorithm A_1 performs *better* than algorithm A_2 for seed track t, then A_1 is ranked *higher* than A_2 for this track. Results of the Friedman test are presented as a multiple comparison graph, such as e.g. shown in Figure 7. Algorithms are represented as lines that indicate significance bounds, with the x-Axis denoting the mean rank of an algorithm over all seed tracks ($\frac{1}{T}\sum_{t=1}^{T} r_{A_i,t}$). I.e., "better" algorithms are indicated by lines more to the right, and if the lines of two algorithms don't overlap, then they show a significant difference according to the Friedman test.

In the experiments, there is typically some sort of baseline algorithm one wants to improve upon, e.g., an existing algorithm. Usually, the baseline algorithm

is shown in blue colour, and its significance bounds are shown as gray vertical lines. Algorithms for which no significant difference is measured by the Friedman test in the current experiment are shown in gray colour, while algorithms for which significance is measured are shown in red.

The order algorithms are listed in the figure from top to bottom can be chosen arbitrarily. In this thesis, they are sorted according to an overall score, which in most cases is the average percentage of closest tracks in the correct genre $\overline{\text{acc}_\text{A}}$. Better algorithms are ranked on the top. In general, algorithms with higher $\overline{\text{acc}_\text{A}}$ are ranked more to the right, but there can be exceptions to this general tendency.

2.2.4. Music Collections Used

In this work, the following audio collections were used for evaluation. The names are the same as in [Pampalk, 2006b] for those collections that already are used there. Basic statistics are given in Table 1. More details about the genres each music collection contains are given in Table 2. DB-NORM, DB-MS and DB-L* were used to test ideas and optimize parameters. DB-CC and DB-HOMB were not used for optimization (to avoid overfitting). They are only used for evaluating the resulting algorithm.

Name	*Tracks*			*Genres*	*Artists*			*Naive*
	Total	in a Genre			Total	in a Genre		*Baseline*
		min	max			min	max	
DB-MS	729	26	320	6	128	5	40	43.9%
DB-L*	2445	44	477	13	103	3	28	19.5%
DB-CC	871	81	200	6	469	27	150	23.0%
DB-NORM	100	10	10	10	92	7	10	10.0%
DB-HOMB	1886	47	504	9	1492	39	448	26.7%

Table 1: Music collections used in this work. *Naive Baseline* is the result of an algorithm that would during classification assign the most frequent class to each item.

2.2.4.1. DB-MS. The DB-MS (Magnatune, small) collection consists of 729 tracks, divided into 6 genres. It is compiled from music from Magnatune[8] that is available under a creative commons license. This collection was compiled by

[8] www.magnatune.com

MTG[9] and was used as training set in the genre classification part of the Audio Description Contest held at ISMIR 2004. The contained music tracks and associated meta data can be downloaded from the ISMIR 2004 web site[10]. This allows for reproducing and comparing results obtained on this collection by different researchers. This collection has been used in a number of MIR-related publications, e.g. [Levy and Sandler, 2006, Moerchen et al., 2006, Pampalk, 2006b, Jensen et al., 2007].

2.2.4.2. DB-L*. The tracks in this collection are mostly identical to those in DB-L in [Pampalk, 2006b], but instead of having 22 genres, the number of genres was reduced to 13 to reduce genre overlaps. This collection was also used in [Pohle et al., 2007].

2.2.4.3. DB-CC. The DB-CC (Compilation Collection) is an in-house collection that was compiled with both audio and web based MIR research in mind. It consists of a selection of music compilations bought at local record stores. Such a compilation typically contains music by many different artists. Each compilation has a title related to a specific music style or genre, for example Rock'n'Roll, or Jazz Classics. In most cases, the titles of the compilations are used as genre labels, which is quite straightforward. Music styles are selected to be clearly distinct from each other to obtain a sound ground truth. The music contained in this collection covers a wide range of acoustical quality.

2.2.4.4. DB-NORM. The normalization collection (DB-NORM) is compiled from 100 tracks belonging to 10 different genres, each of those containing 10 tracks. Tracks in each genre were chosen to share typical elements of this genre, and being from a variety of different artists and recording conditions. For example, the *piano* genre covers differently sounding pianos and recording settings. At the same time, genres are chosen so that songs in different genres are unlikely to be considered as being similar. This collection was used in different ways. For example, it was used to quickly test the effect of modifications to algorithms, and to determine normalization factors for the combination of similarity measures.

[9] http://www.iua.upf.es/mtg
[10] http://ismir2004.ismir.net/genre_contest/index.htm

2.2.4.5. DB-HOMB. This collection is the one presented in [Homburg et al., 2005]. It consists of 10 second song excerpts collected from the Garageband web site. It contains excerpts from 1886 tracks by 1463 different artists. Due to the large number of artists, classification results with and without artist filtering are relatively close. The data is available to the research community, which allows comparison of results presented in different publications and by different authors. An advantage of the audio excerpt's shortness is that all publications use the full – and exactly the same – excerpts, which supports comparability of the obtained results.

DB-MS	classical (320/40), world (122/19), electronic (115/30), rock/pop (101/26), metal/punk (45/8), jazz/blues (26/5)
DB-NORM	african (10/10), dancefloor (10/10), hawaiian (10/7), mambo (10/8), metal (10/10), orchestral (10/10(?)), piano (10/9), rap (10/10), rocknroll (10/8), volksmusik (10/10)
DB-HOMB	rock (504/448), jazz (319/214), raphiphop (300/210), folk-country (222/177), alternative (145/121), blues (120/80), pop (116/106), electronic (113/97), funksoulrnb (47/39)
DB-L*	metal (477/14), electronica (452/28), rap (262/12), punk-rock (255/6), folk-rock (233/5), italian (142/5), jazz (133/9), celtic (132/5), a capella (112/4), bossa nova (72/4), acid jazz (68/4), blues (63/4), reggae (44/3)
DB-CC	rocknroll (200/127), volksmusik (197/150), punk (160/85), latino (125/46), reggae (108/27), dancefloor (81)

Table 2: Genre labels of the used music collections, and number of tracks / number of artists in each genre.

2.3. Starting Point: G1C

In this section, Pampalk's G1C music similarity algorithm [Pampalk, 2006a, Pampalk, 2006b] is described. It is well suited as a starting point for experiments (which will be presented in the next sections) and as evaluation baseline for several reasons. Its main component is based on MFCCs, which can be considered as state-of-the-art features for music similarity algorithms. Additional components based on Fluctuation Patterns (FPs) introduce descriptions of temporal developments in the signal that are not described by the MFCC component. The G1C algorithm scored first in the 2006 MIREX Audio Music Similarity and Retrieval task [Pampalk, 2006a]. An implementation is avail-

able[11]. Detailed descriptions of the algorithm's parts are given in the next sections.

2.3.1. MFCC Component

A music track is analysed by converting it into mono format at 22050 Hz sample rate, only keeping the central two minutes of the track. The signal is cut into frames of length 512 samples (with a hop size of 512 samples, $f_0 = 43,1$ Hz), and each frame is represented on the perceptually motivated mel scale by calculating the power FFT and subsequently applying a triangular-shaped mel filterbank with 36 bands. MFCCs are approximated by calculating the discrete cosine transform (DCT) on the logarithm of this representation, only keeping coefficients 1 to 19.

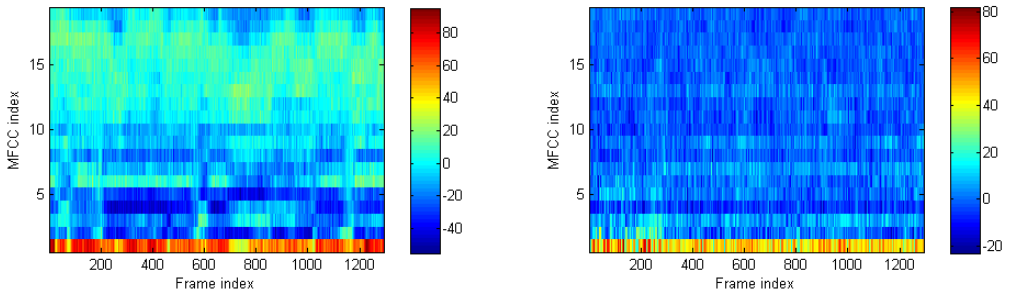

Figure 5: MFCCs of 30 second excerpts of a choir piece (left) and a pop song (right).

The feature data kept for a track consists of the mean and full covariance matrix of all MFCC frames (*Single Gaussian* model). Two tracks are compared by calculating the symmetric Kullback-Leibler (KL) distance (Equation 38) D_{KL} of the Gaussian models. The resulting distance is scaled to the range of the unit interval by

$$D_{\mathrm{G1}} = -\exp(-1/\texttt{fact} \cdot D_{\mathrm{KL}}) \tag{4}$$

For `fact` a value of 450 is used, based on empirical evaluations. Examples for MFCCs are given in Figure 5.

[11] http://www.pampalk.at/ma/download.html

2.3.2. Fluctuation Patterns

To also model aspects of time-dependent properties of the music such as rhythm, *Fluctuation Patterns* (FP) [Pampalk et al., 2002,Pampalk et al., 2003] are computed on the Mel-Frequency frames. To reduce the size of the resulting features, the 36 Mel bands are reduced to 12. Then, the frames of the whole two-minute excerpt are divided into blocks of 128 frames (i.e., 2.97 sec) length, and a hop size of 64 frames. For each of the blocks, the temporal development in each of the 12 Mel bands is analysed by a FFT. The lowest 31 of these fluctuation time scales are used as features for the block (i.e., fluctuation speeds from 0.34 Hz to 10.4 Hz are considered). The resulting amplitude spectrum is post processed to enhance the detection and ease comparison of perceived fluctuation strengths. This results in feature data of size $12 \cdot 30$ for each frame. The median of each of these values over all frames is the final Fluctuation Pattern for the track.

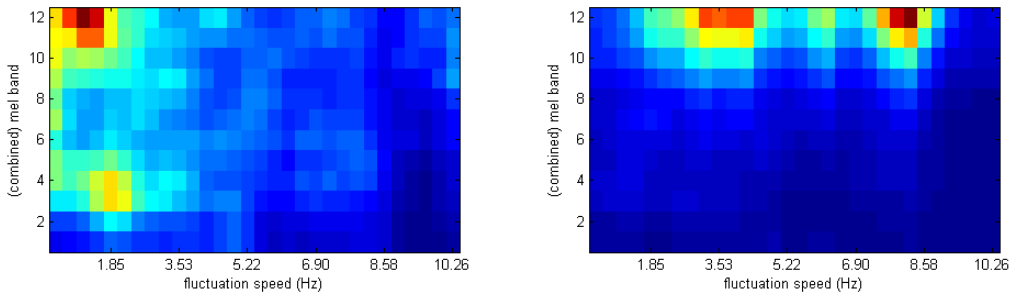

Figure 6: FPs of 30 second excerpts of a choir piece (left) and a pop song (right). It can be seen that due to summing up more bands in the higher frequencies, most energy is concentrated in rows 11 and 12. Colours are scaled independently for the two figures to improve visibility. Absolute values are smaller for the choir piece than for the pop song.

In G1C, FPs are used in three ways to determine the similarity of two tracks.

1. D_{FP}: The Euclidean distance between the FPs of the two tracks is computed.
2. D_{FPG} (Gravity): The values contained in the FP are summed up along the first dimension, and the centroid of the resulting vector is computed. This value is called *Gravity*, which obviously depends on the fluctuation strengths measured at different fluctuation speeds, and is assumed to be related to aspects of the rhythm such as overall tempo. The (absolute) difference between the Gravity values of the two tracks is computed.

3. D_{FPB} (Bass): The fluctuation values in the range 1 Hz to 10.4 Hz in the two lowest bands are summed up to obtain information about the bass frequencies. The difference between the Bass values of the two tracks is also computed.

Example images of FPs are given in Figure 6.

2.3.3. Combination.

The various basic distance measures D_{G1}, D_{FP}, D_{FPG}, and D_{FPB} are combined into one overall similarity measure by

$$\begin{aligned} D_{\mathrm{G1C}}(A,B) = \quad & 0.1 \cdot (D_{\mathrm{FP}}(A,B) - \mu_{\mathrm{FP}})/\sigma_{\mathrm{FP}} \\ + \quad & 0.1 \cdot (D_{\mathrm{FPB}}(A,B) - \mu_{\mathrm{FPB}})/\sigma_{\mathrm{FPB}} \\ + \quad & 0.1 \cdot (D_{\mathrm{FPG}}(A,B) - \mu_{\mathrm{FPG}})/\sigma_{\mathrm{FPG}} \\ + \quad & 0.7 \cdot (D_{\mathrm{G1}}(A,B) - \mu_{\mathrm{G1}})/\sigma_{\mathrm{G1}} \\ + \quad & c \end{aligned} \tag{5}$$

where μ_X and σ_X are the mean and standard deviation of distances measured with basic similarity measure X, empirically determined over a large number of track pairs. The tracks that are used to determine μ_X and σ_X are taken from one music collection that is not necessarily the same collection as the music collection on which the similarity measure is actually applied for retrieval. For such an independent music collection that is used to determine normalisation factors, in this thesis the term "reference collection" is used. μ_X and σ_X are determined once, and then hard-coded to be used for all similarity computations. The constant c ensures that the result is always positive. This non-negativity can be ensured as it is known that all $D_x(A,B)$ are non-negative. A constant c can be chosen empirically to be large enough for most distance computations, or it could be determined from Equation 6: $c = 0.1 \cdot (\mu_{\mathrm{FP}}/\sigma_{\mathrm{FP}} + \mu_{\mathrm{FPB}}/\sigma_{\mathrm{FPB}} + \mu_{\mathrm{FPG}}/\sigma_{\mathrm{FPG}}) + 0.7 \cdot \mu_{\mathrm{G1}}/\sigma_{\mathrm{G1}}$.

2.4. A Variant of G1C: G1Cmod

Before going on with presenting the experiments, here we briefly describe a modified variant of the G1C algorithm from 2007. Although the choice of modifications was not based on extensive systematic tests, our automated evaluations indicated that this algorithm has a (slightly) better similarity computation performance than the original G1C algorithm. This can be seen as

a motivation for the experiments presented in the next sections (i.e., to test whether further improvements can be found by evaluating a number of modifications of algorithm parts such as the distance computation function).

As described in [Pohle and Schnitzer, 2007], we have submitted a modified version of G1C to the 2007 MIREX Audio Similarity and Retrieval task. In the evaluation based on human judgments, this submission ranked first. No significant difference to six out of the eleven other submissions was measured by the Friedman test. The modifications applied to G1C were

- Based on the results presented in [Bosteels and Kerre, 2007b], D_{FP} was not calculated by Euclidean distance, but rather by cosine similarity (Equation 7) .
- The distance D_{G1} between Gaussians was not calculated by the KL Divergence (Equation 38, page 63) but rather a distance measure based on the Bhattacharyya coefficient was applied (Equation 46, page 64). Furthermore, the number of used MFCCs was increased to 25.
- The normalization factors μ_X and σ_X were determined dynamically for each seed track, based on all distances from a track A to all other tracks in the current collection. Consequently, when comparing two tracks A and B, there are four normalization factors for each basic distance measure X ($\mu_{X,A}$, $\sigma_{X,A}$ and $\mu_{X,B}$, $\sigma_{X,B}$). They are used to calculate the distance D_{G1Cmod} by applying Equation 6 twice (once with each of the normalization factors for track A and track B), and summing up the results. The resulting distances can be negative, which either has to be compensated by adding up a constant determined for the current collection, or it has to be taken into account when retrieving tracks, in which case the closest tracks may have a negative distance value associated.

In genre classification experiments performed before submitting the algorithm to MIREX'07, G1Cmod slightly outperformed G1C [Pohle and Schnitzer, 2007], while it remained unclear whether there is a significant difference.

2.5. Modification of Fluctuation Pattern based Distances

There is some indication that the modifications proposed in G1Cmod have a positive effect on the overall algorithm. One of the contributions of this thesis is the evaluation of a larger number of possible modifications of the various parts of the G1C algorithm, aiming to further improve upon G1C (or G1Cmod,

respectively). Thus, we start this section by presenting experimental results from evaluating possible alternative distance measures between Fluctuation Patterns. Previous work in this area is [Bosteels and Kerre, 2007b]. Also, possible alternatives to the descriptors FP Bass and FP Gravity are presented and evaluated. The next section (Section 2.6) is focused on modifications of the Gaussian component. Combining the various parts in a different manner than in the G1C algorithm is discussed and evaluated in Section 2.7. Finally, a modified algorithm is created based on the presented evaluations, and its relative performance to G1C and G1Cmod is tested in Section 2.8.

For comparing Fluctuation Patterns, in [Pampalk et al., 2005b,Pampalk, 2005, Pampalk, 2006b] Euclidean distance is applied. While Euclidean distance is straightforward and efficiently computed, there may be better ways from a conceptional point of view. In Euclidean distance computation, each component of the FPs is interpreted as one dimension in a high-dimensional space. The computed distance has the notion of a "line" between two points in this space, each represented by one fluctuation pattern. This concept has no immediately obvious link to the spatial organization of FPs (and thus may be inappropriate). Other distance measures may perform better. So in this section, a variety of different distance measures is evaluated. There are two major flavors. The first kind of distance measures is based only on comparing corresponding bins (or "entries") of the two FPs (as e.g., the Euclidan distance), here denoted *same-bin-measures*. The second kind of measures also takes into consideration the spatial proximity of FP bins, so that activations that are only few bins apart also can be matched. This kind of measure is called *inter-bin-measure*. The discussion starts with the first kind of measures.

Previous work regarding bin-to-bin distance (*same-bin-measures*) between FPs has been done by [Bosteels and Kerre, 2007b]. A FP is interpreted as a fuzzy set and a number of fuzzy similarity measures is evaluated on them. [Bosteels and Kerre, 2007b] use a collection of 128 tracks, and similarity refers to finding fragments of the same song. One of the conclusions is that applying the cosine similarity measure is preferable over using Euclidean distance. It remains unclear how the results scale to larger collections and similarity between songs. By taking these similarity measures as a subset of the experiments presented here, it is aimed to increase knowledge in this direction.

The evaluated same-bin distance measures between two FPs $F = \{f_i\}$ and $G = \{g_i\}$ are described in the next sections, largely following [Rubner et al., 2000, Bosteels and Kerre, 2007b].

2.5.1. Distance Measures

2.5.1.1. p-Norm, or Minkowski-form distance is defined as (e.g., [Rubner et al., 2000])

$$d_{L_p}(F,G) = \left(\sum_i |f_i - g_i|^p\right)^{\frac{1}{p}} \tag{6}$$

Here, L_1 (Manhattan- or taxicab distance), L_2 (Euclidean distance), L_4 and L_∞ are evaluated, with $L_\infty = \max(|f_1 - g_1| \ldots |f_n - g_n|)$. d_{L_4} is efficiently implemented as `d_L4 = sqrt(sqrt(sum(abs(f-g).^2.^2)))`. As preliminary experiments indicated that accuracy may increase with smaller integer p, also non-integer values smaller than one were tested for p ($\frac{1}{2}$ and $\frac{1}{3}$).

2.5.1.2. Cosine Similarity. The cosine similarity of two vectors measures the cosine between the two vectors. Considering the two FPs F and G as vectors, this takes the form (cf. [Bosteels and Kerre, 2007b])

$$d_{cos}(F,G) = \frac{\sum_i (f_i \cdot g_i)}{\sqrt{\sum_i f_i^2} \cdot \sqrt{\sum_i g_i^2}} \tag{7}$$

When both vectors have only non-negative entries, this measure has a range of [0..1].

2.5.1.3. Histogram Intersections. Various forms of histogram intersections are evaluated. The basic form is (cf. [Swain and Ballard, 1991, Rubner et al., 2000])

$$d_{\cap f,g}(F,G) = 1 - \frac{\sum_i \min(f_i, g_i)}{\sum_i g_i} \tag{8}$$

which results in the distance from F to G being zero if G is fully contained in F (i.e., if F is a superset of G). Obviously, the opposite point of view (the distance from G to F is zero if F is a subset of G) is given by:

$$d_{\cap g,f}(F,G) = d_{\cap f,g}(G,F), \tag{9}$$

We combine both as

$$d_{\cap symm}(F,G) = d_{\cap f,g}(F,G) + d_{\cap g,f}(F,G), \tag{10}$$

Based on this concept of set operations, an arbitrary number of further distance measures can be defined. When calculated from normalised FPs, FPs can be interpreted as fuzzy sets, and the corresponding similarity measures are denoted *fuzzy* audio similarity measures [Bosteels and Kerre, 2007b].

In the experiments, a number of such measures is evaluated both in a completely unnormalised version[12] and after each FP has been normalised by dividing each value by the FP's maximum value. Redefining cardinality as summing up the elements, $F \cap G := \{\min(f_i, g_i)\}$, and $F \cup G := \{\max(f_i, g_i)\}$ these take the form (evaluated for normalised, fuzzy FPs in [Bosteels and Kerre, 2007b], citing [Cross and Sudkamp, 2002]):

$$d_5(F,G) = 1 - \frac{|F \cap G|}{|F \cup G|} \tag{11}$$

$$d_6(F,G) = 1 - \frac{|F \cap G|}{\sqrt{|F| \cdot |G|}} \tag{12}$$

$$d_7(F,G) = 1 - \frac{2\,|F \cap G|}{|F| + |G|} \tag{13}$$

$$d_8(F,G) = 1 - \frac{|F \cap G|}{\min(|F|, |G|)} \tag{14}$$

$$d_9(F,G) = 1 - \frac{|F \cap G|}{\max(|F|, |G|)} \tag{15}$$

$$d_{10}(F,G) = 1 - \frac{\min(|F|, |G|)}{|F \cup G|} \tag{16}$$

$$d_{11}(F,G) = 1 - \frac{\max(|F|, |G|)}{|F \cup G|} \tag{17}$$

$$d_{12}(F,G) = 1 - \frac{\min(|F|, |G|)}{\max(|F|, |G|)} \tag{18}$$

[12] for the measures considered in this section, this is equivalent to scaling all values by the same factor $f = 1/v_{\max}$ with $v_{\max}$ being the largest value appearing over all FPs of the whole music collection

2.5.1.4. Correlation Coefficient. To determine the correlation between the values in two FPs, Pearson's linear correlation coefficient is calculated, which is defined as (cf. [Seyerlehner and Schnitzer, 2007]):

$$\rho_{F,G} = \frac{\sigma_{FG}^2}{\sigma_F \sigma_G} = \frac{\sum_i (f_i - \overline{f})(g_i - \overline{g})}{\sqrt{\sum_i (f_i - \overline{f})^2} \sqrt{\sum_i (g_i - \overline{g})^2}} \tag{19}$$

with $\overline{f}$ and $\overline{g}$ being the mean of all values in FP F and G, respectively.

2.5.1.5. Jensen-Shannon Distance. For discrete distributions, the Kullback-Leibler divergence is defined as (e.g., [Diaconis and Zabell, 1982])

$$d_{\mathrm{KL}}(F, G) = \sum_i f_i \log \frac{f_i}{g_i} \tag{20}$$

Following [Rubner et al., 2000], for the experiments the numerically more stable Jeffrey distance (Jensen-Shannon distance) is used:

$$d_{\mathrm{JS}}(F, G) = \sum_i \left(f_i \log \frac{f_i}{m_i} + g_i \log \frac{g_i}{m_i} \right) \tag{21}$$

with

$$m_i = \frac{f_i + g_i}{2} \tag{22}$$

In the implementation, for numerical stability, those i with both f_i and g_i being zero (or below a threshold of `1e-10`) are left out, as such i do not increase the distance between F and G.

An alternative way to compute Equation 21 is given by [Lin, 1991]

$$d_{\mathrm{JS}} = H(m) - \frac{1}{2}H(f) - \frac{1}{2}H(g) \tag{23}$$

where $H(x)$ is the Shannon entropy

$$H(x) = -\sum_i x_i \cdot \log x_i \tag{24}$$

$\sqrt{d_{\mathrm{JS}}}$ is a metric (e.g., [Briët and Harremoës, 2009]).

2.5.1.6. Chi-Square Statistics The χ^2 distance between two discrete probability distributions is defined as

$$d_{\chi^2}(F, M) = \sum_i \frac{(f_i - m_i)^2}{m_i} \tag{25}$$

Here again M is set to $m_i = \frac{f_i + g_i}{2}$ as above [Rubner et al., 2000]. As $|f_i - m_i| = |g_i - m_i|$, this measure is symmetric. Again, those i with both f_i and g_i being (close to) zero are left out during computation.

2.5.1.7. Bhattacharyya Coefficient. On discrete probability distributions, the sample estimate of the Bhattacharyya coefficient is given by (cf. [Comaniciu et al., 2003], [Kailath, 1967])

$$\rho(F, G) = \sum_i \sqrt{f_i \cdot g_i} \tag{26}$$

Although it is defined on probability distributions (i.e., both F and G sum to one), in the experiments all of unnormalised FPs, FPs divided by the maximum, and FPs normalised to sum to one were evaluated.

2.5.1.8. Hellinger Distance Hellinger Distance is given as [Diaconis and Zabell, 1982]

$$d_H(F, G) = \sqrt{\sum_i \left(\sqrt{f_i} - \sqrt{g_i}\right)^2} \tag{27}$$

As Hellinger distance is related to the Bhattacharyya coefficient by $d_h(F, G) = \sqrt{2 - 2 \cdot \rho(F, G)}$ (cf. [Jebara and Kondor, 2003]) for probability distributions, it produces the same ranking of tracks as the Bhattacharyya coefficient in the case when F and G are normalized to sum to one.

2.5.1.9. Mutual Information For two (discrete) random variables X and Y, Mutual Information is defined as [Kraskov et al., 2003]

$$\begin{aligned} I(X, Y) &= \sum_{i,j=1}^{n} p_{ij} \log \frac{p_{ij}}{p_i(X) p_i(Y)} \\ &= H(X) + H(Y) - H(X, Y) \end{aligned} \tag{28}$$

with $H(X)$ and $H(Y)$ denoting the entropy of X and Y (Shannon entropy, Equation 24), and $H(X,Y)$ being the joint entropy of X and Y. Based on this, it is possible to define a metric as follows (cf.[13] [Kraskov et al., 2003]):

$$\begin{aligned} d_I(X,Y) &= H(X,Y) - I(X,Y) \\ &= H(X,Y) - (H(X) + H(Y) - H(X,Y)) \\ &= 2H(X,Y) - H(X) - H(Y) \end{aligned} \tag{29}$$

This metric can be modified to be not biased by the dimensionality n of the data and to yield values in the unit interval [Kraskov et al., 2003]:

$$D_I(X,Y) = \frac{d_I(X,Y)}{H(X,Y)} \tag{30}$$

To calculate D_I on fluctuation patterns, X and Y here are defined as histograms over all values appearing in Fluctuation Patterns F and G. The joint entropy $H(X,Y)$ is calculated from the joint histogram of pairs of values appearing at corresponding positions of the two Fluctuation Patterns. The number k of histogram bins in X and Y (and consequently the number $k \cdot k$ of histogram bins of the joint histogram) is determined by Sturge's rule ([Legg et al., 2007]), which is used to determine the width w of each histogram bin:

$$w = \frac{r}{1 + \log_2(n)}, \tag{31}$$

where r is the data range and n is the number of input points ($n = 30, 20$, and $20 \cdot 30$ in the case of the first dimension, second dimension and full Fluctuation Pattern, respectively). As w is calculated separately for each Fluctuation Pattern (on the range r from 0 to the maximum value appearing), normalization does not play a role for this distance measure, as in each case the histogram bins are adapted to the full range of values appearing in the given Fluctuation Pattern, and thus the same histogram results regardless of normalization.

2.5.1.10. Inter-Bin Distance Measures. The distance measures discussed so far only compare the corresponding bins of the two FPs. However, if the two FPs have peaks at different bins that are close to each other, then they might be considered more similar than having peaks that are far away. To some extent, this is taken into account by blurring during computation of the FP. To evaluate if it is beneficial to take such possible inter-bin associations into account, also the quadratic-form distance, Mahalanobis distance and Earth Mover's Distance are computed.

[13] also cf. http://en.wikipedia.org/wiki/Mutual_Information

2.5.1.11. Quadratic-Form Distance. The quadratic-form distance is defined as [Niblack et al., 1993]

$$d_A(F,G) = \sqrt{(f-g)^T A (f-g)} \tag{32}$$

f and g are vectors that contain all bins (f_i and g_i) of the two FPs F and G, and A is a matrix containing at A_{ij} the *similarity* between bins i and j. A is computed from a ground-truth distance d_{ij} between bins i and j by $a_{ij} = 1 - d_{ij}/d_{\max}$ ([Niblack et al., 1993]), with $d_{\max}$ being the largest value over all d_{ij} . This definition of A results in d_A being a metric [Rubner et al., 2000].

For applying this distance measure to FPs, the ground-truth distances d_{ij} have to be defined. Here, *ground truth* refers to the impact a given shift in frequency and fluctuation speed has on the perceived similarity. In this context, it seems reasonable to assume a small impact for bins that are close neighbors. For example such a small shift may occur when comparing a piece of music to a slightly faster, or slower, version of itself.

In general, these ground truth distances d_{ij} would have to be defined by listening tests (which, however, may be difficult to design). Here, as a rather simple and straightforward way of defining distances between different frequency bands and between different fluctuation speeds, a linearly increasing cost is assumed (transformation from slowest to fastest fluctuation speed has a cost of 1, and transformation from lowest to highest frequency has also a cost of 1 associated). When it comes to combining both these dimensions into one common measure things get even more arguable (i.e., how does an increase in frequency relate to an increase in speed?). As a solution, for the experiments the city block distance was chosen as it reflects the concatenation of the transformation in speed and a consecutive transformation in frequency.

The FPs used in the experiments have a size of 20×30, so A gets very large (360.000 elements), which results in a slow computation. By resampling the FPs dimensions to 8×20, the number of elements in A is reduced to 25.600. With this size, computation is much faster. Resampling is done by using 0.5 values on either side of the current sample.

2.5.1.12. Earth Mover's Distance. The Earth Mover's Distance (EMD) [Rubner et al., 2000] formalizes the concept to model the distance of two distributions (e.g., considered describing the topology of "piles of earth") by the amount of "work" necessary to transform one distribution into the other.

Here, each FP is assumed to represent a two-dimensional description of such piles of earth, comparable to a topological map. For defining the amount of "work", the same inter-bin distance definition as in the quadratic-form distance evaluation is used. For calculating[14] the EMD, also the reduced-size version of the FPs is used for reasons of computational time.

2.5.2. Results of (full) FP Distance Measures.

The multiple comparison test for the discussed distance measures on *DB_MS* is given in Figure 7, and the corresponding results for *DB_102a13g* are given in Figure 8. The way the distance between FPs is computed in the G1C algorithm is taken as baseline algorithm we want to improve upon (i.e., Euclidean distance measure on unnormalised FPs of size $12 \cdot 30$). As discussed in Section 2.2.3, the baseline algorithm is given in blue colour, and the corresponding significance bounds are marked as gray vertical lines. On both collections l2 performs better when calculated from FPs of size $20 \cdot 30$ (downsampled from FPs of size $36 \cdot 30$ by linear interpolation) instead of the way it is calculated in $G1C$, which is taken as indication that this representation is better suited. As a consequence, all other distance measures (except the Quadratic-Form Distance and Earth Mover's Distance) were calculated from this representation of size $20 \cdot 30$.

Although some of the measures are defined on probability distributions, each distance measure was evaluated on unnormalised data (i.e., the FPs as computed, denoted by *normno*), each FP normalised to have a maximum value of one (denoted by *normmax*), and each FP normalised to sum to one (denoted by *normsum*, which is the normalization approach that is conceptually closest to a probability distribution). Each algorithm was tested as described in Section 2.2.3, considering the 20 closest neighbors after artist filtering.

In both figures, d_{12}normsum represents some kind of worst case scenario, because it produces a constant value of zero (as can be seen from the definition, $|F|$ and $|G|$ are both 1). Cosine similarity, correlation and the mutual information measure are independent from normalization. Distance measures *EMD* and *Quadratic Form Distance* were only calculated on *DB_MS*. As they showed to be not better than the baseline they were not computed on the other collection due to computation time.

[14] A Matlab wrapper was used to run the C files written by Y. Rubner, http://vision.stanford.edu/~rubner/

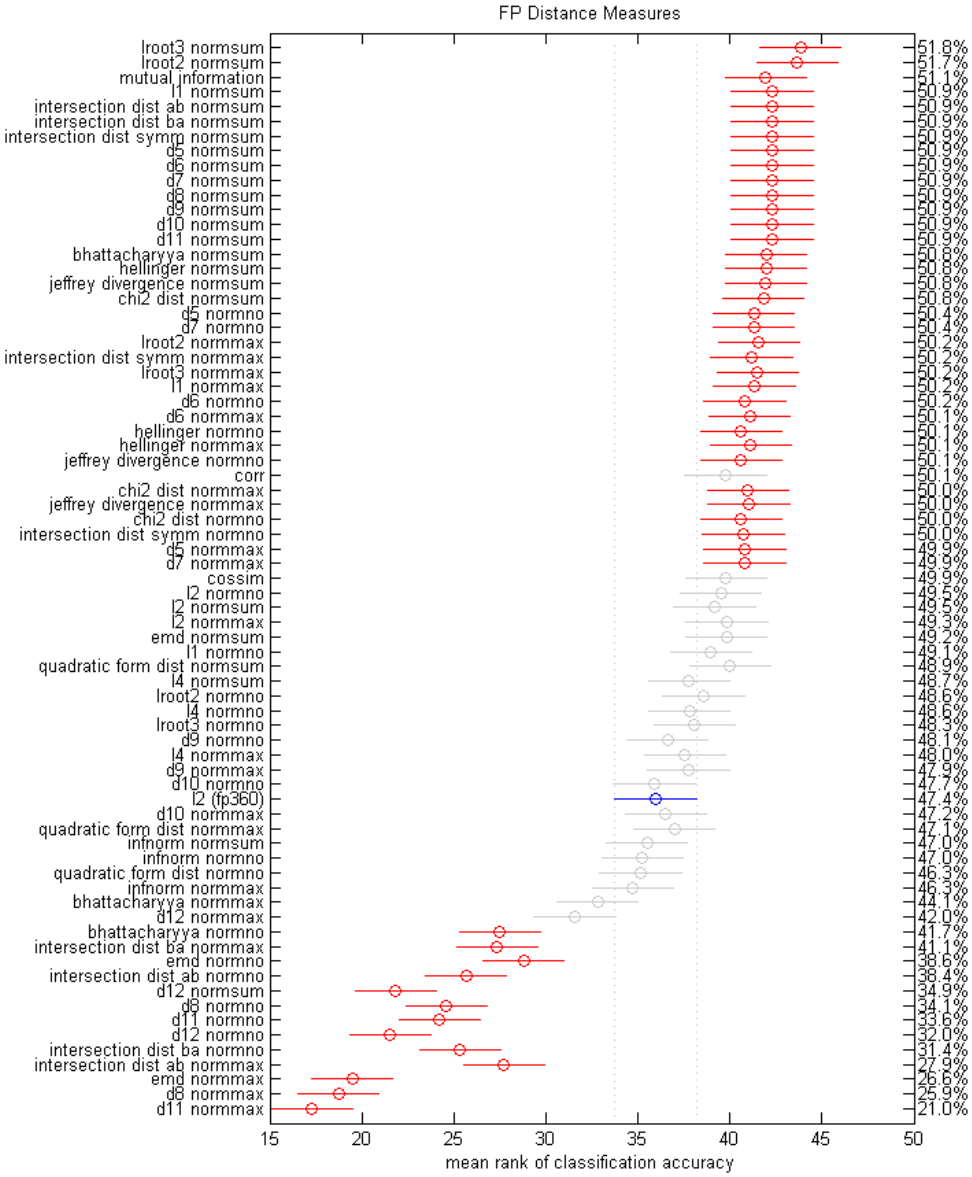

Figure 7: Multiple comparison of the examined FP distance measures. Leave one out genre classification experiments, ISMIR set, 20 first neighbors considered, artist filtering. Sorted from top to bottom by average (probabilistic) classification accuracy.

As can be seen, a number of the evaluated measures perform better than the chosen baseline (Euclidean distance measure on unnormalised FPs of size $12 \cdot 30$) in these experiments. Cosine similarity, which is recommended in [Bosteels and Kerre, 2007b], performs better than the baseline on *DB_MS*, but is below the baseline on *DB_103a13g*, i.e., cosine similarity does not always outperform Euclidean distance. It is interesting to note that all measures that outperform the baseline on *DB_102a13g* (Figure 8) also outperform the baseline on *DB_MS* (Figure 7). As both collections have many differences (different sizes, number and sizes of genres and different music styles), this can be seen as an indication that these measures generally outperform the baseline measure. Furthermore, the seven measures top-ranked on *DB_102a13g* (which are *significantly* better than the baseline on this collection) are also *significantly* better than the baseline on *DB_MS*. All of these are calculated from unnormalised FPs, which may be seen as an indication that in general, normalizing FPs is not beneficial for distance computations. All of these best-performing measures seem suited for using them as a replacement of the original measure. Considering the average (increase of) classification accuracy of each measure on the two collections, d_5 and d_7 seem the best candidates, particularly as they have the nice property of producing values in the unit interval, despite the fact that they are computed on unnormalised FPs. Also, they seem meaningful from a conceptual point of view.

2.5.3. Tempo Component from FP

After having evaluated a number of alternative distance measures between FPs, we here go on by extending the experiments to the two additional descriptors based on FPs that are used in the G1C and G1C_mod algorithms, respectively: *Gravity* (FPG) and *Bass* (FPB). G1C uses Gravity (FPG) as a tempo-related feature. In this section, experiments are presented that help assessing whether FPG can be replaced by a different way to find songs with similar perceived tempo. Without doubt, perceived tempo is an important aspect of music similarity. The Bass (FPB) descriptor will be discussed in the next section.

In [Seyerlehner and Schnitzer, 2007], an approach is described to use the rhythm structure of a piece to determine its tempo. The authors use the sum of the first dimension of FPs as an indicator of the song's rhythmic structure, that is a vector containing the sums over all frequencies for each periodicity (see Figure 9). The vector containing the sum of the first dimension of FPs is denoted *FPR* here. In [Seyerlehner and Schnitzer, 2007], this feature is used in conjunction with a nearest neighbor approach to determine the song's tempo

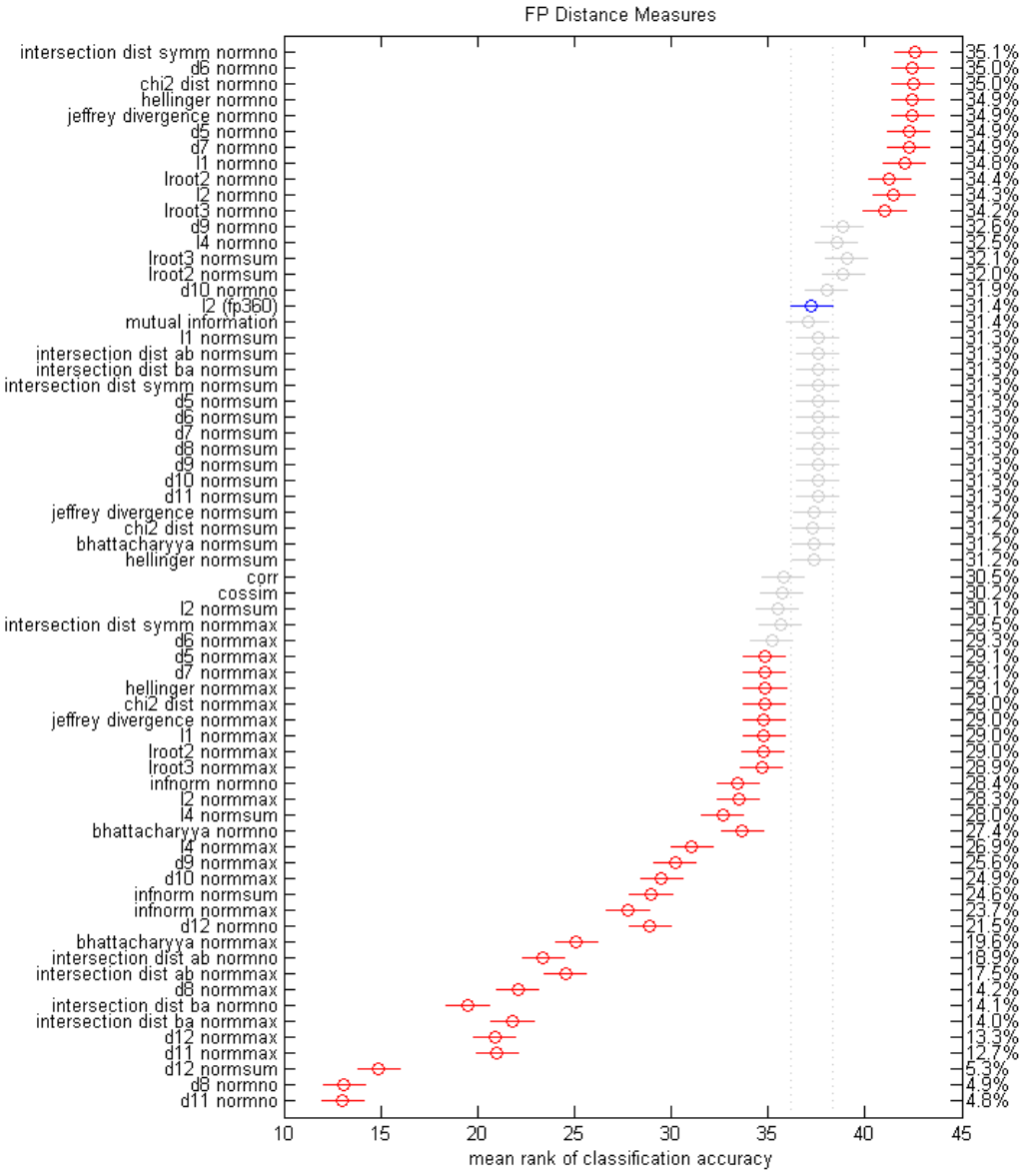

Figure 8: Same evaluation as in Figure 7 but on 102a13g collection.

value in beats per minute (*bpm*). The nearest neighbors are determined by correlation.

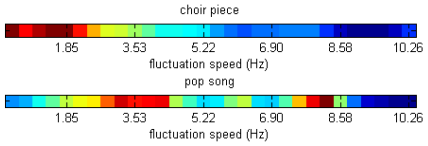

Figure 9: FPR of the choir piece and the pop song from Figure 2.3.1. Note that the colours are scaled independently for each figure to improve visibility.

To assess if the use of *Gravity* in *G1C* can be replaced by a better-performing estimation of the similarity of the perceived tempo, it is compared against the algorithms that result when comparing FPRs with the distance measures discussed above. However, ground truth is not given by genre labels this time, but rather by human-annotated tempo information.

As in the research presented here, the aim is not to find a good estimation of the seed song's tempo in bpm, but rather to find tracks that match the seed song's perceived tempo, evaluation is modified. Evaluation is based on a leave one out nearest neighbor scheme as in the previous experiments, but no explicit tempo estimation for the seed song is carried out (i.e., no explicit *bpm* value is calculated). Rather the seed track's true tempo is compared against the nearest neighbor's true tempi (as annotated). Goodness of match s between the seed track's tempo t and a nearest neighbor's tempo $\hat{t}$ is calculated by (cf. [Seyerlehner and Schnitzer, 2007])

$$s = \frac{\max(t, \hat{t})}{\min(t, \hat{t})} \tag{33}$$

For evaluation, *DB_MS* was annotated by tap-along tempo by one musically trained individual. Those tracks that do not have a recognizable (or various or changing tempi) were marked and left out during evaluation. Also, it was marked if tap-along tempo can be considered arguable, and if so, which factor would be acceptable otherwise (double, half, both of these, three times as fast or one third as fast). During evaluation, these were also considered as correct matches by taking the minimum s between the tracks in question and the tapping tempo alternatives.

During evaluation, 20 nearest neighbors were considered. An artist filter was not applied, as tempo is assumed to be independent from the artist, which can easily be seen from the notion that suggesting only tracks by the same artist

might even degrade performance. While FPG was calculated in a manner as in G1C, FPRs were derived from the FPs of size $20 \cdot 30$ (downsampled from FPs of size $36 \cdot 30$), as this is the way FPs are calculated in Section 2.5.2. However, it is assumed that the alternative way of calculating FPs (combining upper frequency bands as in G1C) would not make a big difference as the number of fluctuation time scales is the same (i.e., 30).

In Figure 10, it can be seen that FPG surprisingly does not perform significantly better than choosing nearest neighbors randomly on DB-MS in our evaluation setting (but it does on the second used test collection, as can be seen in Figure 11). A possible explanation is that FPG seems not only to measure the actual *tempo*, but also the relation between the amounts of slower and faster metrical levels present in the piece. The latter is an aspect of rhythm that is not directly taken into account in our evaluation setting.

The best-performing algorithm in terms of average score s is L2 normalised to a maximum of 1.0. However, Correlation is the algorithm that has a higher score for more seed songs than any other algorithm (i.e., it is the algorithm positioned rightmost). The absolute difference measured by the tempo score s seems rather small.

To gain more insight which algorithm can be considered preferable, the algorithms are also run on a different collection, which was used in the ISMIR 2004 tempo contest[15] [Gouyon et al., 2006]. It is called *Songs* set and contains 465 audio excerpts of 20 seconds length with tempo annotations created by a professional musician.

The corresponding results are given in Figure 11. It turns out that Correlation is the best performing algorithm among those evaluated, both by means of s and number of seeds with best score. Considering all algorithms that are within the significance bounds in this experiment (all top-ranked algorithms up to Infnorm normalised to sum to 1.0), it turns out that all of them are also within the significance bound of the best-performing algorithm in Figure 10.

In both experiments, absolute figures of s seem rather low – it indicates that even for the best-performing algorithms, the annotated tempi of the 20 retrieved tracks deviated from the seed track's annotated tempo by about $\frac{1}{3}$ on average. Maybe this number would decrease if the number of songs within a collection were greater.

[15] http://mtg.upf.edu/ismir2004/contest/tempoContest/

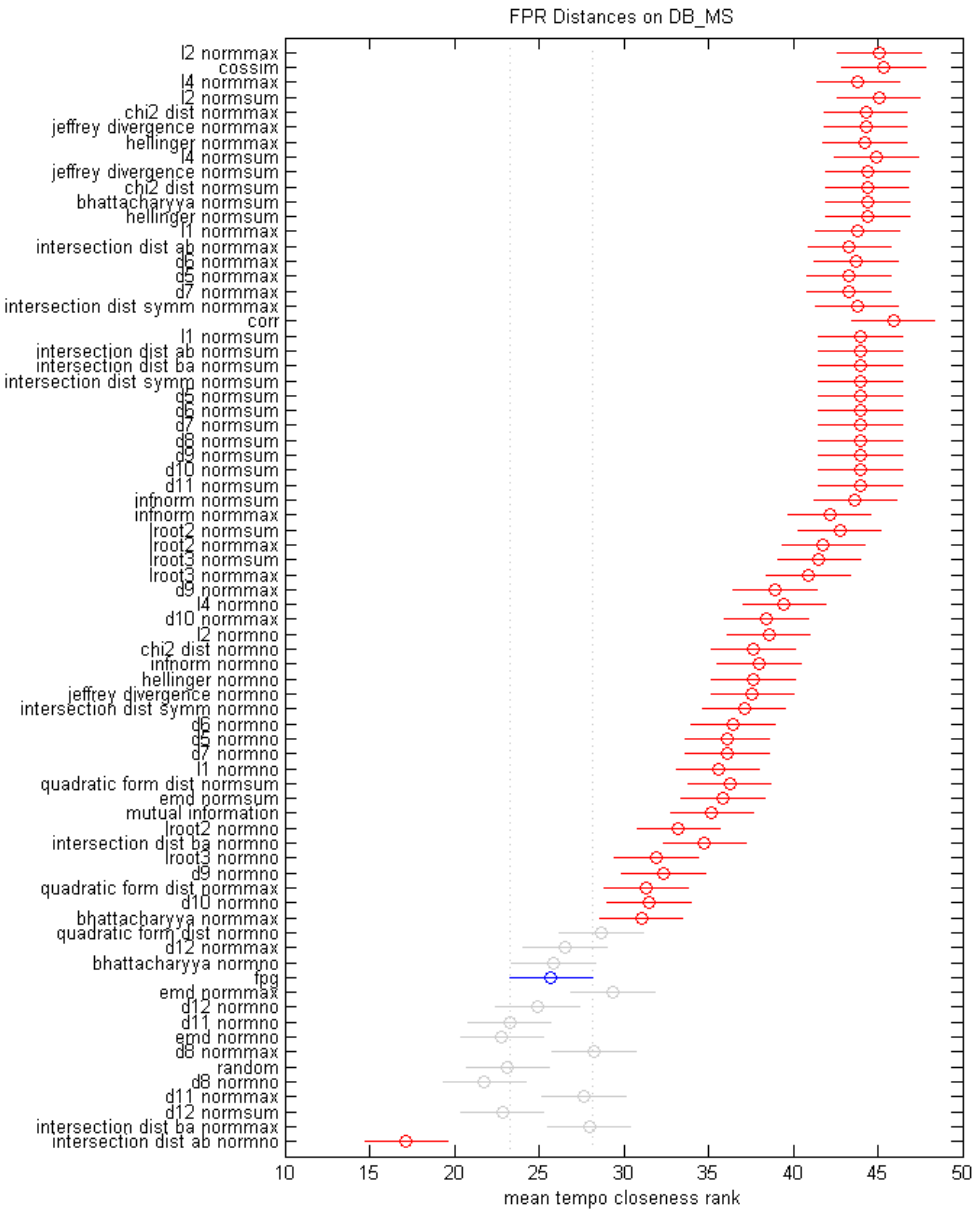

Figure 10: Tempo evaluation on DB_MS for 20NN. Values sorted according to average raw tempo matching score s. Scores for Random: 1.47, FPG: 144.5, Correlation: 135.6, L2 Normmax: 135.1.

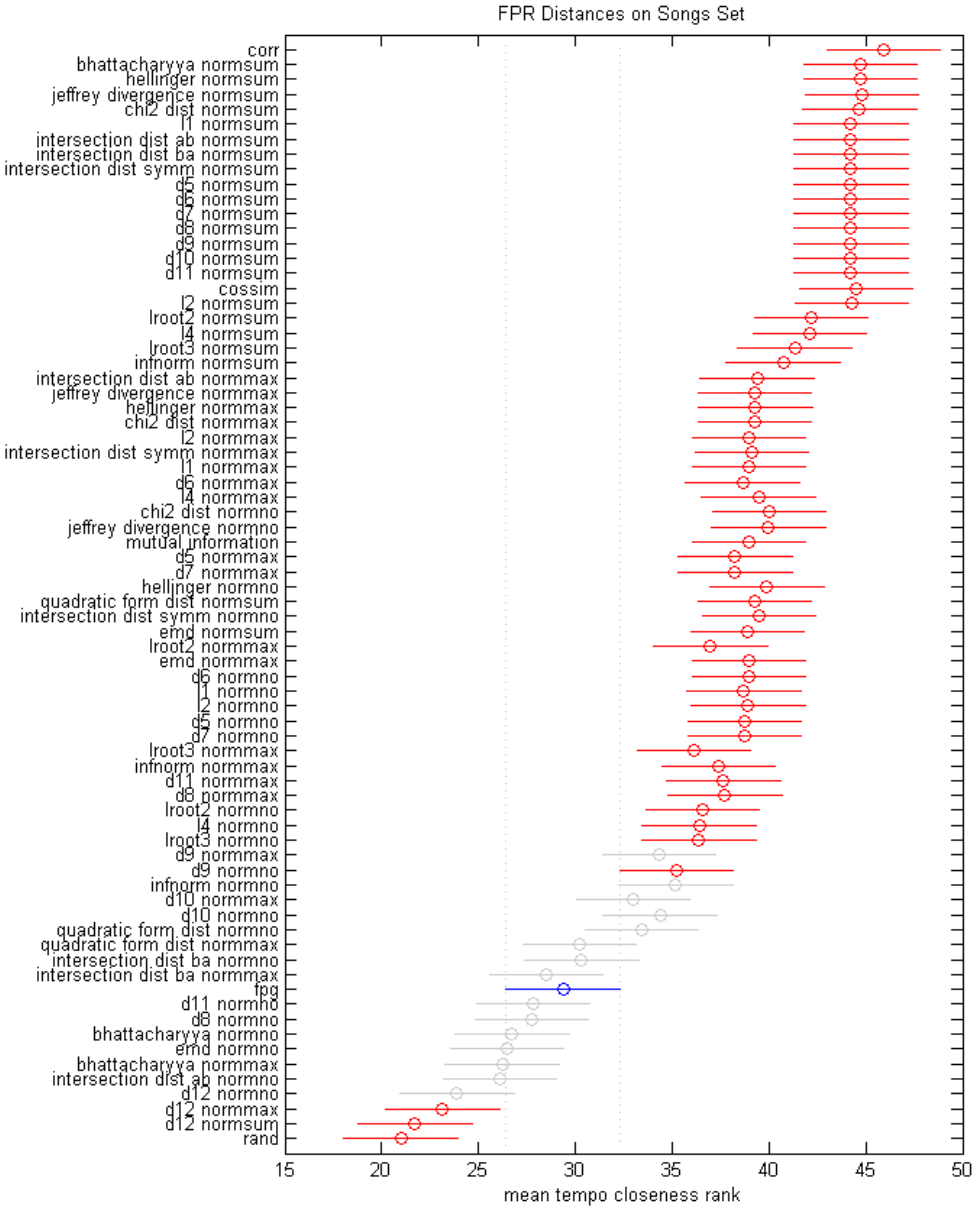

Figure 11: Tempo evaluation on Songs Set for 20NN. Values sorted according to average raw tempo matching score s. Scores for Random: 1.47, FPG: 1.41, Correlation: 1.36.

Surprisingly, it turns out that also genre classification accuracy increases when replacing FPF with Correlation between FPRs. On *DB_MS*, for rank 20, accuracy after artist filtering increases from 42.84% for FPG to 48.19%.

As a conclusion from this tempo evaluation (or FPG evaluation, respectively), it can be stated that replacing the FPG part of G1C with Correlation seems promising, as it brings a significant improvement in finding songs with similar perceived tempo, and additionally an increase in genre classification accuracy. Finally, it should be noted that here only FP based features are evaluated, and only 30 fluctuation time scales. There may be improvements when using different representations that could turn out to be better suited (candidates for this might be, e.g., 60 fluctuation speeds [Seyerlehner and Schnitzer, 2007], or autocorrelation as in [Ellis, 2006]).

2.5.4. Spectrum Summarization: FP Based Timbre (FPT)

In Section 2.5.3, the sum along the first dimension of the FP was taken as a basis for tempo estimation, which is used to replace FPG. In this section, it is examined if the second feature in G1C that is based on FPs – *Bass* (FPB) – might also be replaced by a conceptually similar algorithm, taking the sum of the FP in the *second* dimension. This section concludes the experiments with respect to Fluctuation Pattern based distance measures.

Bass (FPB) is calculated by taking the sum of the values in the lower FP bands. From a conceptual point of view, it may make sense to replace this by the sum of the values in the second dimension of the FP, i.e., the sum over all periodicities in the FP for each frequency band. While the feature data of FPB is one scalar value per track, the resulting feature values in this case is equal to the number of frequency bands in the FP (see Figure 12). This feature is denoted *FPT* here, *T* for *timbre*. The distance of FPTs can be determined by one of the distance measures described in Section 2.5.

This approach can be motivated by comparing the aspects it presumably describes. While MFCCs capture the frequency spectrum of individual frames, FPTs may combine frequencies into a common feature vector that are not activated together at the time level of one individual frame. Furthermore, MFCCs and FPTs work on different time scales, as FPTs are only based on fluctuations up to a certain fluctuation time scale, i.e., they do not consider the time scale of individual frames.

Here, it is evaluated how FPTs perform when compared by the different distance measures, and how they compare to FPB (compared by the absolute

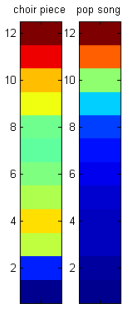

Figure 12: FPT of the choir piece and the pop song from Figure 2.3.1. Note that the colours are scaled independently for each figure to improve visibility, and that in the actual experiments presented here, there are 36 frequency bands instead of 12, as the bands are not combined.

difference). As baseline, FPTs compared by L2 distance, calculated on FPs of size $12 \cdot 30$, i.e., 360-element FPS, are taken. In the other cases, FPTs are calculated on 36 frequency bands. Evaluation is done based on genre labels, accepting the common assumption that genre labels are suited as an indicator for timbre similarity. Detailed result figures are given in Figures 13 and 14. Comparing these, it gets obvious that the top-ranking 13 algorithms are the same on both collections (although in a different order). When choosing as a baseline Euclidean distance on the second dimension of FPs (calculated as in G1C), all these algorithms are significantly better than the baseline. While the results seem somewhat inconclusive which of these to choose, it seems that the χ^2 distance is a good choice, or D_5 (or D_6) if it is important that the range of resulting values is in the unit interval.

When comparing these results to the performance of FPB in these experiments (which may not be an appropriate comparison as FPB is designed to focus on just one aspect, bass) it becomes apparent that most examined algorithms clearly outperform FPB in the genre classification task. While this performance boost may be beneficial for calculating timbre similarity, one has to keep in mind that there likely is a high correlation between FP, FPR and FPT based measures. Taking these into account when combining these measures might improve overall performance.

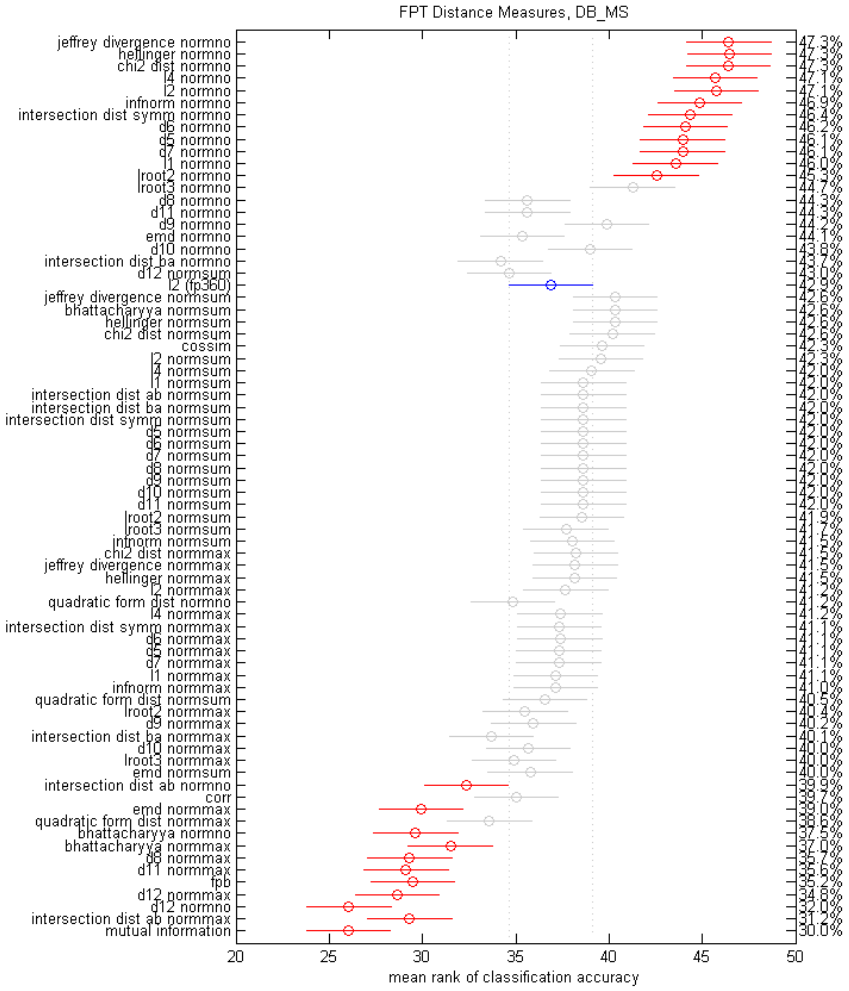

Figure 13: DB-MS: FPT distance measures.

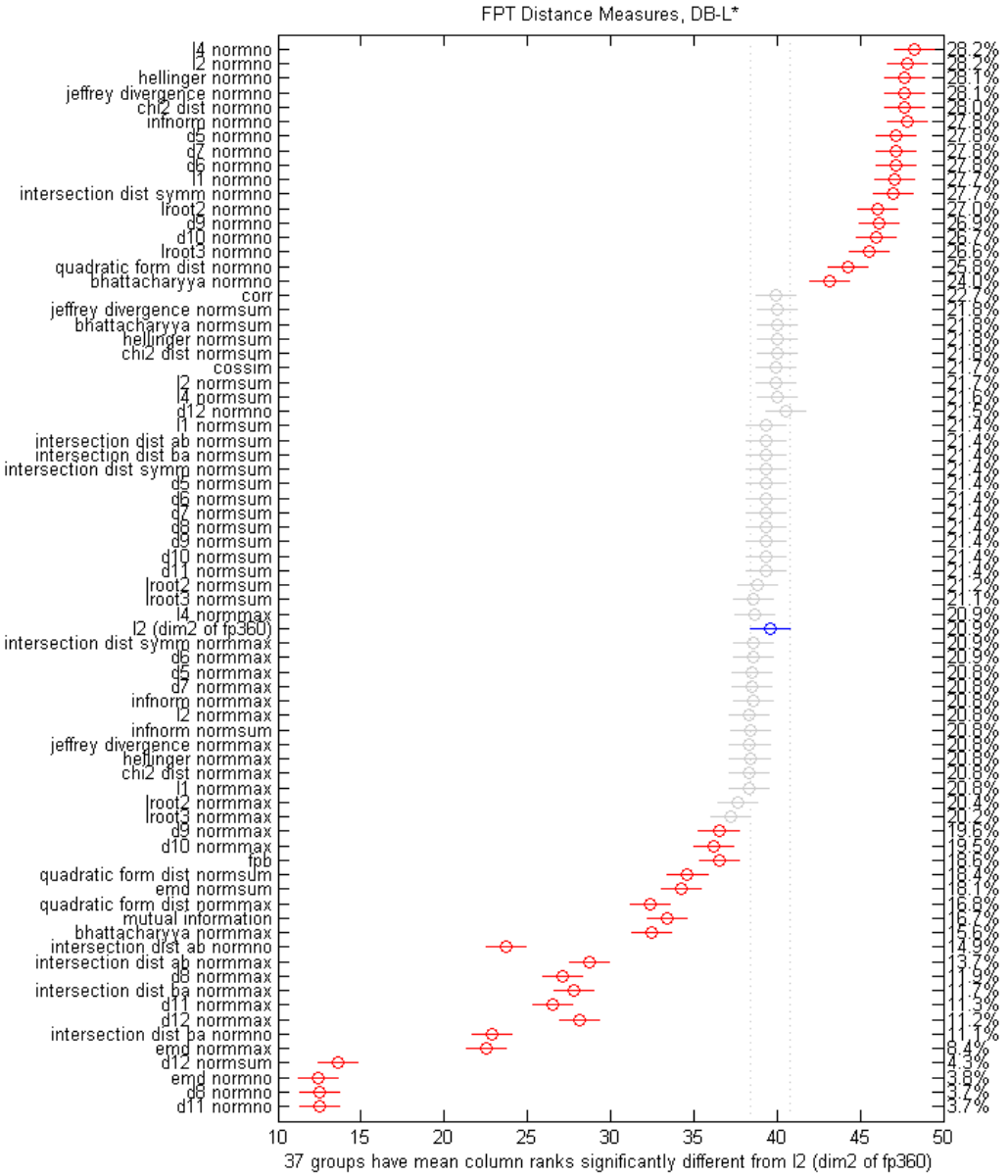

Figure 14: DB-L*: FPT distance measures.

2.6. Modification of the Gaussian Component

In the previous section alternative ways of using FP data were evaluated. This section is devoted to the basic similarity measure that has the highest weight in G1C, the MFCC based component. The experiments take two directions: First, it is evaluated if the quality of the similarity computation can be increased by adding different audio features. Second, as for FPs, alternative distance measures are evaluated. The runtimes of there distance measures is discussed. Also, the impact of some simple modifications of this part of the algorithm is assessed.

While there are many possible ways to compare the MFCC frames that are calculated from the music signal, the experiments presented here will focus on the so-called *bag-of-frames* (BOF) approach. The BOF approach can be considered the most common one. The temporal order of the MFCC frames is discarded, and only the overall statistical characteristics of the frames are kept as feature data. Two songs are compared by comparing their statistical models.

While there have been approaches that also take into consideration the temporal order of the frames (most notably, Hidden Markov Models, HMMs), they have not been generally accepted to outperform the BOF approach ([Aucouturier and Pachet, 2004, Aucouturier, 2006]).

To build a statistical model of the frames, usually Gaussian Mixture Models (GMMs) or Single Gaussians are applied. As previous work indicates that using Single Gaussians performs comparably to using GMMs [Pampalk, 2006b, Levy and Sandler, 2006], here only the Single Gaussian case is considered. A possible alternative to Gaussian modeling is using codebook approaches such as Vector Quantization (VQ), however in [Aucouturier, 2006] such experiments did not show an improvement of such approaches over GMM modeling with optimal settings. While recent research has indicated that VQ based approaches can perform comparably to similarity measures based on Single Gaussians [Seyerlehner et al., 2008], still for VQ algorithms a codebook has to be trained. Thus they seem less flexible than Single Gaussian approaches when it comes to generalizing to unseen music. Thus, VQ approaches are not considered here.

Based on the thoughts above, the experiments have two major directions.

1. In a like manner as in Section 2.5 it is evaluated if the commonly used way to calculate the similarity between Single Gaussians can be replaced by a different similarity measure that performs better. This approach has

been followed before for (multiple-gaussians) GMMs in [Jensen et al., 2007]. Here a larger number of possible measures is evaluated.

2. As MFCCs do not capture certain aspects of the musical signal (short-time developments and fine-grained shape of the spectrum), it is evaluated if additional features can be appended to each MFCC vector before creating the Gaussian model, to obtain a better-performing overall measure.

As changes in each of these directions may have an influence on the performance of the other (e.g., the performance of a similarity measure may depend on the type of audio features), these are evaluated together. Next, the additional audio features are discussed, followed by a description of the evaluated distance measures.

2.6.1. Additional Audio Features

The features based on MFCCs are tailored to capture the overall distribution of sounds, and the features based on FPs are designed to describe the sound's periodic mid- to long term changes. Both approaches use a relatively broad representation of the spectrum (e.g., these descriptors are based on 36 mel bands, which are combined into the equivalent of 25 bands in the case of MFCCs, or to 12 bands in case of FPs). In this representation, information about the shape of the spectrum on a finer scale is lost. Most importantly, no distinction is made if only a few sinusoidal components in the band are clearly activated and the other frequencies are zero, or if the band contains noise (assumed that in both scenarios there is the same signal power in the considered band). From a perceptual point of view, this makes a clear difference. Also, information about short-time changes between consecutive frames is discarded.

2.6.1.1. Differential features from the literature. To incorporate such information into the audio similarity measure, additional features have been suggested in the literature. Here, three of them are evaluated: *Delta Coefficients / Acceleration Coefficients* [Aucouturier and Pachet, 2004], and *Spectral Contrast*.

Delta Coefficients are calculated by [Aucouturier and Pachet, 2004] as

$$d_t = \frac{\sum_{\theta=1}^{\Theta} \theta(c_{t+\theta} - c_{t-\theta})}{2\sum_{\theta=1}^{\Theta} \theta^2} \tag{34}$$

where d_t is the resulting delta coefficient for MFC coefficient c at time t, and Θ is the size of the window on which the delta coefficient is calculated. *Acceleration Coefficient*s are calculated by applying this formula twice (i.e., the formula is applied on the Delta Coefficients to calculate Acceleration Coefficients). In this work, Θ was set to 1 (and the normalization factor of 2 was omitted based on the notion that KL divergence is invariant to linear scaling factors). There are as many Delta / Acceleration Coefficients as MFCCs. They are appended to each MFCC vector, yielding a vector of twice the original dimensionality.

Spectral Contrast [Jiang et al., 2002] measures the logarithm of the power difference between the most silent and the most activated FFT bins in a (logarithmically spaced) band. While in [Jiang et al., 2002] eight octave-scaled bands are used, [Aucouturier and Pachet, 2004] use the mel bands from MFCC computation. These values are decorrelated by the KL-transform ([Jiang et al., 2002]) or by DCT ([Aucouturier and Pachet, 2004]). In this work, the way of calculating Spectral Contrast is adopted from [Aucouturier and Pachet, 2004], i.e., the mel bands and DCT from MFCC calculation are re-used, and the resulting contrast features are appended to each MFCC frame. This procedure yields a vector with a dimensionality of twice the number of MFCCs. [Aucouturier, 2006] reports an improvement of about 1% compared to standard MFCCs, measured by R-Precision.

2.6.1.2. Kernel-based features. Another approach to add information that is not contained in the MFCC vectors is to apply particularly designed filter kernels to the 2D representation of the spectrogram to detect the presence of specific aspects of the signal. This approach has been adopted from computer graphics. For music feature extraction, [Deshpande et al., 2001] use 25 directional Gaussian filters, while [Pohle et al., 2006a,Yu and Slotine, 2009] calculate the kernels from a number of randomly selected patches of training examples. In this thesis, filter kernels are designed manually after inspecting the spectrograms for the shape of harmonic and percussive structures (cf. [Ono et al., 2008]). The used time-frequency resolution and the two filters are discussed next. The approach could also be extended to other structures of interest, such as specific overtone constellations or short-time developments.

Time-Frequency Representation. The features described in this section are meant to describe finer-scaled aspects of the spectrogram, so the spectrogram is divided into 128 instead of 64 bands as used for MFCC computation. To obtain a representation that allows a straightforward interpretation of the audio signal of music, the signal is represented on a cent scale. The signal parts below about

70 Hz and above 95% of the Nyquist frequency are discarded. The applied filterbank transforms the magnitude FFT (i.e., amplitudes, not power) which is calculated to have a T_0 of approximately 0.1 sec into cent bands that are 66.7 cent apart by a weighted sum of the corresponding FFT bins. Subsequently, in line with previous work that use the sone scale (e.g., [Pampalk, 2001]), a scale transform is applied to obtain values presumably more closely related to human loudness perception. Based on the notion that over a wide range, perceived loudness approximately doubles when the signal pressure level is increased by 10 dB [Fastl and Zwicker, 2007], each value a in the cent-scaled spectrogram is transformed by

$$a_{\text{loudness}} = 2^{\log_{10}(\frac{a}{p_0})} \tag{35}$$

The effect of changing the constant p_0 is a linear scaling of a_{loudness} (the actual scale factor is $\frac{1}{2^{\log_{10}(p_0)}}$)[16].

Harmonicness. When observing spectrograms, it gets apparent that tones in the music typically appear as horizontal lines stacked above each other. Depending on timbre and clarity, the strengths and positions of these partial tones are different. "Harmonicness[17]" is a feature that is designed to extract such horizontal lines. Based on the observation that notes are typically held for at least five frames, the width of the kernel is set to five. The actual values of the kernel are chosen based on the observations that bands close to a partial typically contain only little power, but that a given partial may be measured in several (e.g., two or three) neighboring bands. After mean removal, values are scaled so that all positive values sum up to one. The resulting kernel is depicted in Figure 15. An example spectrogram of the effect of Harmonicness filtering and subsequent half-wave rectification is given in Figure 17. As can be seen, Harmonicness is related to strength of tone harmonics.

Attackness. Attackness is a similar concept applied to the clear vertical structures of the spectrogram, which correspond to drum and percussive-like sounds. The kernel is based on the same as the one used to compute Harmonicness, but rotated by 90 degrees. The focus lies on detecting onsets, i.e.,

[16] Note: Due to a bug in the implementation, the natural logarithm was used in the computations presented here, which approximately corresponds to taking the 2.3^{th} power of the intended values

[17] We chose this rather awkward name because this feature is loosely associated with harmonicity, but does not actually measure it exactly.

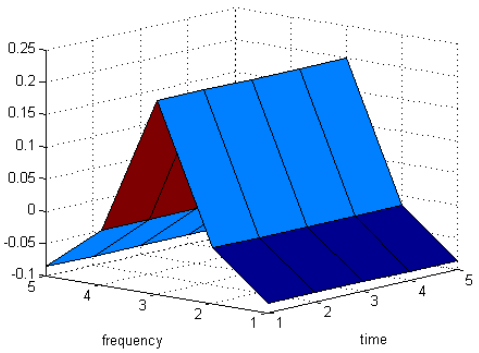

Figure 15: Kernel used to compute Harmonicness on a cent-scaled time-frequency resolution. Bands are 66.7 cents, and frames are 46.4 ms apart.

it is only of importance that the energy in many bands increases simultaneously, and not how long it remains at the higher level. Therefore, the values in the two rightmost columns are set to zero, i.e., the values in the rightmost frames have no effect on the resulting value. The resulting kernel is depicted in Figure 16. This feature resembles Percussiveness as described in [Pampalk, 2006b]. However, it is computed not on two, but on three consecutive frames, and assigned one particular frame (which is important for combination with MFCCs).

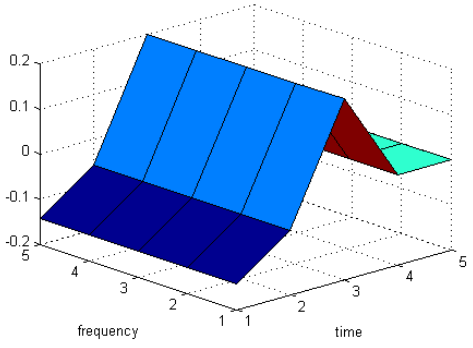

Figure 16: Kernel used to compute Attackness on a cent-scaled time-frequency resolution. Bands are 66.7 cents, and frames are 64.4 ms apart.

Combining Harmonicness and Attackness with MFCCs. Harmonicness and Attackness are meant to complement the information contained in MFCCs, so they are added to each frame's MFCC vector as additional components. To

this end, all filtered values of a frame are half-wave rectified and summed up to obtain one value for the feature. The Gaussian model is then built with the MFCC vectors enriched with two additional values. As the MFCC frames and cent frames may have a different frame size and hop size, Harmonicness and Attackness values are interpolated appropriately. The way of comparing Gaussians is not changed.

Harmonicness and Attackness also can be used in other ways in MIR applications. For example, they can be used to order music in a collection by the amount of percussive drum events as opposed to the amount of melodic sounds contained in the signal. Such a combined feature value is calculated by the H2A Ratio

$$\text{H2Aratio}_i = \frac{h_i}{h_i + a_i} \tag{36}$$

where h_i and v_i are the sum of the half-wave rectified Harmonicness and Attackness values over all frames of song i, divided by the total sum of all values in song i's cent spectrogram. Generally, a low H2A Ratio indicates mostly percussive tracks with few sustained sounds, while a high H2A Ratio indicates tracks with mostly steady and harmonic instruments with few percussion sounds. Ordering the tracks in a collection by H2A Ratio gives a roughly steady progression between these two extremes. An example use of this descriptor is given in Section 7.1. It is related to *Percussiveness* [Pampalk, 2006b], but is independent of the overall loudness, and measures aspects of both harmonic and percussive elements.

2.6.2. Alternative Distance Measures between Gaussians

In this section, the evaluated distance measures between Gaussians are given.

2.6.2.1. KL Divergence In the domain of music similarity computation, the Kullback Leibler (KL) Divergence can be regarded as the most commonly used measure for comparing Gaussians. The (asymmetric) KL divergence between Gaussians $\mathcal{N}_1$ and $\mathcal{N}_2$ with means μ_1, μ_2 and covariance matrices Σ_1, Σ_2 can be computed as [Penny, 2001]

$$\begin{aligned} D_{\text{KL}}(\mathcal{N}_1, \mathcal{N}_2) = & \quad 0.5 \log \frac{|\Sigma_2|}{|\Sigma_1|} + 0.5 Tr(\Sigma_2^{-1}\Sigma_1) \\ + & \quad 0.5(\mu_1 - \mu_2)^T \Sigma_2^{-1}(\mu_1 - \mu_2) - \frac{d}{2} \end{aligned} \tag{37}$$

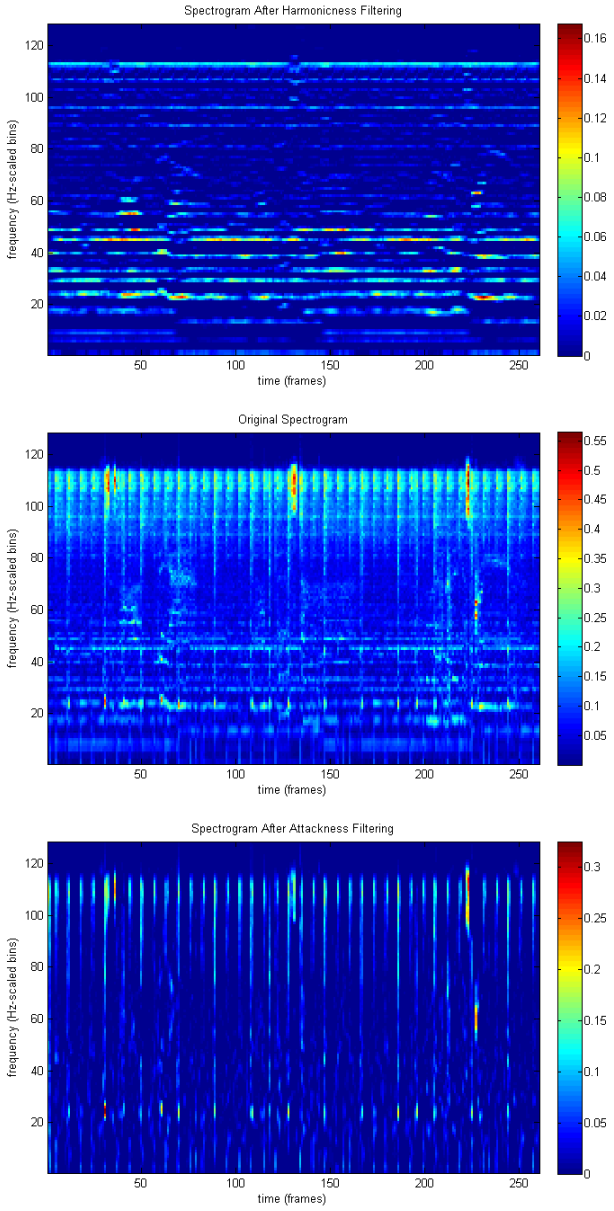

Figure 17: Spectrogram of a song excerpt before and after Harmonicness / Attackness filtering and half-wave rectification. The original spectrogram is given in the center to facilitate comparison with the filtering results. The different scales can be accounted for by multiplication with a constant factor.

where $|\Sigma|$ is the determinant of matrix Σ and d is the number of dimensions. The constant term $\frac{d}{2}$ can be disregarded in the context considered here. A symmetrised version is given by [Mandel and Ellis, 2005b]

$$\begin{aligned} D_{\mathrm{2KL}}\left(\mathcal{N}_1, \mathcal{N}_2\right) = & \quad Tr\left(\Sigma_2^{-1}\Sigma_1 + \Sigma_1^{-1}\Sigma_2\right) \\ & + \left(\mu_1 - \mu_2\right)^T \left(\Sigma_2^{-1} + \Sigma_1^{-1}\right)\left(\mu_1 - \mu_2\right) \\ & - 2d \end{aligned} \tag{38}$$

In the experiments presented below, this is used as a baseline algorithm.

2.6.2.2. Alternative to symmertised KL divergence. As an alternative to calculating the symmetrised KL divergence, we use Equation 23 with the entropy calculated on multivariate normal distributions (cf. [Lamberti and Majtey, 2003], first equation on p. 84).

$$D_{\mathrm{JS}} = H(\mathcal{M}) - \frac{1}{2}H(\mathcal{N}_1) - \frac{1}{2}H(\mathcal{N}_2) \tag{39}$$

where H denotes the entropy, and $\mathcal{M}$ is the GMM resulting from merging $\mathcal{N}_1$ and $\mathcal{N}_2$ with same weights. We approximate $H(\mathcal{M})$ in a straightforward way by merging $\mathcal{N}_1$ and $\mathcal{N}_2$ into a Single Gaussian [Huber et al., 2008]. Two Gaussians are merged according to the following equations [Ma and He, 2005]:

$$\begin{aligned} z_3 &= z_1 + z_2 \\ \mu_3 &= (z_1\mu_1 + z_2\mu_2)/z_3 \\ \Sigma_3 &= (z_1\Sigma_1 + z_2\Sigma_2 + z_1\mu_1\mu_1' + z_2\mu_2\mu_2' - z_3\mu_3\mu_3')/z_3 \end{aligned} \tag{40}$$

where z_1 and z_2 give the relative weights of $\mathcal{N}_1$ and $\mathcal{N}_2$ (0.5 in this work).

The entropy of a Single Gaussian $\mathcal{N}(\mu, \Sigma)$ is given by (e.g., [Srivastava and Gupta, 2008]):

$$h(X) = \frac{d}{2} + \frac{d\ln(2\pi)}{2} + \frac{\ln|\Sigma|}{2} \tag{41}$$

We use the square root of D_{JS}.

2.6.2.3. Mahalanobis Distance In [Xu et al., 1998], Mahalanobis distance between two Gaussians is calculated as follows:

$$\begin{aligned} dis_1 &= \left(\mu_2 - \mu_1\right)^T \Sigma_1^{-1}\left(\mu_2 - \mu_1\right) \\ dis_2 &= \left(\mu_1 - \mu_2\right)^T \Sigma_2^{-1}\left(\mu_1 - \mu_2\right) \\ dis &= \left(dis_1 + dis_2\right)/2 \end{aligned} \tag{42}$$

Note that this definition is different from the one given in [Levy and Sandler, 2006], where instead of using the covariance matrices of the two songs directly as above, covariance matrices were used to reflect the variance of mean and covariance over the whole collection. While comparison time is reported to be magnitudes faster, this showed to produce lower accuracies than using KL divergence.

2.6.2.4. L2 Distance The normalised L2 Distance between two Gaussians is defined as [Jensen et al., 2007]

$$d_{nL2}(\mathcal{N}'_1, \mathcal{N}'_2) = \int (p'_1(x) - p'_2(x))^2 dx \tag{43}$$

where $\mathcal{N}'$ is $\mathcal{N}$ scaled to unit L2-norm by $\mathcal{N}' = p(x)/\sqrt{\int p(x)^2 dx}$. By basic math, Equation 43 can be reformulated as $d_{nL2}(\mathcal{N}'_1, \mathcal{N}'_2) = 2(1 - \int p'_1(x) p'_2(x) dx)$, which can be computed by applying formulae (5.1) and (5.2) from [Ahrendt, 2005]. [Jensen et al., 2007] report that L2 on GMMs of MFCCs is slightly inferior to the KL distance with respect to genre classification accuracy. Here it is included to test its performance when other features are added.

2.6.2.5. Bhattacharyya Distance The continuous form of the Bhattacharyya distance for normal distributions is [Fukunaga, 1997]

$$\begin{aligned} D_B(\mathcal{N}_1, \mathcal{N}_2) = & \frac{1}{8} (\mu_1 - \mu_2)^T \left(\frac{\Sigma_1 + \Sigma_2}{2} \right)^{-1} (\mu_1 - \mu_2) \\ + & \frac{1}{2} \ln \left(\frac{\left| \frac{\Sigma_1 + \Sigma_2}{2} \right|}{\sqrt{|\Sigma_1|} \cdot \sqrt{|\Sigma_2|}} \right) \end{aligned} \tag{44}$$

The Bhattacharyya coefficient[18] is calculated from the Bhattacharyya distance as

$$\rho(a, b) = \exp(-D_B(a, b)) \tag{45}$$

[Comaniciu et al., 2003] prove that – in contrast to the Bhattacharyya coefficient itself – for the discrete case the distance

$$D_{\mathrm{BC}}(a, b) = \sqrt{1 - \rho(a, b)} \tag{46}$$

[18] see Online Encyclopaedia of Mathematics, http://eom.springer.de/default.htm, "Bhattacharyya distance" http://eom.springer.de/B/b110490.htm

is a metric. In this thesis, both $D_{\mathrm{BC}}(a, b)$ and a simple variation of it is evaluated. We merge $\mathcal{N}_1$ and $\mathcal{N}_2$ by applying Equation 40 (cf. Section 2.6.2.2 and 2.5.1.5), and sum up the distances D_{BC} from each Gaussian to the merged Gaussian. This distance measure is denoted JSB.

2.6.3. Evaluation: Choosing Distance Measure and Additional Features

Evaluation is carried out as in Section 2.5. Altogether, there were 96 different combinations of similarity algorithms and feature sets. All these were evaluated with 26 MFCCs *including* the 0^{th} coefficient (i.e., coefficients 0 to 25). As a comparison to G1Cmod, also the corresponding evaluation results for MFCCs 1 to 25 without additional features are given for the KL divergence. Results for DB-MS and DB-L* are given in Figures 18 and 19. In the evaluation on DB-MS, it turns out that the evaluated variant of the Mahalanobis distance (Equation 43) was significantly below those of the baseline algorithm even when adding the evaluated additional features. For better visibility, results for Mahalanobis distance are not shown in the figure. Average classification accuracies for the evaluated Mahalanobis distance were between 52.2% (MFCCs + Delta Coefficients + Acceleration Coefficients) and 56.6% (MFCCs + Harmonicness & Attackness + Spectral Contrast).

2.6.3.1. Estimating distance measure performance. On DB-MS, for any given feature combination of those evaluated, JSB has a higher classification accuracy than all other distance measures, and L2 has a lower classification accuracy than all other distance measures. Extending this way of counting how often similarity measures outperform other measures for a given feature set, it is possible to create a performance ranking. As D_{JS} outperforms B in 13 out of 16 comparisons, and B outperforms KL in 14 out of 16 comparisons, this ranking takes the form: JSB – JS – B – KL – L2 – Mahalanobis. Based on these results, L2 and Mahalanobis distance were not evaluated on DB-L*. The corresponding evaluation for the remaining four distance measures based on results obtained on DB-L* yields the same ranking.

2.6.3.2. Comparing feature performance. By counting in how many cases classification accuracy increases when a particular feature is added (as compared to the analogous algorithm without this particular feature), it is possible to create a ranking of features. On DB-MS, altogether there are 16 feature

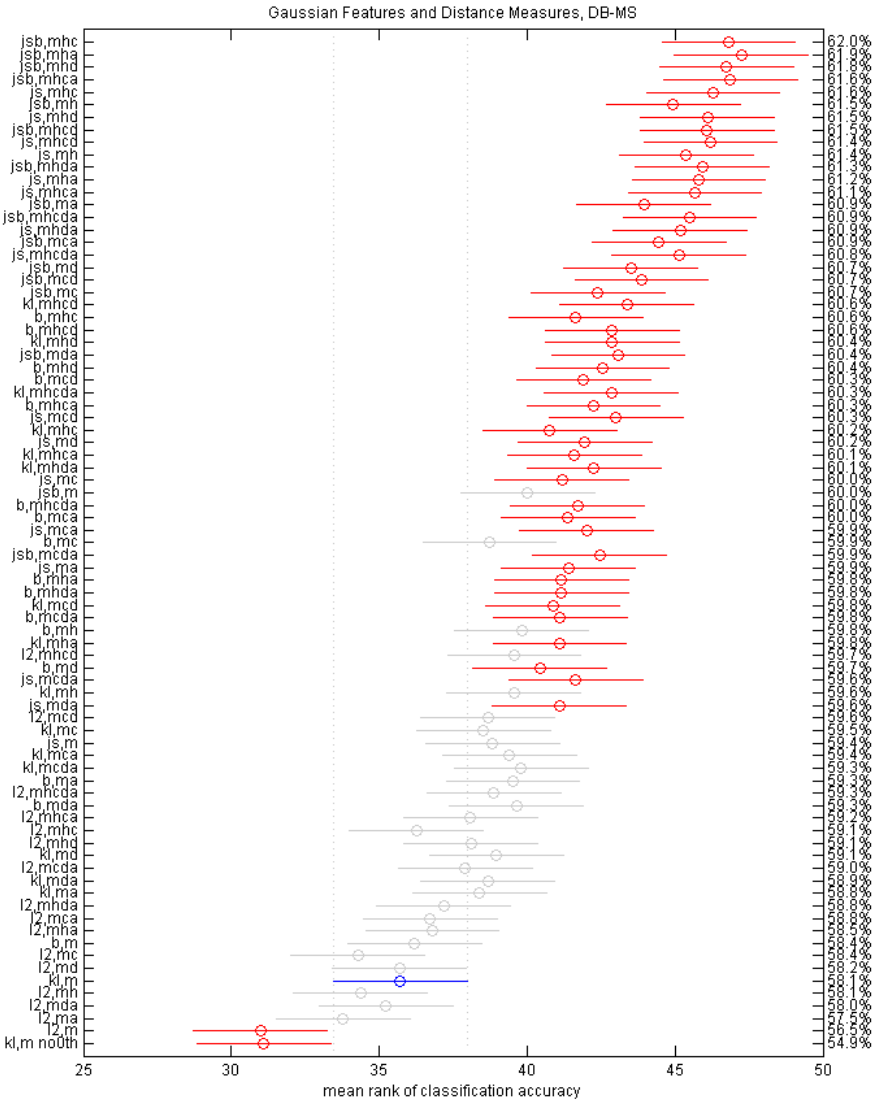

Figure 18: DB-MS: Variants for Gaussian Component. Distance measures: l2 - Euclidean Distance, kl - KL Divergence, b - Bhattacharyya Coefficient, js - D_{JS}, jsb - modified measure using Bhattacharyya Coefficient. Features: d - Delta Coefficients, a - Acceleration Coefficients, c - Spectral Contrast, h - Harmonicness / Attackness

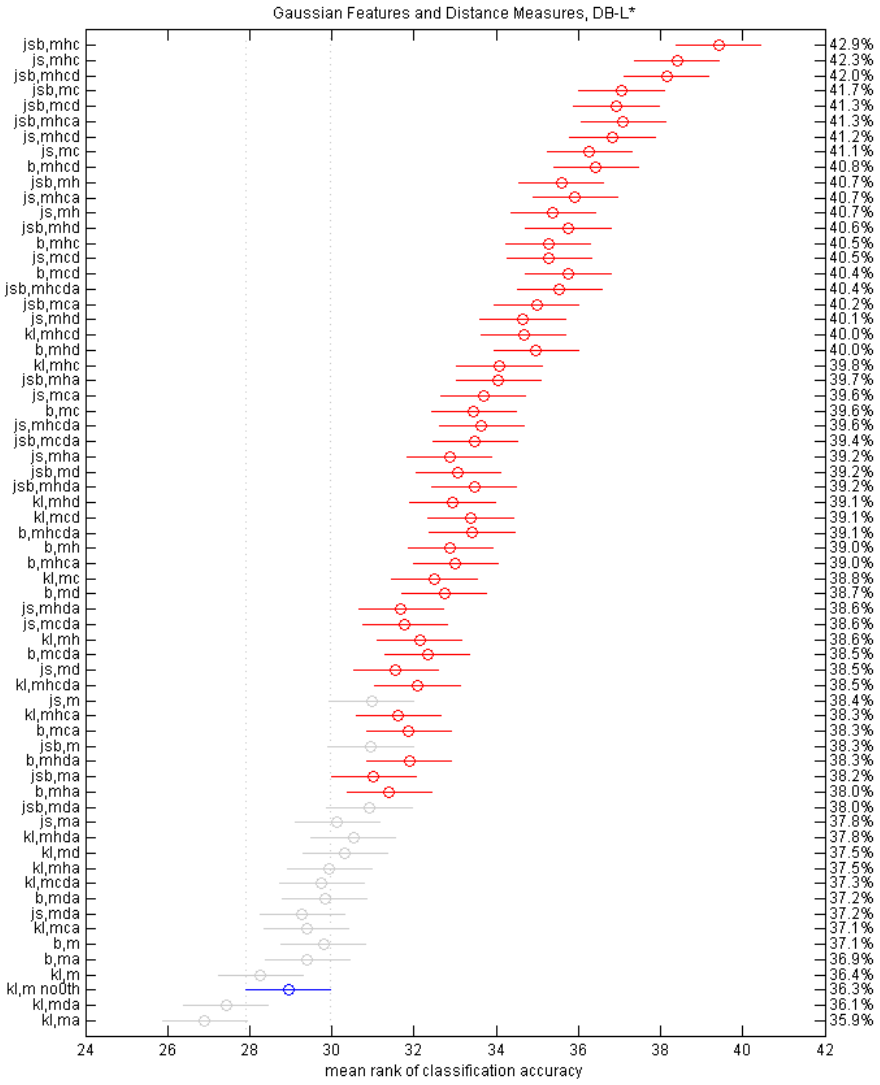

Figure 19: DB-L*: same evaluation as in Figure 18.

combinations (of which 8 contain a particular feature and 8 don't contain this feature). Counting this over all 5 distance measures gives a maximum score of 40. The only feature that scores this high on DB-MS is Harmonicness / Attackness, i.e., adding this to any existing feature combination (and distance algorithm) always increases classification performance in the evaluations conducted here. The corresponding scores for the other features were 31 (Spectral Contrast), 25 (Delta Coefficients) and 13 (Acceleration Coefficients). The same ranking is obtained on DB-L*, where the maximum possible score is 32 as L2 was not considered: Harmonicness / Attackness (32), Spectral Contrast (32), Delta Coefficients (18) and Acceleration Coefficients (0). All these can be calculated from the figures in Appendix A.

The results considering Spectral Contrast are in line with those in [Aucouturier, 2006], i.e., there is a (slight) improvement when adding Spectral Contrast to MFCCs.

2.6.4. Estimating the Impact of Some Simple Modifications

Having assessed the performance of the features in their basic form, in this section a number of simple modifications are discussed, and their impact on the overall performance is estimated.

2.6.4.1. Estimating a good number of Spectral Contrast coefficients.

To estimate the contribution of each Spectral Contrast Coefficient to the overall performance, and to see if using fewer coefficients might improve performance, genre classification experiments were conducted. The results are reported in Figure 20.

It can be seen that on the Normalization Collection, the optimum classification accuracy is already obtained with about 10 coefficients, and stays more or less constant when using more coefficients. However, on DB-MS, classification accuracy continuously increases when adding up to 25 coefficients (which was the maximum evaluated). When combining the Spectral Contrast coefficients with 25 MFCCs, differences become rather small, although there still seems to be some improvement with increasing number of coefficients. The conclusion from these experiments seems to be that one can use all (25) coefficients, including the 0^{th}, but it should also be safe to reduce the number of used coefficients to 10, for instance.

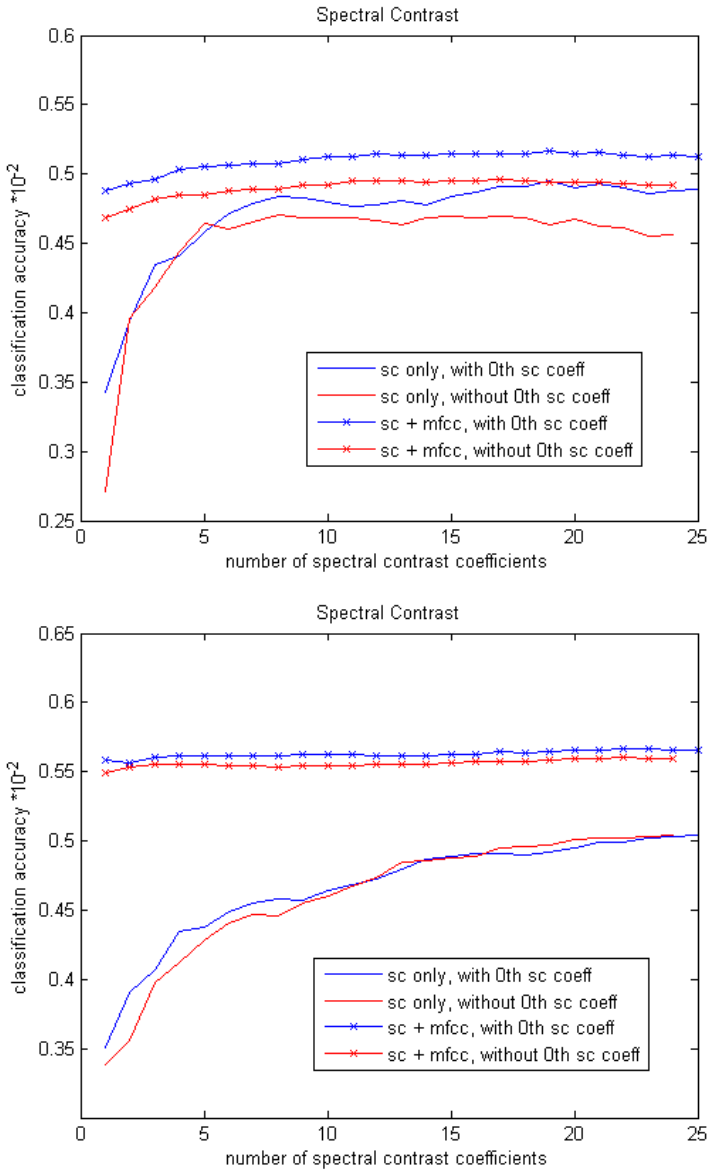

Figure 20: Classification accuracies (given on a scale from 0 to 1) when using Spectral Contrast as the only feature, KL Distance. Above: Normalization Collection, 10NN. Below: DB-MS, 20NN.

2.6.4.2. Estimating the use of 2D MFCCs. The experiments presented above indicate that the most recommendable feature combination is MFCCs, Spectral Contrast and Harmonicness / Attackness. The previous experiment indicates that modifying the number of Spectral Contrast coefficients has no large impact on the performance. Delta and Acceleration Coefficients are not part of the selected feature set, and thus the set contains no feature based on temporal succession of MFCCS.

In this section, it is examined if modeling such short-time progressions by *two-dimensional MFCCs* is suited to improve classification accuracy. Two-dimensional MFCCs (cf. [Bouvrie et al., 2008]) are calculated by taking for each MFCC frame the n next MFCC frames, and calculating the n-point DCT over these frames for each MFCC coefficient. When computing c MFCC coefficients, this procedure yields $n \cdot c$ coefficients, which are stacked into one feature vector for the current frame. Two tracks can be compared with the Single Gaussian approach.

For the corresponding experiments given in Figure 21, $n = 3$ and $n = 5$ were evaluated. It can be seen that classification accuracy does not exceed the one obtained with 25 (one-dimensional) MFCCs. Consequently, in subsequent experiments, 25 MFCCs in their basic form are used. Evaluating if in combination with other features, 2D-MFCCs perform better than basic MFCCs is left for future work.

2.6.5. Runtimes of distance measures.

With the experiments that are presented in the previous sections, indication is given that the techniques evaluated so far can not be improved with the discussed simple modifications. As a consequence, the combined measure (and final resulting algorithm of this thesis) will be built on the basic distance measures that were evaluated in the Friedman tests presented above. Before proceeding to combining the basic distance measures into a common measure (to which Section 2.7 will be dedicated), the important aspect of runtime is discussed. Figure 22 gives the results of the distance measures between single Gaussians Features were implemented and run in Matlab. No specific runtime optimizations were done except precomputation of partial results that are needed during distance calculation (such as entropy values or matrix inverse and determinants). The computer had a 3 GHz CPU and a usual software setup.

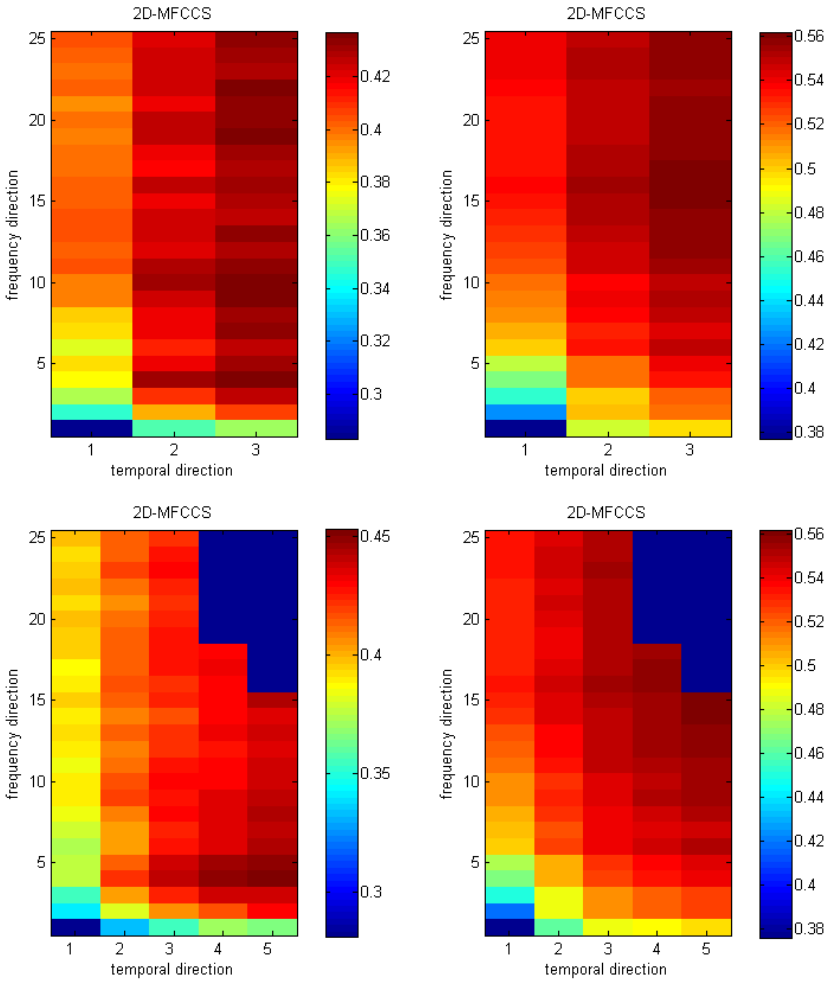

Figure 21: Classification accuracies on 2D MFCCS calculated from 3 and 5 consecutive frames when all coefficients within the matrix up to a particular pair of indices are used. E.g., the value at position $5, 2$ indicates the accuracy based on the coefficients in the matrix with lower left edge $1, 1$ and upper right edge $5, 2$. KL Distance. Left: Normalization Collection, 10NN. Right: DB-MS, 20NN. Above: 3 consecutive frames, below: 5 consecutive frames (variants including more than 75 coefficients were not computed and marked blue). Altogether, 366 classification experiments were run.

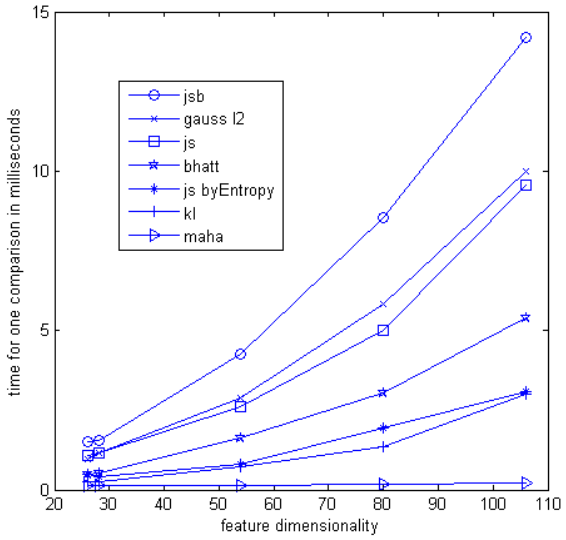

Figure 22: Runtimes of distance measures between single Gaussians for various dimensionalities of the feature data.

It can be seen that the evaluated variant of the Mahalanobis distance, which was the measure with the lowest performance in the experiments, has the fastest runtime. On the other side of the scale, the measure that performed best (JSB) has the highest runtime. Two implementations of D_{JS} are evaluated. The first is to calculate and sum up the KL divergence of each Gaussian to the merged Gaussian (denoted *js*), and the second is the calculation by entropy given in Section 2.6.2.2. With the used implementations, computing D_{JS} based on the entropy instead of the KL divergence brings a clear performance improvement. In these experiments, its runtime is comparable to the basic implementation of the symmetrised KL divergence. As we see no obvious optimization for JSB, and as D_{JS} outperforms KL divergence in the classification experiments, we choose D_{JS} to be used in the final combined similarity measure of this thesis.

2.6.5.1. Conclusion Combining the results from this section, it follows that of the evaluated feature combinations, MFCCs with Spectral Contrast and Harmonicness / Attackness performs best, which will be selected for subsequent evaluations. While the most recommendable distance measure is JSB, this measure takes most time to be computed in the evaluated implementation. Replacing it with D_{JS} that is much faster to compute brings a minor, yet not

significant, decrease in classification accuracy when using the selected feature set. All these aspects make this combination (denoted JS-MHC) the candidate that was finally selected for combination experiments.

2.7. Combining the Parts into one Similarity Measure: G1Cmod2

In the previous sections, the various basic similarity measures used in G1C / C1Cmod are discussed, and variants are proposed and evaluated. Each of the basic similarity measures focuses on a more or less particular aspect of the signal. To obtain one overall similarity measure, these basic similarity measures have to be combined. In this section, various approaches to accomplish this are discussed and evaluated. The outline is as follows. To restrict the potentially endless space of potential experiments, first for each basic similarity measure, one variant of those evaluated in the previous sections is chosen. In this choice, various aspects are taken into consideration, including the performance of the various variants in the previously presented experiments. Subsequent experiments focus on how to combine these chosen basic measures. Generally, the weighting of the basic measures is the same as in G1C / G1Cmod (i.e., 0.7, 0.1, 0.1, and 0.1). This weighting is adapted from [Pampalk, 2006b], and an evaluation of the effect of modifying these weights for the modified features is left for future work. From the experiments, a number of conclusions is drawn.

2.7.1. Chosen basic similarity measures.

When choosing a variant of a basic similarity measure as a building block for a combined measure, various aspects play a role. First of all, its performance in the previous experiments should be above the baseline (i.e., higher than the measure used in G1C / G1Cmod). If all chosen basic similarity measures are metrics (most notably, fulfill the triangle inequality), then the combined measure also might be a metric, subject to the chosen combination method. This is a condition useful for using some fast indexing and retrieval methods, such as M-Tree [Ciaccia et al., 1997] or FastMap [Faloutsos and Lin, 1995]. Another preferable property is that the basic similarity measure be bounded, which can make the range of the resulting combined similarity measure more predictable. A measure that is not bounded can be transformed to be bounded (e.g., by applying Equation 4).

Based on such considerations, for FPs (Section 2.5.2), D_5 calculated from unnormalised FPs was chosen. It has a high ranking on both evaluation collections, it is bounded in the range $[0, 1]$, and empirical tests indicate that it fulfills the triangle inequality. Furthermore it is meaningful from a conceptual point of view. The same measure (D_5 calculated from unnormalised data) was also chosen for the FP timbre component FPT (Section 2.5.4) for similar reasons.

For the rhythm component FPR, the experiments indicated that Correlation seems the best choice (cf. Section 2.5.3). However, it does not fulfill the triangle inequality. The performance of Jensen-Shannon (JS) divergence on FPRs normalised to sum to one is not much lower (no significant difference), and its square root is a metric. Furthermore, the resulting values are bounded (cf. [Briët and Harremoës, 2009]), as for a given number of bins, there is a maximum value for the resulting entropy.

For the Gaussian component, the experiments (Section 2.6) indicate that there is not much of a choice. As discussed in Section 2.6.3, the square root of D_{JS} is chosen. From [Huber et al., 2008], Theorem 3, we conclude that the (exact) result of Equation 39 is bounded by $-\log 0.5$. However, experiments on the DB-Norm collection indicate that our approximation (which gives an upper bound of the exact value, which we conclude from [Huber et al., 2008], Theorem 4), in many cases yields values above this bound even for the closest tracks. For example, when calculated on Gaussians calculated from MFCCs containing coefficients 0..25, resulting distances are larger than $-\log 0.5 \approx 0.69$ even in all cases, except for identity. They have an average value of 3.23, and a maximum value of 8.1. A possible reason may be the dimensionality of the data.

2.7.2. Pre-Normalization.

The similarity measures that are to be combined are likely to be of different magnitudes and to be differently distributed along their scales. An approach to handle this is to assume a Gaussian distribution, and apply mean removal and division by standard deviation (z-normalization[19]). In [Pampalk, 2006b], normalization factors are determined over a large number of track distances, and then kept fixed for a given basic distance measure. While the resulting combination has the same effect as arithmetic weighting (as constant offsets do not have an effect on the similarity ranking), weighting factors appear as

[19] http://en.wikipedia.org/w/index.php?title=Standard_score&oldid=316839844

separate terms and are more intuitively understandable (0.1, 0.1, 0.1 and 0.7, which sum up to one). To normalize the distances of a particular track to all other tracks in the collection by z-normalization has been proposed by Schnitzer [Pohle and Schnitzer, 2007]. Possible alternatives to z-normalization include those presented in [Pohle et al., 2006b].

Such normalization approaches that determine normalization factors for each seed track separately have three aspects that need to be taken into consideration. First, in most cases this invalidates the triangle inequality. Second, when normalization factors are set for each seed track separately, then the resulting similarities are not symmetric any more. This can be compensated by summing up the calculated distances in each direction. Third, normalization factors depend on (all) the other tracks in the music collection. This implies that distances between tracks may change when new tracks are added to the collection, or when tracks are removed. To avoid this, it is possible to calculate the normalization factors not on the (whole) music collection, but rather use a small fixed set of songs to compute the factors. Such a set is called *normalization collection* in this work. The music contained in it could be chosen to be representative for the music (expected to be) in the collection, or aiming to cover a wide range of different music styles.

2.7.3. Transforming Distances to Probabilities.

An alternative to such pre-normalization is to transform distances to probabilities based on class membership knowledge (cf. [Mahamud and Hebert, 2003]). A linear combination of basic distance measures assumes that the perceived "quality" associated with the distance decreases linearly, which is unlikely to hold. Here, an empirical approach to model how calculated distance and actual similarity presumably are related is evaluated. Using the DB-Norm collection, a number of (quite) optimistic assumptions is made. First, it is assumed that all tracks within a genre (class) are "similar", and all pairs of tracks that are in different classes are "not similar". Furthermore it is assumed that due to the variety of the contained music, the range of measured distances and their distribution is what one usually could expect to observe. By Bayes' rule, the probability that for a measured distance d between two tracks i and j the two tracks are not in the same class y is given by [Mahamud and Hebert, 2003]

$$p(y_i \neq y_j|d) = \frac{p(d|y_i \neq y_j) \cdot p(y_i \neq y_j)}{p(d)} \tag{47}$$

Following [Mahamud and Hebert, 2003], for the range of observed distances d, this is modeled by kernel density estimation. The modeled distribution is sampled at 100 equidistant values, which are stored for use by interpolation (table lookup). While it would be possible to build elaborated combination approaches upon this (cf. [Mahamud and Hebert, 2003]), here the interest is rather if the transformation to probabilities is beneficial.

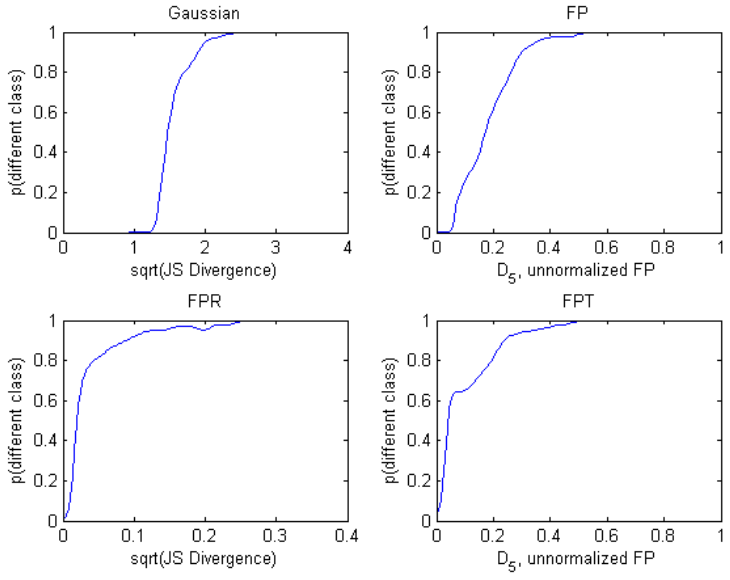

Figure 23: Probability that for a given distance, two tracks that have the given distance are in different classes of DB-Norm. The highest FPR values produced a sharp decay in the curve, which is most likely due to outliers (tracks in the same class having high distances), which was removed manually by setting the associated probabilities to 1.0.

When transforming the distances to probabilities for the chosen basic similarity measures, the resulting probabilities associated with each distance are given in Figure 23. These are used to transform the occurring distances to probabilities in some of the classification experiments presented below.

2.7.4. Combination approaches.

A common and straightforward way to combine different audio similarity measures is by arithmetic weighting. In [Pampalk, 2006b], it is stated that the arithmetic weighting has the conceptual problem that a single aspect of the

music pieces is sufficient to be considered similar by a human listener, which is not reflected by arithmetic weighting. Based on this concept, two algorithms submitted to the MIREX'07 AudioSim task [Bosteels and Kerre, 2007a] computed the minimum and maximum of FP and GMM distances, respectively. Results indicate that taking the minimum value of the similarity measures yields better results than taking their maximum similarity. However, based on these results only, it can not be concluded that taking the minimum or maximum outperforms arithmetic weighting.

In this thesis, a number of combination approaches is evaluated:

1. Z-Norm Global: Static normalization factors as used in G1C. Means and variances for each basic similarity measure are determined on all pairwise distances calculated on the current collection.

2. Z-Norm Local: This is the approach as used in G1Cmod. Normalization factors are determined for each track based on the distance of this track to all other songs in the collection. When two tracks are compared, the resulting basic similarities are normalised twice (once with each set of normalization factors belonging to each of the two tracks), and the result is summed up.

3. Z-Norm Normcollection: Same principle as Z-Norm Local, but the normalization factors associated with each track are determined by calculating the distances to a (static) normalization collection that is distinct from the music collection, and was determined beforehand.

4. Probability-Based: Distances are transformed to probabilities as discussed above. The basic similarity measures are combined by taking the minimum, maximum and mean of their estimated probabilities. Furthermore, a simple multiplication-based weighting is evaluated. After transforming each basic distance measure D_x to the range of 0..1, the distance is computed as $D = 1 - \prod_x (1 - w_x \cdot D_x)$.

The approaches are evaluated on the two optimization collections. Results are given in Figure 24.

It can be seen that the combination method is an important step, in which potential increases in quality can be lost if the wrong method is applied. The three top-ranked combination methods have the same ranking on both collections, indicating a certain stability of the results. When used together with the selected basic similarity measures, they outperform the baseline algorithms (G1C / G1Cmod) on both collections (although no significance is measured on DB-L* between G1Cmod and the global method). The experiments don't

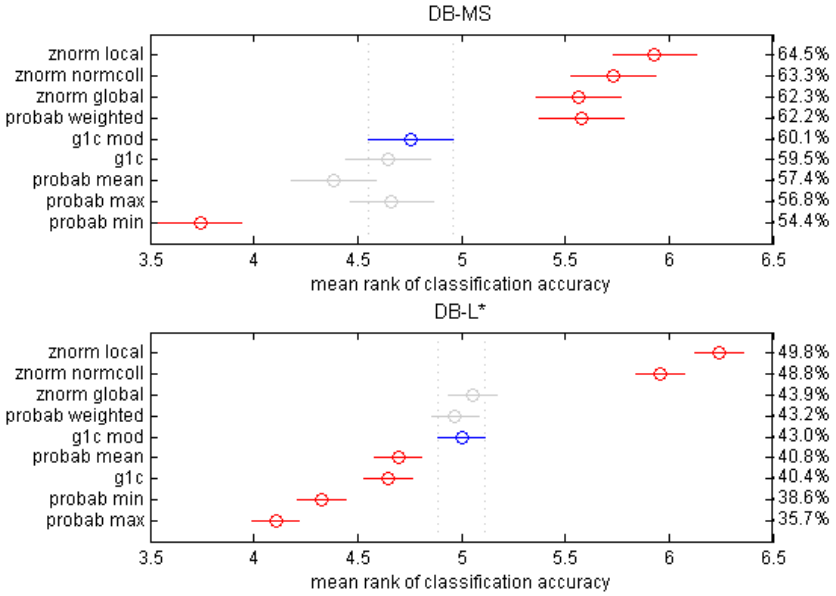

Figure 24: Combination approaches.

show that the probability based method is favorable, although the *weighted* combination slightly outperforms G1Cmod.

The global method – although not significantly better than G1Cmod on DB-L* – has the advantages over the local methods that empirical results show it is a metric, which is not the case for the local methods (and neither for G1C nor G1Cmod). Using a normalization collection instead of the current music collection to determine the normalization factors brings a slight yet not significant performance decrease as compared to the best-performing method, local z-normalization.

To facilitate a comparison with other publications, the classification accuracies achieved with this method on DB-MS are given in Table 3. In this experiment, classification accuracies are determined differently than in most other experiments presented in this thesis. In most other experiments, accuracies are average *percentages* of tracks in the set of k closest tracks that are in the correct class. In contrast, for this experiment a majority voting is done, i.e., the predicted class is the class to which most of the k closest tracks belong. If this class is the correct class, then the seed track is considered as classified correctly (and gets a score of 1), otherwise not (score 0). In case of ambiguity, i.e., if there is no clear majority voting decision, then the seed track gets a score of $\frac{1}{n}$ if the correct class is among the n candidate classes, otherwise it gets a score

of 0. This way of scoring simulates a very large number of experiments with randomly assigning one of the n candidate classes.

NN	1	3	5	10	20	50
No AF	**87.2%**	84.8%	83.8%	81.4%	79.8%	72.8%
With AF	**73.0%**	73.0%	74.5%	73.6%	70.6%	67.6%

Table 3: Leave-one-out genre classification of G1Cmod2 on DB-MS. Classification is based on majority voting, in contrast to most other experiments in this thesis, that give percentages of matching songs. With 10-fold cross validation, the 1NN value is 72.8% with AF.

NN	1	3	5	10	20	50
No AF	**83.7%**	81.6%	80.2%	77.5%	74.3%	69.7%
With AF	**71.6%**	70.2%	70.4%	69.6%	66.6%	64.9%

Table 4: Same evaluation as in Table 3 but based on only 30 seconds from the center of each audio file. With stratified 10-fold cross validation, 1NN values are 83.3% without artist filter (AF) and 71.0% with AF.

In comparison with previously published approaches (most notably, G1C [Pampalk, 2006b]), the results in Table 3 are about 5 percentage points higher. Table 4 give the results obtained when features are extracted from 30 s (sampled at 22 kHz) from the center of each audio file, instead of using 120 s. The obtained results are clearly above those typically found in the literature for this algorithm. For example, also using 30 s / 22 kHz excerpts, [Moerchen et al., 2006] report classification accuracies (obtained by 10-fold cross validation) of up to 70% based on up to tens of thousands of audio features and state-of-the-art machinge learning algorithms. [Pohle, 2005] reports a classification accuracy of up to 72.3% (based on 30 s excerpts sampled at 11 kHz). These results are about 11.4 percentage points below those obtained with G1Cmod2. With these results, it has to be kept in mind that the G1Cmod2 algorithm has been optimised on DB-MS, so that there is the danger of overfitting. Consequently, comparisons of this new algorithm G1Cmod2 to G1C and G1Cmod on other music collections are carried out to estimate to which extent these results are generalizable. These experiments are presented in the next section.

2.8. Evaluation of the Resulting Algorithm

In Figure 25, comparisons of the developed algorithm (G1Cmod2) to the baseline algorithms (G1C and G1Cmod) are given. In the evaluation, G1Cmod2

performs significantly better than the baseline algorithms. The improvement is 6.8 and 11.2 percentage points in comparison to G1C, and 5.8 and 10.2 percentage points in comparison to G1Cmod.

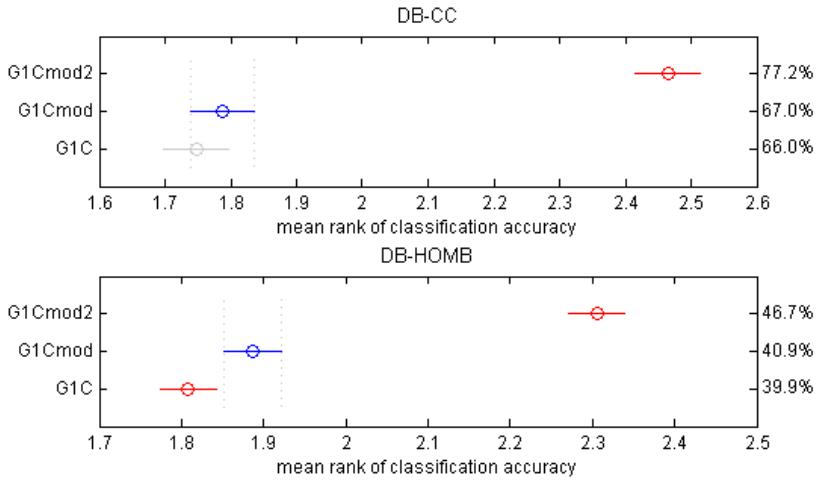

Figure 25: Evaluations on music collections that were *not* used for optimizing the algorithm. Right: Average percentage of items within the 20 nearest neighbors that have the same genre as the seed, with artist filter.

In Table 5, genre classification accuracies on DB-HOMB are listed. When using DB-NORM for determining the normalization factors for DB-HOMB, classification accuracy drops slightly (1NN: 52.7%, 10NN: 56.1%). For this data set, initial experiments in [Homburg et al., 2005] report classification accuracies of up to 53.23%, and [Moerchen et al., 2006] report accuracies of up to about 55%, based on 10-fold cross validation, using feature selection and machine learning algorithms. Usually, one would expect such algorithms that are optimised for high classification accuracy (most notably, SVM based classification algorithms) to outperform simple kNN approaches. This notion reflects for example the results of the MIREX'07 audio genre classification task, where only one out of seven submissions was based on a kNN classifier, and this submission ranked lowest[20]. In the MIREX 2008 Genre Classification Task no submission was based on a kNN approach.

Based on this expectation, it is surprising that the accuracies of G1Cmod2 obtained with a kNN classifier are clearly above these previously reported accuracies. This might be seen as an indication that the applied features are in fact well suited to model music similarity, and that they are used in an

[20] http://www.music-ir.org/mirex/2007/index.php/Audio_Genre_Classification_Results

NN	1	3	5	10	20	50
G1C	45.0%	46.9%	49.4%	50.9%	51.0%	51.0%
G1Cmod	47.6%	49.7%	51.9%	53.1%	53.2%	51.9%
G1Cmod2	53.6%	56.2%	57.5%	**58.6%**	58.3%	56.2%

Table 5: Leave-one-out genre classification on DB-HOMB. Classification is based on majority voting. No artist filter was applied. With artist filter (AF), the highest accuracy drops from 58.6% to 57.7%. When using stratified 10-fold cross validation it is 58.2 (no AF).

appropriate way. Probably classification accuracy can be improved further when sophisticated machine learning algorithms are used on the features used in this thesis. However, such experiments are not done here because the focus of this thesis lies on music *similarity*, not on music *classification*.

2.9. Conclusion

Many things have been tried to improve the canonical MFCC / GMM algorithm, and there is the common assumption that there exists a glass ceiling that can not surpassed [Aucouturier and Pachet, 2004]. The work presented in this chapter indicates that after extensive evaluations, there was some success in improving the overall algorithm. The resulting algorithm performs significantly better than G1C and G1Cmod on the two music collections it was developed on, and on the two additional music collections it was tested on.

Work following the work presented in this chapter includes [Pohle et al., 2009], where we propose modifications to the way Fluctuation Patterns are computed. Combining these modified Fluctuation Patterns with the Gaussian component proposed here, we arrived at an algorithm that – relative to our evaluation setting – performs well both in the task of rhythm similarity computation *and* for general music similarity computation. We submitted this algorithm to the MIREX'09 Audio Similarity and Retrieval task, where it ranked first[21].

[21] results can be found at http://www.music-ir.org/mirex/2009/index.php/Audio_Music_Similarity_and_Retrieval_Results

3. Similarity of Web Pages

In Chapter 2, techniques have been discussed to estimate how similar two music tracks sound by analyzing their audio signal. It is common sense that there is more to music than the plain audio signal. For example, the associations evoked by lyrics are not captured by audio analysis. Also, different kinds of popular music are commonly associated with particular subcultures. These associations may have an impact on the user's musical taste, and may influence (and thus be reflected within) human-generated recommendations. The Internet offers rich sources to obtain such recommendations, opinions and socio-cultural information, which is for example reflected in music reviews. This kind of data can be retrieved (semi-)automatically (e.g., by using a search engine, or specialised web services), and text information retrieval techniques can be used to extract terms with associated weights. Such a *vector space of descriptive textual terms* is called "community metadata" [Whitman and Smaragdis, 2002, Whitman and Lawrence, 2002]. Community metadata can be used to complement audio-based techniques.

This chapter starts with a review of a number of ways to obtain data relevant for music retrieval from the Internet. Then, the focus is laid on an approach to automatically collect information about similarity of music artists. By querying a search engine, a number of web pages is collected for each artist, and the subsequent use of text mining techniques allows for computing a similarity score between two artists. The main contribution of this chapter is that a large number of possible variants known from text information retrieval is evaluated for their potential to compute an appropriate similarity score between music artists based on this data. These experiments lay the foundation for Chapter 5, where a step is taken into the direction of automatically deriving semantically meaningful concepts to describe various kinds of music artists, and thus music itself.

3.1. Approaches to Collect User-Generated Metadata

In this section, a number of approaches to collect music metadata based on human knowledge and opinions is presented.

3.1.1. Explicit Data Collection

The most straightforward way to collect information about artist similarity, or related information such as genre, is by letting people explicitly deliver it. For example, [Berenzweig et al., 2003] present a user survey asking people about how similar they rate artists of a set of 400 artists.

Another explication of musical knowledge are expert opinions. For example, on *allmusic.com* for an artist a list of similar artists is provided, and a list of genres the artist is assigned to. In a number of publications, such expert opinions have been used as ground truth to evaluate automated approaches (e.g., [Berenzweig et al., 2003]).

In recent years, *tagging* has become more popular. For example, the online music platform *last.fm* lets users assign tags to pieces of music, or music artists. This tag data is made available over an API. Another approach is to collect tags in the form of a game [Law et al., 2007, Mandel and Ellis, 2007, Turnbull et al., 2007, Turnbull et al., 2008]. The basic principle of the game [Ahn and Dabbish, 2004] is to present the same item (which is a song in this case) to two different people, asking them to provide tags. Points are rewarded when both users provide matching tags. Tags that are proposed multiple times are taken as valid annotations for the item.

3.1.2. Playlists and User Collections

While explicitly asking people can be a source of high-quality data, it is time-consuming to participants. A less time-consuming alternative to obtain certain types of information about music is to analyze user data, such as which music users have in their music collection, and how often they listen to which music. Presented in [Whitman and Lawrence, 2002], "peer-to-peer similarity" is a way to calculate artist similarity based on the number of users that have a particular artist in their music collection (determined via OpenNap[22]), compensating for artist popularity. Alternatively, playlists can be analysed for tracks or artist co-occurrence patterns [Logan et al., 2003, Stenzel and Kamps, 2005]. Playlists created by human users can be obtained e.g. from *artofthemix.org*.

[22] http://opennap.sourceforge.net/

3.1.3. Text Information (Web Pages)

Analyzing playlists and user collections can give insight about how similar two artists (or tracks) are. Another source for similarity estimation is offered by the use of *text mining techniques* that create a term weight vector for each item (artist or song) that represents the weights of the terms in a dictionary. As most of the text based work in this thesis is based on this method, the foundations are discussed in more detail in the following.

3.2. Obtaining Community Metadata by Text Information Retrieval

The process of obtaining community metadata by using text information retrieval techniques can typically be divided into three phases: *data acquisition*, *data analysis* and *usage*, which are discussed in the next sections.

3.2.1. Data Acquisition.

In the first step, data acquisition, relevant text documents for each item are retrieved. For example, [Baumann and Hummel, 2003,Knees et al., 2004,Whitman, 2005] use a web search engine to find relevant text documents for an item (e.g., artist), and retrieve the top ranked documents for each item. To narrow the search and increase the number of relevant documents, it is common usage to send a query string like `"artist name" music review`. [Whitman and Ellis, 2004] retrieve documents from particular web sites (All Music Guide, Pitchforkmedia.com). Such a limitation has the advantage that more assumptions can be made about the kind of data returned. For example, the retrieved data can be better focused to record reviews, and specific assumptions about page structure can be made which facilitates parsing of relevant information. Limiting the data source to specific web sites has the potential drawback that the chosen web sites have a focus on particular communities and thus may contain biased data (record reviews written only by members of a specific social group, for instance). Other potential sources include RSS feeds [Celma et al., 2005, Celma et al., 2006], and the API offered by last.fm[23], where for example weighted tag lists for a query artist can be obtained.

[23] http://www.lastfm.com/api

3.2.2. Text Data Analysis and Processing.

The data acquired in the previous step usually has the form of a natural language text, or a HTML-formatted web page. In order to distill the desired information (such as a vector space representation of items) out of this, techniques of text mining and natural language processing can be applied. [Whitman and Lawrence, 2002] extract *unigrams* (single words occurring in the texts), *bigrams* (pairs of words following each other in the texts), words that are likely to be *adjectives* (by applying a Part-of-Speech (POS) tagger), and *noun phrases*. Each of these forms a possible basis for a vector space, where each term (e.g., bi-gram) is one dimension. In [Pampalk et al., 2005a], as an alternative to generating the space out of the retrieved documents, a predefined dictionary of words is used that are meaningful in the music domain. To cope with different forms of the same word, a stemming algorithm can be used [Celma et al., 2006, Schnitzer et al., 2007] at the expense of potentially introducing ambiguities. In many cases, words that are very frequent (such as *the, I*) and thus are assumed to not carry a meaning in the particular domain are removed by using *stop word lists*.

The actual value a particular artist is assigned in each dimension of the space over the terms is obtained based on the frequency with which the term occurs in documents related to this artist (*term frequency*), and typically is normalised by the frequency the term occurs in the collection of text documents (*document frequency*). The resulting vector is generally referred to as TFxIDF vector. In Section 3.3 a number of realizations of this general scheme is listed.

The above procedure is used to create a TFxIDF vector space from the retrieved documents. However, there are other scenarios where other representations are used. For example, for the task of artist recommendation, in [Cohen and Fan, 2000] lists of artists are extracted from web pages to eventually construct pseudo-users for a collaborative filtering approach. In [Pachet et al., 2001], texts are analysed for the occurrence of track and artist names to facilitate co-occurrence and correlation analysis for similarity computation. [Schedl et al., 2007] combines named entity detection and a rule-based detection approach to detect band memberships.

3.2.3. Usage.

In a number of cases, data usage is tightly coupled with the previous steps (i.e., data retrieval and processing are chosen and designed with a particular

application in mind). Instances of such applications have already been mentioned in the previous section. However, some of the data representations can be used for a variety of applications. Most notably, if a similarity function can be built on the extracted data, potential data usages include clustering, classification and recommendation (cf. Chapter 2). Besides genre classification [Knees et al., 2004], it has been proposed to classify record reviews into classes of like or dislike [Hu et al., 2005], which eventually could be used to create recommendation systems with improved recommendation performance, e.g., by using only those record reviews that are known to be in line with the user's (dis)likes.

3.2.3.1. Combining signal based and text information retrieval techniques. Audio signal analysis techniques and text information retrieval methods deliver in many ways complementary results. For example, signal-based techniques do not capture sociocultural aspects of music, but there is no necessity for them to first collect review information, tags, or user ratings for a new item before the item can be recommended (the early-rater problem [Avery and Zeckhauser, 1997], cf. [Celma, 2008]). So it is a consequent approach to combine both methods.

For example, work on automatically labeling previously unseen music tracks with terms based on signal and text analysis is presented in [Whitman and Ellis, 2004, Turnbull et al., 2006]. Eleven different *autotagging* systems have been evaluated in the "Audio Tag Classification" task of MIREX'07. For a brief review, see e.g. [Barrington et al., 2009].

Search Sounds [Celma, 2008] is an application that lets users search for audio by text queries, and refine the query by finding similar sounding music. The audio was retrieved along with textual descriptions by using RSS feeds that point to mp3 blogs, and both analysed by signal and text analysis methods.

3.3. Evaluating different variants of TFxIDF and distance measures

As outlined above, it is a common technique to obtain descriptions of artists by analyzing the text of web pages returned by a search engine queried with the artist name (and additional query terms to narrow the search to pages more relevant for the domain of music). This "search engine" approach has several advantages. For example, the obtained data can be used in different ways (similarity computation, assigning tags), and this approach does not crucially

depend on the availability of a specific online platform providing the particular type of data some approaches are built upon. Current trends in music (e.g., emerging genres) are likely to be reflected in the returned pages quickly. Furthermore, future advances in search engine technology (finding more relevant pages related to an artist) can be expected to enhance the results.

In this chapter, experiments related to this search engine based approach are presented[24]. The main topic of this chapter is how different algorithm variants compare against each other for calculating artist similarity, measured by genre classification experiments (for a motivation to use genre classification performance as an indicator of similarity see Chapter 2). It is assumed that those variants that perform well in this task capture information about the artists in a more "meaningful" way than the other evaluated variants. Based on the results of these experiments, a basis representation of artists in vector space is chosen for the work presented in Chapter 5, where an approach is discussed to automatically derive human-understandable concepts from TFxIDF vectors to facilitate a better description and retrieval of artists.

The remainder of this Section is organized as follows. First, some exemplary incarnations of the transformation from web pages to a vector space representation of artists are described in Section 3.3.1. The choice of the variants evaluated in this thesis is pointed out in Section 3.3.2. Finally, evaluations on two different sets of artist names and evaluation results are discussed in Sections 3.3.3 and 3.3.4.

$\mathcal{D}$	set of documents
N	number of documents
$f_{d,t}$	number of occurrences of term t in document d
f_t	number of documents containing term t
F_t	total number of occurrences of t in the collection
$\mathcal{T}_d$	the set of distinct terms in document d
f_d^m	the largest $f_{d,t}$ of all terms t in d
f^m	the largest f_t in the collection
$r_{d,t}$	term frequency, see Table 7
w_t	inverse document frequency, see Table 8
W_d	document length of d

Table 6: Denominations for terms in text information retrieval (cf. [Zobel and Moffat, 1998])

[24] This is joint work with Markus Schedl.

3.3.1. Approaches in Previous Work

A look in the literature reveals that there exist different ways the general framework discussed in Section 3.2 is applied to transform web pages to a vector of term weights for artists. For example, differences lie in the way basic concepts of text information retrieval, most notably the concept of a *document*, are transferred to music artists that are represented by a number of web pages. In the following, we use the denominations listed in Table 6 to refer to various terms of this domain.

[Whitman and Lawrence, 2002, Whitman, 2005] treat each artist as one document for calculating the document frequency (f_t), while term frequency ($f_{d,t}$) is the percentage of web pages containing the term. Both f_t and $f_{d,t}$ are normalised, being considered a probability density function (f_t are normalised after summing up over all artists, while $f_{d,t}$ are normalised for each artist), then TFxIDF is computed and normalised for each artist individually in the range 0..1, the smallest value being zero. Optionally, very frequent and very infrequent terms are downweighted by a Gaussian function. The similarity of two artist vectors is calculated by summing up the term weights of terms occurring for both artists.

In [Baumann and Hummel, 2003, Knees et al., 2004], $f_{d,t}$ is the number of occurrences of term t on the web pages related to an artist d, and the document frequency f_t is the number of web pages the term occurs on (not the number of artists for which the term occurs).

[Baumann and Hummel, 2003] and [Knees et al., 2004] differ in the way N is defined and the TFxIDF vector is calculated, while both use the cosine similarity measure to compare artist vectors. [Baumann and Hummel, 2003] defines n as the "size of the entire artist collection", and TFxIDF is computed by

$$w_{d,t} = f_{d,t} \cdot \log\left(\frac{n}{f_t}\right) \tag{48}$$

In [Knees et al., 2004], N is the total number of pages that were retrieved. For TFxIDF computation the ltc variant is used:

$$w_{d,t} = \begin{cases} (1 + \log_2 f_{d,t}) \log_2 \frac{N}{f_t} & \text{if } f_{d,t} > 0, \\ 0 & \text{otherwise} \end{cases} \tag{49}$$

The cosine similarity measure seems a commonly used way to calculate the similarity of two artist vectors. However, as motivated in the above examples, there is no standard way to calculate TFxIDF vectors from retrieved web

pages, and it is unclear which way to calculate it is preferable. In the next section, a number of variants to obtain TFxIDF vectors (and how they can be compared) is evaluated to gain some insight into this question.

3.3.2. Evaluation Approach

To evaluate a large amount of possible ways to calculate artist TFxIDF vectors and the similarity between them, we opt for an approach comparable to [Zobel and Moffat, 1998]. A large number of possible combinations of various ways to compute different parts of the algorithm (such as the term frequency $r_{d,t}$, the inverse document frequency w_t, and the combining function $S_{q,d}$) are evaluated. Most ways to compute the parts origin from previous work in text information retrieval. We adopt this systematic approach at the benefit that a large amount of previous work in text information retrieval is incorporated in these experiments. This goes at the expense not to exactly reproduce previous work in the domain of Music Information Retrieval, but may deliver insights for future work in this domain.

3.3.2.1. Document modeling. As already pointed out, the most central step is the modeling of fundamental text information retrieval concepts such as documents and term frequencies. Once this step is accomplished, known methods to calculate TF and IDF can be evaluated. In a common retrieval task, each document is considered a separate entity. In contrast, in our task each artist is an entity which is represented by a number of documents (i.e., web pages). There are several ways how to deal with this situation. We evaluate five of them.

1. *sum.* All term frequencies appearing in the web pages associated with the artist are summed up. This corresponds to a simple concatenation of all web pages related to the artist to one long document.

2. *mean.* The term frequency of a term is calculated by taking its mean over all pages. This is similar to approach 1 but with the difference that it is independent of the number of web pages actually retrieved, and a different range of values which has an impact on some TF calculation approaches.

3. *max.* Take the maximum of each term frequency over all retrieved web pages.

4. *numPagesRel.* Following [Whitman, 2005], the number of web pages containing a term is used as term frequency. This number is divided by the number of pages retrieved for the artist.

5. *numPagesAbs.* As approach 4, but with the absolute page count, which has an impact on some TF calculation approaches.

We refer to the representation that results from merging a number of web pages retrieved for an artist as *virtual document*.

	Description	Formulation
A	Formulation used for binary match. SB = b	$r_{d,t} = \begin{cases} 1 & \text{if} t \in \mathcal{T}_d \\ 0 & \text{otherwise} \end{cases}$
B	Standard formulation. SB = t	$r_{d,t} = f_{d,t}$
C	Logarithmic formulation.	$r_{d,t} = 1 + \log_e f_{d,t}$
C2	Alternative logarithmic formulation suited for $f_{d,t} < 1$.	$r_{d,t} = \log_e(1 + f_{d,t})$
C3	Alternative logarithmic formulation as used in *ltc* variant.	$r_{d,t} = 1 + \log_2 f_{d,t}$
D	Normalized formulation.	$r_{d,t} = \frac{f_{d,t}}{f_d^m}$
E	Alternative normalized formulation. As [Zobel and Moffat, 1998] we use $K = 0.5$. SB = n	$r_{d,t} = K + (1 - K) \cdot \frac{f_{d,t}}{f_d^m}$
F	Okapi formulation. For W we use the vector space formulation, i.e., the Euclidean length.	$r_{d,t} = \frac{f_{d,t}}{f_{d,t} + W_d / av_{d \in D}(W_d)}$

Table 7: Evaluated approaches to calculate the term frequency $r_{d,t}$, cf. [Zobel and Moffat, 1998].

3.3.2.2. Page length normalization. Based on the idea that web pages with many terms (i.e., long web pages) could dominate shorter but nonetheless relevant web pages, additionally an optional normalization step was done before these aggregation functions are calculated. To minimize interference with the TF calculation approaches (that may depend on the magnitude of the values), the number of terms in each page was normalised to the *average unnormalised page length* (as measured by the sum over the document's raw term frequency count vector). This optional normalization step was done

before calculation of the TFs because it intends to simulate pages of same length.

It has to be noted that there is another interesting method to combine the web pages of one artist. It would be possible to calculate the TFxIDF value for each web page separately (i.e, in the initial setup, each web page corresponds to one document), and then combine all pages belonging to one artist by a simple aggregation function such as minimum, mean, median or maximum. We refrain from this method because the notion our method is based on is to level out page length. This alternative way to combine pages could rather be seen as an attempt to level out different relevances of the retrieved pages. Differing web page relevance is not considered in our evaluations, as the retrieval of relevant pages is delegated to the search engine.

3.3.2.3. Modeling Document Frequency. In the experiments, we opted to model (inverse) document frequency f_t in two ways. The first way is to take the way TF is calculated as a basis (i.e, N is the number of artists, and f_t is based on the "virtual documents", vd). The second way is to take the number of web pages as the number N of documents, and the calculation of f_t is based on individual web pages (wp).

3.3.2.4. Calculation of TF, IDF and their combination. In our experiments, eight different methods for calculating the term frequency $r_{d,t}$ are evaluated, as given in Table 7. Correspondingly, Table 8 gives the evaluated methods to calculate the inverse document frequency w_t. Table 9 lists the evaluated comparison functions. It should be kept in mind that it is likely the considered functions interfere with the (generally unknown) algorithm used by the search engine, and probably also with the query terms (cf. [Knees and Pohle, 2008]).

3.3.2.5. Chosen terms. In the literature, there exist a variety of ways to define the terms associated with each dimension of the vector space. To not further complicate the experiments, we opt for using a manually defined dictionary containing 1379 music-related terms. Assuming that the way of choosing the dictionary avoids common stop words and such terms that appear very infrequently, no downweighting of very frequent and very rare terms is done.

	Description	Formulation
A	Formulation used for binary match. SB = x	$w_t = 1$
B	Logarithmic formulation. SB = f	$w_t = \log_e \left(1 + \frac{N}{f_t}\right)$
B2	Logarithmic formulation used in *ltc* variant.	$w_t = \log_e \left(\frac{N}{f_t}\right)$
C	Hyperbolic formulation.	$w_t = \frac{1}{f_t}$
D	Normalized formulation.	$w_t = \log_e \left(1 + \frac{f_m}{f_t}\right)$
E	Another normalized formulation. SB = p	$w_t = \log_e \frac{N - f_t}{f_t}$
	The following definitions are based on the term's noise n_t and signal s_t.	$n_t = \sum_{d \in \mathcal{D}_t} \left(-\frac{f_{d,t}}{F_t} \log_2 \frac{f_{d,t}}{F_t}\right)$ $s_t = \log_2(F_t - n_t)$
F	Signal.	$w_t = s_t$
G	Signal- to noise ratio.	$w_t = \frac{s_t}{n_t}$
H		$w_t = \left(\max_{t' \in \mathcal{T}} n_{t'}\right) - n_t$
I	Entropy measure.	$w_t = 1 - \frac{n_t}{\log_2 N}$

Table 8: Evaluated approaches to calculate the inverse document frequency w_t, cf. [Zobel and Moffat, 1998].

	Description	Formulation
A	Inner product.	$S_{q,d} = \sum_{t \in \mathcal{T}_{q,d}} (w_{q,t} \cdot w_{d,t})$
B	Cosine measure.	$S_{q,d} = \frac{\sum_{t \in \mathcal{T}_{q,d}} (w_{q,t} \cdot w_{d,t})}{W_q \cdot W_d}$
E	Alternative inner product.	$S_{q,d} = \sum_{t \in \mathcal{T}_{q,d}} \frac{w_{d,t}}{W_d}$
F	Dice formulation.	$S_{q,d} = \frac{2 \sum_{t \in \mathcal{T}_{q,d}} (w_{q,t} \cdot w_{d,t})}{W_q^2 + W_d^2}$
G	Jaccard formulation.	$S_{q,d} = \frac{\sum_{t \in \mathcal{T}_{q,d}} (w_{q,t} \cdot w_{d,t})}{W_q^2 + W_d^2 - \sum_{t \in \mathcal{T}_{q,d}} (w_{q,t} \cdot w_{d,t})}$
H	Overlap formulation.	$S_{q,d} = \frac{\sum_{t \in \mathcal{T}_{q,d}} (w_{q,t} \cdot w_{d,t})}{\min(W_q^2, W_d^2)}$
I	Euclidean similarity.	$D_{q,d} = \sqrt{\sum_{t \in \mathcal{T}_{q,d}} (w_{q,t} - w_{d,t})^2}$ $S_{q,d} = (\max_{q',d'}(D_{q',d'})) - D_{q,d}$
J	Jeffrey divergence based similarity.	$S_{q,d} = (\max_{q',d'}(D_{q',d'})) - D_{q,d}$, where D is computed according to Equation 21.

Table 9: Evaluated combining functions $S_{q,d}$, cf. [Zobel and Moffat, 1998].

3.3.2.6. Closest Models to Previous Work. To give a rough orientation how the evaluated techniques are associated with previously used combinations, the closest models to [Whitman and Lawrence, 2002, Baumann and Hummel, 2003, Knees et al., 2004, Whitman, 2005] are given here:

The closest model to [Baumann and Hummel, 2003] is BB2Bvd, and the closest to [Knees et al., 2004] is CB2Bwp, with the differences of a different logarithm base. Approach BCAvd is the one most similar to [Whitman and Lawrence, 2002, Whitman, 2005]. When calculating TFxIDF by $\frac{f_{d,t}}{f_d}$ and all values are nonnegative, then separate normalization of f_t and f_d is equivalent to a common normalization of the result. In [Whitman and Lawrence, 2002, Whitman, 2005] this common result is normalised again in the range $[0, 1]$. As we assume that each artist has at least one term with weight zero, this is equivalent to another scaling (to a maximum value of 1). Differences lie in the way terms are defined, and in the formulation of the combining function (sum of term weights vs. sum of the product of corresponding term weights).

3.3.3. Experiments

Experiments are performed on two sets of artists. The first set consists of 323 names of artists that are assumed to be among the best-known artists from their respective genre. From each genre, approximately the same number of artists was selected.

The second set, which is more than nine times as large as the first set, are 3000 artist names randomly selected from 50.000 artists without considering the number of artists per genre. Consequently, this set is assumed to contain many artists from the "long tail", i.e., rather unknown artists, and genre sizes differ largely.

It is assumed that the best performing TFxIDF approaches will do well on both sets. This results in two stages of experiments. In the first stage, all variants are evaluated on the first set. Only the algorithm variants found to perform best in these experiments on the 323 artist set are then evaluated on the larger set in the second stage.

Both sets of artists are divided into the same genre categories (the number of artists of the two sets in each genre is given in parentheses): *avant garde* (19/22), *blues* (20/65), *celtic* (12/19), *classical* (17/54), *country* (15/81), *easy listening* (18/23), *electronica* (18/182), *folk* (19/66), *gospel* (18/118), *jazz* (19/297), *latin* (15/137), *new age* (17/62), *rnb* (20/107), *rap* (20/133), *reggae* (20/41), *rock* (20/1301), *vocal* (19/55), *world* (17/237).

3.3.3.1. First Stage: Evaluation on 323 artists set. We model the experiments as a retrieval task. In some major aspects, we follow [Buckley and Voorhees, 2000, Sanderson and Zobel, 2005]. Given a query artist, the task is to find artists that are similar to the query artist with respect to the genre that is assigned to each artist. We use Mean Average Precision (MAP) as the basic performance measure. Average Precision is "the mean of the precision scores obtained after each relevant document is retrieved, using zero as the precision for relevant documents that are not retrieved" [Buckley and Voorhees, 2000]. Following [Sanderson and Zobel, 2005], we first calculate MAP of each distinct algorithm variant (i.e., considering that some combinations yield the same results, e.g., TF_A with and without normalization of pages). These are 9120 variants. Variants that fulfill both of the following two conditions are discarded:

1. there is a relative MAP difference of 10% or more to the top ranked variant,

2. and the t-test shows a significant difference to the top-ranked variant.

When doing so, and subsequently ranking all 9120 variants according to MAP, the top 123 variants have a relative MAP difference of less than 10%. The t-test shows significance for all variants except for the top 134 and the 136th ranked variant. This sharp cutoff of non-significant vs. significant results and the relatively high accordance of our two criteria (less than 10% MAP difference and significance) makes us confident that our reasoning is valid, and that these top-ranked algorithms are those worth examining further.

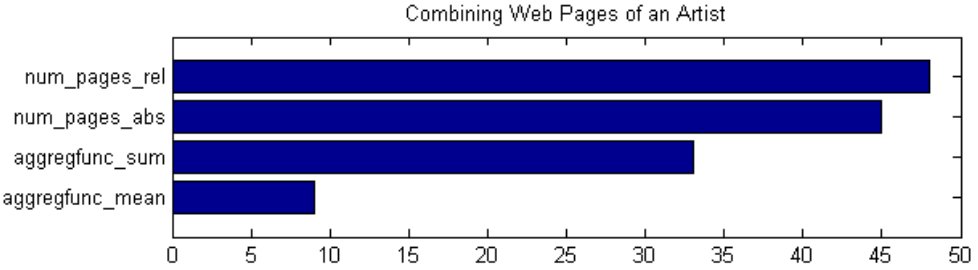

Figure 26: Methods to combine terms appearing on an artist's web pages. Only those appearing in the 135 selected top algorithms are shown, and the number of times they appear (totaling 135).

To get more insight into which components are of high value, we look at each of the algorithm's components separately, and examine which approaches appear in these 135 selected algorithms, and how often they appear. First, it gets apparent that only variants based on *un*normalised web page lengths appear in the top-ranked variants. Thus, normalization seems not to contribute to the performance. Also, only IDF computation approaches based on virtual documents are encountered. I.e., calculating the inverse document frequency on web pages instead of artist level seems not beneficial.

Figures 26 to 29 contain histograms of the other algorithm components, for which several variants appear. They give a first weak insight into the relative performance of variants. The algorithm representing the most frequently appearing variant of each component (i.e., numPagesWithTermRel – TF_C2 – IDF_I - cosineSimilarity) is ranked top 21 in the overall ranking.

However, it can not be assumed that the shown frequencies are mutually independent. For example, when for one algorithm part two highly similar variants are evaluated, the other algorithm parts that perform well in combination with these variants will appear more frequently.

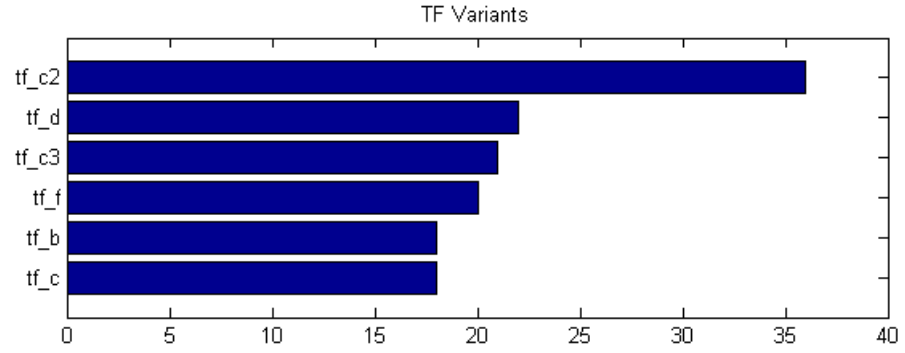

Figure 27: TF approaches appearing in the 135 selected top algorithms, and the number of times they appear (totaling 135).

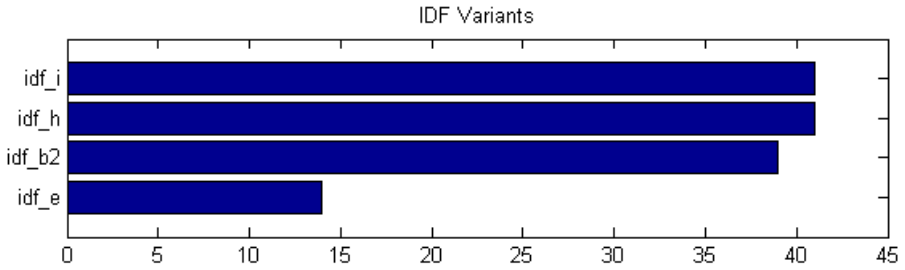

Figure 28: IDF variants appearing in the 135 selected top algorithms, and the number of times they appear (totaling 135).

So instead of analyzing the figures more deeply, we go on by evaluating all possible algorithms that can be created with the variants appearing in these figures on the second set consisting of 3000 artists. Thus, the only assumption is that variants that do not appear in the 135 selected algorithms are not well suited as a part of the kind of algorithm we are interested in. In detail, additionally to normalizing web page length and calculating document frequency on the web page level, the variants that are discarded here are:

- *Document modeling: max*, i.e., taking the maximum number of appearances over all web pages of an artist.
- *TF computation:* Variant *A* (*binary match*, i.e., if a term is contained in a document or not) and variant *E* ("alternative normalised formulation").
- *IDF computation:* Variants *A, B, C, D, F, G* (cf. Table 8).
- *Similarity Measure:* Variant *A* (inner product), *E* (alternative inner product), *H* (overlap formulation), *I* (Euclidean similarity).

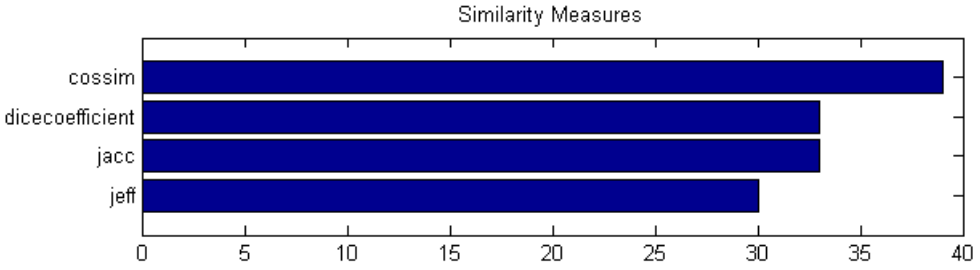

Figure 29: Similarity measures appearing in the 135 selected top algorithms, and the number of times they appear (totaling 135).

The remaining variants can be used to create 384 different combinations of TFxIDF approaches and similarity measures.

3.3.3.2. Second Stage: Evaluation on 3000 artists set. In the second stage of the experiments, the 3000 artist set is used for evaluation, although the obtained MAP values are much lower for this set (0.007 compared to 0.387 for the respective highest ranked algorithm), which presumably is an effect of having less popular artists and different genre cardinalities in this set. It is evaluated if both artist sets yield a comparable ranking of the 384 algorithms of interest, and which of these algorithms are top-ranked on both sets of artists. To clarify the first aspect, Spearman's rank-order correlation coefficient is computed on the two rankings on the two artist sets. This experiment shows a correlation coefficient of **0.82**. This relatively high value indicates that in general, the ranking of the algorithms is not largely influenced by factors such as size of artist collection, number of artists per genre, and artist popularity.

To get an insight into which out of the 384 algorithms are top-ranked on both sets of artists, a ranked list of all algorithms is created. In this list, algorithms are sorted based on their *maximum rank in either of the two experiments*. For example, if an algorithm ranked second in the algorithm ranking based on the set of 323 artists, and 15^{th} on the set of 3000 artists, then the value associated with this algorithm is 15. The full list is given in Appendix A.1.

As can be seen from the list, the TFIDF algorithm used in [Baumann and Hummel, 2003], computed on our term data, has a maximum rank of 306. The algorithm used in [Knees et al., 2004] does not appear in the list, as it uses the number of web pages to determine the document frequency, which was outside the significance bounds in the first stage. However, approach sum.tf_c3.idf_b2.cosSim which has a maximum rank of 15 in the two experiments resembles this algorithm (counting the number of artists instead of

counting the number of web pages a term appears on). This may be seen as an indication that the LTC variant is a good choice for the considered area of application, and it also shows that changing only one factor can have an important impact on the performance of an algorithm. Based on the latter observation, it seems no valid statement about the relative performance of the algorithms used in [Whitman and Lawrence, 2002, Whitman, 2005] can be made, as the exact similarity measure used there was not evaluated in our experiments.

3.3.4. Conclusions

The experiments in this section have been conducted with two aims in mind. First, to get insight into how artist similarity can be computed based on socio-cultural metadata obtained from search engines, when nothing is available but the artist name. For this application, the actual quality of the similarity computed by an algorithm is of interest. The second aim of the work in this section was to prepare the experiments in Section 5, where the TFxIDF vectors are used as the starting point to create a higher-level representation of artists that capture parts of semantic aspects. For this scenario, it is of importance that the chosen TFxIDF representation is "robust", which is an ill-defined concept but is likely to be also reflected in the experiments conducted here.

3.3.4.1. Conclusions Concerning Artist Similarity. The conclusions for calculating artist similarity can be summarised as follows. A minor finding is that normalization of each web page so that each web page has the same weight showed not to be of benefit. It seems of more importance that the document frequency for calculating IDF should be determined on virtual documents rather than on individual web pages. The commonly used cosine similarity appears for many top-ranked algorithms.

- Factors concerning the *collection* such as size of collection, artist popularity, and numbers of artists per genre seem to have only a minor impact on the relative performance of the 'best' algorithms, as far as can be concluded from the evaluated parameter ranges.
- In contrast, a small change to an *algorithm* (document frequency calculated on web pages or on artist level) can have an important impact on the algorithm's (relative) performance.

The latter observation makes the future evaluation of different text processing, term selection and term weighting functions appear of importance.

3.3.4.2. Conclusions Concerning Experiments in Section 5. It seems reasonable to assume that a way of computing TFxIDF vectors that works well with several similarity measures is better suited as a basis for factorization experiments (see Section 5) than one that works only with a particular approach of calculating the similarity. Such a representation shows to be robust against the factor "similarity measure", and so it also may be robust against influences that are effects of factorization. A closer look on the list given in Appendix A.1 reveals that the top combinations seem not to be independent of the similarity measure. The approach that best conforms to this requirement is numPagesRel.tf_f.idf_b2. Thus, in Section 5, this will be compared against previously used algorithms.

4. Higher Concepts Derived from Audio Signals

In Chapter 2, techniques are discussed how to retrieve similar sounding music from a music collection. These techniques are based on an *implicit* model of music audio similarity. They are shown to work (and they are optimized) by empirical experiments, not by explicitly defining what specific aspects make two pieces be similar (or not be similar, respectively). In this chapter, which mainly follows the contents presented in [Pohle et al., 2006a], an approach is discussed to explicitly model the relevant content of the audio signal for similarity computations. In particular, a method is examined to use a data-driven approach applying Independent Component Analysis (ICA) to automatically derive building blocks of song spectrograms, and describe songs by such building blocks. While the resulting accuracies are not above those obtained in the experiments presented in Chapter 2, this approach might be a step towards modeling the human view in a more sensible way. Preliminary experiments indicate that when changing parameters such as the spectrogram representation, and using Non-Negative Matrix Factorization (NMF) instead of ICA, results obtained in the used evaluation setting can be improved .

4.1. Approach

The basic idea for the experiments presented in this section is adopted from work in image similarity computation [Le Borgne et al., 2004], where images are represented as activations of individual components. The components are found in a data-driven way by applying an Independent Component Analysis (ICA) algorithm on a large number of *patches* of size 12×12 pixels randomly chosen from the images. An image is analyzed by considering each of its patches as a linear combination of those components that are most highly activated over a large number of images, and taking the histogram of component activations as feature data.

The authors of [Le Borgne et al., 2004] evaluate a number of methods to calculate image similarity from this feature data. As the components found by ICA can be assumed to be sparsely activated and uncorrelated, modeling of the interdependencies of component activations can be disregarded.

1. *Mean.* The first approach reduces the feature data to a vector of length n, where n is the number of considered components. The vector holds the mean of the activation of each component over all patches of an

image. Two vectors (i.e., images) are compared by taking the Euclidean distance.

2. *KL* and *KL_zero*. A more fine-grained modeling of a component's activation is done by calculating the mean and variance of its activations over all patches of the image. Two images are compared by taking the KL distance between these models. To take into account the high kurtosis of the component activation histograms, an alternative way of comparison is to mirror the histograms at 0 and only consider the variance of the resulting distribution (as the resulting mean is zero, this approach is here denoted as *KL_zero*).

3. *Spline*. Finally, the authors evaluate a density estimation by B-Splines, in which case the area common under both splines is taken as the indication of similarity.

The similarity measures are evaluated with a nearest-neighbor approach on an image collection consisting of 540 images divided into four categories, based on the notion that images within a category are more similar than images from different categories. The authors obtain classification accuracies of up to 87%.

4.2. Application to Audio Signals

One might hypothesize that the components found by an approach as described above are related to meaningful building blocks underlying the observed data. [Abdallah and Plumbley, 2001] report that ICA applied to time-domain signals can find wavelet-like bases. In [Smaragdis, 2001, Abdallah, 2002] individual frequency-domain frames have been analyzed by ICA to obtain sparse-coding representations.

When the data is the music signal represented in a spectrogram-like form, and patches consist of several frames [Cho et al., 2003], one can hypothesize that such building blocks might be acoustically meaningful entities, such as percussive onsets, sustained notes in various frequency bands, or noises of different shape. Detecting such entities can be useful in analyzing the audio signal, as has already been shown in Chapter 2 for Harmonicness and Attackness. Using an automated approach to define and detect such entities might result in algorithms that outperform manually defined concepts such as Harmonicness and Attackness.

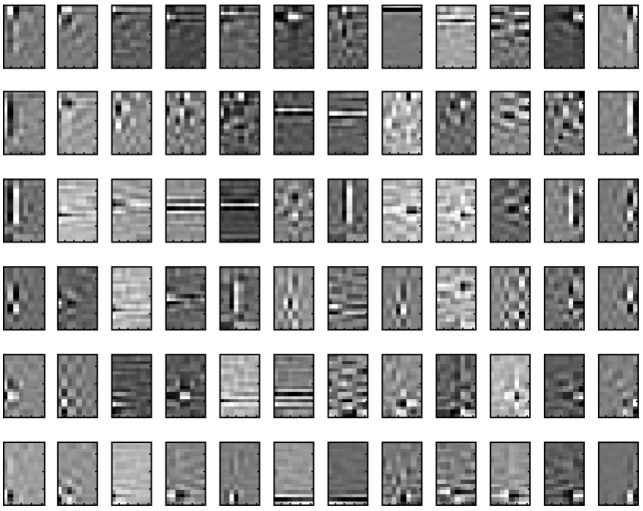

Figure 30: Components found by ICA on randomly chosen patches of length 0.15 sec with 50% PCA compression applied [Pohle et al., 2006a]

4.3. Experiments

To apply the discussed approach to audio spectrograms, it is necessary to reduce the number of frequencies in order to retain computational feasibility. For the experiments in this section, the number of frequencies was reduced to 18 Mel scaled bands (cf. Figure 2). Each patch was chosen to cover all frequencies. Considering the duration of quarter, $\frac{1}{8}$ and $\frac{1}{16}$ notes of a $\frac{4}{4}th$ metre played at 100 bpm, initial patch lengths were chosen to be of lengths 0.6, 0.3 and 0.15 sec. The *FastICA*[25] algorithm was run on patches of corresponding size. Initially, the number of calculated components is equal to the number of input dimensions (which is number of frames × number of frequencies). To avoid obtaining too many components, additionally PCA compression was applied on the patches before ICA computation (cf. [Le Borgne et al., 2004]). This is done by stacking all values of a given patch into a vector, applying PCA on the vectors obtained from all patches, and only retaining a certain percentage of principal components that have the largest eigenvalues associated. ICA is performed on this data with reduced dimensionality (thus, also the number of calculated independent components is reduced). After computing the components, PCA compression is inverted. Three levels of PCA compression were evaluated: No compression (i.e., ICA was run on the original patches), 50% and 75%. The experiments include three flavors of obtaining components:

[25] http://www.cis.hut.fi/projects/ica/fastica/

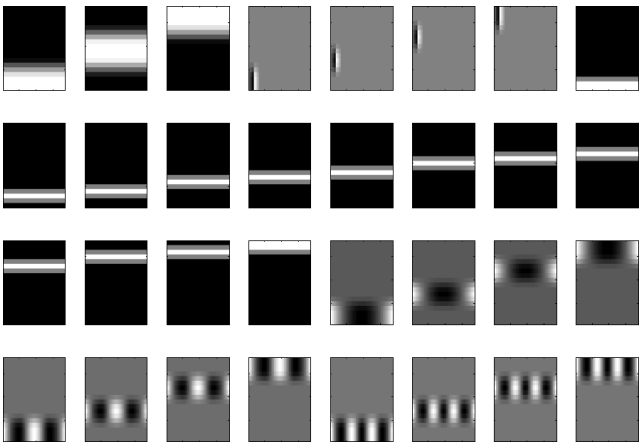

Figure 31: Manually defined components [Pohle et al., 2006a]

1. *Pat-Rand* Components were calculated on patches randomly chosen from 100 tracks from DB-MS.
2. *Pat-Onset* Components were calculated as in bullet point 1, but patches start at positions that are likely to be onsets, as detected by a simple onset detection algorithm. An example of the found components is given in Figure 30.
3. *Pat-Manualdef* Components are not determined by ICA, but defined manually, as depicted in Figure 31. These are defined in an ad-hoc way. Some of the components aim to capture similar features as Harmonicness and Attackness.

The components obtained this way are evaluated in several ways, as discussed in the next section.

4.4. Evaluation

Three ways seem useful for evaluating how well the components from the previous section are suited to meaningfully describe aspects of music. First, from the appearance of the components itself, one can asses the features they represent. Second, it is possible to examine which components are activated in which tracks, and if there are obvious correlations between the activation pattern of a particular component and certain audible events. Third, automated evaluation can be done by genre classification experiments. All three approaches are discussed below.

4.4.1. Assessing the Shape of Components

Visually inspecting the resulting components in most cases did not give clear indication that musically meaningful entities are represented. But in some cases, one might interpret so. For example, in Figure 30, components found from patches with a length of 8 frames with 50% PCA compression are shown. Some of these components seem to be related to percussive onsets (e.g., third row, first, seventh and penultimate column), and resemble the Attackness filter. Others have a horizontal orientation (e.g., components found in row/column 1/8, 2/6, 2/7, 3/5 etc.) resemble the filter used for Harmonicness computation. However, even in those experiments that exhibit such interpretable components, many components do not have an obvious interpretation, e.g., the one depicted in row 2, column 4.

4.4.2. Activations in Tracks

Examining the activations of individual components in a particular track was done in two ways. First, the activation of each component over time within a track was visually inspected. This was done on a small number of tracks expected to be suited for this task (e.g., pieces for prepared piano, that have various types of onset sounds, and choir music that is expected to have clear harmonics). These experiments did not reveal clear and strong relationships of audible events and component activations. The second way to examine component activations in tracks was to compare activation histograms. This approach also did in general not reveal clear and meaningful relationships between component activations and perceived audio entities. Only in few cases is there room for such interpretations. Unfortunately, such observations remain anecdotal.

4.4.3. Genre Classification

The evaluations discussed in the two preceding sections are only based on a small number of manually chosen tracks. To complement these experiments, and to do an evaluation in a larger scale in an automated way, the usefulness of the approach for music similarity computation is estimated by genre classification experiments. In a similar way as in [Le Borgne et al., 2004], component activation histograms are used as the basis to define a similarity function, as described in Section 4.1. Experiments were done on 30 second excerpts of DB-MS. Using the functions *mean*, *KL* and *KL_full*, this yields classification

accuracies of 61.5% to 68.5% for *pat-rand*. When changing the way to obtain components from *pat-rand* to *pat-onset* or *pat-manualdef*, results are not improved [Pohle et al., 2006a]. With a basic MFCC/GMM approach, on the same 30 second excerpts, accuracies of about 72% are obtained with an equivalent setup (30 sec excerpts from the center of each file [Pohle et al., 2006a, Pohle, 2005]). These figures do not indicate that when using the evaluated parameter settings, the discussed method is suitable to give qualitatively better results than the existing algorithms. Most notably MFCC/GMM based approaches, such as the one developed in Chapter 2, yield higher accuracies.

4.5. Changing Parameters

In the experiments from [Pohle et al., 2006a] that have been discussed up to this point, a variety of parameter settings have been evaluated. However, two aspects that may be of importance were left unchanged:

- First, the spectrogram is reduced to only 18 frequency bands. This may be too coarse a resolution to capture musically relevant structures, such as partials that typically appear as horizontal lines in a (higher-resolution) spectrogram.
- Second, ICA is applied to find components. ICA might be not the best algorithm for this purpose.

Consequently, instead of using an 18-band mel/sone representation, here a cent-scaled spectrogram is used, and ICA is replaced by Non-Negative Matrix Factorization (NMF, see Section 5.2.3.3). The experiments take two directions. Additionally to evaluating the genre classification accuracy obtained with these changed parameter settings, it is also examined if by this data-driven method, alternatives to the Harmonicness and Attackness descriptors can be found. This is discussed next.

4.5.1. Revisiting Harmonicness / Attackness

Seen in the context of this chapter, Harmonicness and Attackness detect specific features on patches of size 5×5. So it is of interest to examine which decomposition is found by the data-driven algorithm for patches of this size, when the spectrogram has similar parameters as the one used to calculate Harmonicness and Attackness. These experiments may indicate whether there are other features of interest that could be detected on patches of this size, besides

Harmonicness and Attackness, such as noise spread both over frequencies and time (i.e., oframes).

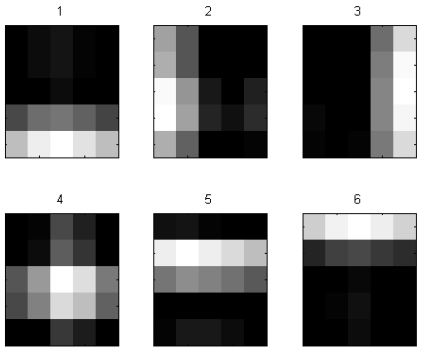

Figure 32: "Basis images" found by NMF on randomly chosen patches of size 5×5, based on cent spectrogram. Frequency bands are 66.7 cent apart. Frame rate is approximately 21.5 fps

In these experiments, the parameter r that denotes the number of basis images (or "components") to calculate has to be set. Based on visual inspection of the found basis images, non-systematic experiments seem to indicate that except for horizontal and vertical lines, apparently no particular features appear. An example with r set to 6 is given in Figure 32, calculated on randomly chosen patches from the tracks in DB-MS. With r set to different values, horizontal and vertical lines in many cases were less clearly visible. In particular, when setting r to values larger than e.g., 10, the tendency is observed that some basis images rather take the form of "spots", as e.g., in images number 2 and 4 in Figure 32. As a conclusion, these experiments do not seem to strongly indicate that Harmonicness / Attackness can be augmented with other features extracted from patches of the examined size.

4.5.2. Using Basis Images Found by NMF for Similarity Computation

Finally, it is of interest what basis images are found by NMF on larger patches that span a frequency range of several octaves. To this end, the same method as discussed above is applied. For reasons of data size, frequency bands are chosen to be about one semi tone apart, measured on the equally tempered scale (99.3 cent). In total, there are 86 frequency bands in the calculated spectrogram. To allow for a certain abstraction from the actual pitches, patch height is chosen to be 74, i.e., about one octave less than the total number of

frequency bands in the spectrogram. The number of frames is set to 4, which corresponds to a total patch length of about 0.18 sec.

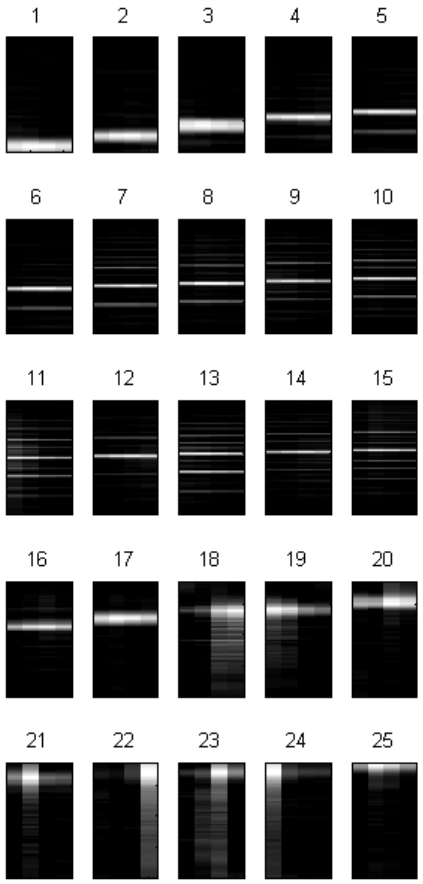

Figure 33: "Basis images" found by NMF, based on cent spectrogram. Size of each patch: 74×4. For display purposes, images are sorted according to the index of the row containing the largest values.

Again, the number r of basis images to compute is a user-provided parameter. In Figure 33, results for $r = 25$, computed on patches randomly chosen on tracks from DB-NORM are shown. It can be seen that with this parameter settings, many basis images are made up from horizontal structures (images 1 to 17) that likely correspond to partials. In particular, it is interesting to note that images 7 to 15 seem to reflect overtone structures. In contrast,

images 21 to 24 seem to contain distinct vertical structures, located at each of the four frames. Overall, compared to the components shown in Figure 30, these basis images seem more closely related to perceptually relevant aspects. In particular, some basis images resemble components defined in an ad-hoc manner, shown in Figure 31.

We use these basis images to extract features from a piece in the following way. We approximate[26] $H(W') = V'$, where W contains the basis images (e.g., those shown in Figure 33) in vector form, i.e., each column of W contains the frames of one basis image stacked above each other to form a vector. Likewise, each column of V is created from (e.g., four) consecutive frames of the cent spectrogram of the piece to analyze, and each row of the resulting matrix H contains the r feature values for one frame. Using the basis images shown in Figure 33 to extract features from DB-MS, and comparing tracks by *KL_full*, a leave-one-out genre classification accuracy of 76.13% is obtained (extracting features on the middle 30 sec of each file as above). [Pampalk, 2006b] reports a 1-NN accuracy of 74% for the $G1$ algorithm when calculated on 30 sec from the center of each file, using a sample rate of 22 kHz as in this section. This result indicates that these basis images may be better suited to represent aspects of the music than MFCCs (computed frame-wise) alone. Thus, this method may be a candidate to replace MFCCs in audio similarity computation. However, accuracy is still clearly below the one obtained by the G1C_mod2 algorithm reported in Table 4 (83.7%).

4.6. Conclusions

While the results from the experiments presented in [Pohle et al., 2006a] do not give clear indication that the evaluated method performs better than existing methods, we have shown in this Chapter that when changing parameters, results can be improved above the MFCC-only baseline (i.e., the result reported for $G1$ in [Pampalk, 2006b]) in our evaluation setting. As the bases found by this data-driven method may be associated with perceptually meaningful aspects of the audio signal (such as partials, see Figure 33), this method could be considered a candidate to replace MFCCs in future audio similarity measures that apply a more straightforward and understandable way of calculating the similarity.

There are two future research directions we currently think of: The first direction is to further analyse the degree to which the bases are activated over

[26] using the Matlab function mrdivide

time. For example, this can be done by analysing the periodicity as in Fluctuation Patterns, but using a basis image decomposition instead of frequency bands. And second, the way the basis images are defined can be modified to reflect more distinct features. For example, one can think of using individual instrument samples as the input source for decomposition, such as drum and instrument samples. Defining basis images in such a semi-automatic way may facilitate the creation of an algorithm that transcribes a given piece into musically meaningful acoustic basis elements.

5. Higher Concepts Derived from Web Pages

In Chapter 3 methods are discussed to determine the similarity of music artists based on web pages associated with each artist. This procedure yields a fixed similarity value for each pair of artists. Such fixed similarity values are useful in a variety of applications such as e.g. playlist generation and clustering of music collections. These artist-to-artist similarities implicitly are (at least weakly) linked to an artist's characteristics, i.e., two artists that share similar characteristics are likely to have a high similarity value associated.

Analogous to the approach in Chapter 4, it is in some cases favorable to have a more meaningful insight into why a pair of artists is considered similar (or not similar, respectively). For example, in recommender systems, it may be useful if the user can not only enter an artist name to query for similar artists, but also that the system allows to indicate the relative importance of aspects associated with the artist. When used appropriately, this additional information may help to increase the quality of the recommendations. For example, in [Lamere and Maillet, 2008], a system is presented that allows the user to modify the importance of individual tags for the recommendation.

In this chapter, a different method of text-based access is discussed. The idea is to automatically derive higher-level concepts from web pages associated with the artists. Such concepts are derived by applying dimensionality reduction techniques (most notably, Non-Negative Matrix Factorization (NMF)) to the TFxIDF vectors to reduce their length to a number r of few dimensions (e.g., $r = 16$) that are likely to represent meaningful "concepts". Such a resulting low-dimensional vector describes the degree to which the artist is associated with each of the r concepts.

Once artists are described by their association with different concepts, this association can be used in a variety of ways. One obvious use would be to search for artists by means of concepts, i.e., the user indicates which concepts he is interested in, and the system determines the artists most "similar" to this given constraint. Another use of concept annotation is to label individual artists that are unknown to the user with the most activated concepts to facilitate a quick impression about this artist. Also, given two artists, it is possible to automatically determine which concepts they share, and in which way they differ (by means of concept annotations).

5.1. Approach to Obtain "Artist Topics"

The TFxIDF vectors and the measures to calculate their similarity that are described in Chapter 3 are useful for determining the similarity of artists, but the high dimensionality of TFxIDF vectors sometimes hinders other usages. For example, the high-dimensional TFxIDF vectors facilitate a query-by-example ("suggest artists that are similar to the given artist"), but having a more detailed influence on the query would be desirable. Such an influence may be achieved by allowing the user to adjust the weight (or importance, respectively) of each term in the vector manually, so that she can determine how much she is interested in each term. A vector modified this way can be assumed to better reflect the user's query when used to determine similar vectors. However, selecting the subjective importance of each individual term in the TFxIDF vector is likely to be ineffective. For example, going through a vector of length 2000 would take about 15 minutes when on average two terms per second are set. This is clearly unfeasible.

To reduce the effort, the TFxIDF vectors can be reduced to a small number r (e.g., $r = 16$) of ideally meaningful dimensions by applying dimensionality reduction techniques. The user then can interact with such a vector in the compressed domain. Obviously, it is of importance that each of the r dimensions has a clear meaning, or concept, associated.

A number of dimensionality reduction techniques have been proposed in the past. Among these are Principal Component Analysis (PCA), Independent Component Analysis (ICA, cf. Chapter 4), and Non-Negative Matrix Factorization (NMF) [Lee and Seung, 1999]. We follow the approach from [Xu et al., 2003], where NMF was applied to cluster documents according to the contained topics.

NMF seems a good choice for this application because of the non-negativity of the factors. This seems to be a more natural representation than allowing for negative weighted terms (such as in PCA and ICA), as the negative weight does not have an immediately intuitive explanation. When applying NMF, both "concept" bases and the resulting compressed artist vectors have only non-negative weights. This allows for comparing (and querying, cf. Section 7.3) artists in the compressed domain, e.g. by the cosine similarity measure.

As presented in [Pohle et al., 2007a], we adopt the method presented in [Xu et al., 2003] by clustering artist TFxIDF vectors instead of document TFxIDF

vectors. Consequently, in our method, the found topics do not describe documents, but artists. NMF is done by finding an approximate decomposition

$$V \approx W \cdot H \tag{50}$$

of the non-negative matrix V into non-negative matrices W and H. V contains the artist TFxIDF vectors, i.e., it is of size $n \times m$ with n being the number of terms, and m is the number of artists in the collection. W is of size $n \times r$ and is interpreted as a codebook containing r "topics". Thus, each column of W describes the strengths of the associations of all terms with the topic that this column represents. H contains the "encodings" of each artist by means of how much each artist is associated with the respective topic.

The number r of topics is an important factor. While there exist methods to determine it in a data-driven way such as the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC), in the experiments reported here the number of components is fixed to 16, assuming that this corresponds roughly to the upper bound that humans easily can keep an overview of. It is assumed that if there are actually fewer topics in the data, then some of them will be split up to highly similar topics, or sub-topics.

5.2. Experiments

To examine the potential of the approach to describe artists, a number of experiments is conducted with varying parameters (three different dimensionality reduction methods, four different approaches to calculate TFxIDF vectors, and two variants of term dictionary / search engine combinations). These parameters are discussed in the following sections.

The different variants are evaluated by genre classification accuracy, and by manually inspecting the terms with highest weight in the found basis vectors.

The use of genre classification accuracy as an evaluation criterion is based on the assumption that very similar artists are in the same genre, and that very similar artists share the most meaningful concepts. Thus, genre classification experiments are assumed to indicate how well the respective projection is suited to find similar artists. If accuracy is high, and additionally the found concepts make sense, it is assumed that a good low-dimensional representation is found.

5.2.1. Chosen TFxIDF Algorithms

To get some indication of the influence of the way to calculate TFxIDF vectors, four methods are compared in the experiments presented in this section. Based on the insights gained in Chapter 3, except for the "robust" variant, we use as term frequency $f_{d,t}$ the sum of the occurrences of a term over all web pages retrieved for an artist, and for the document frequency $r_{d,t}$ of a term the number of artists on whose web pages the term occurs. The evaluated methods are:

- *TF.* Using the term frequency $f_{d,t}$ [Lee and Seung, 1999].
- *Like [Xu* et al.*, 2003].* [Xu et al., 2003] use $f_{d,t} \cdot \log(\frac{N}{r_{d,t}})$. Each TFxIDF vector is normalised to have a Euclidean length of 1.0. [Xu et al., 2003] reports improved results when using the NC weighted form *NCW* [Shi and Malik, 2000]. In [Xu et al., 2003], NCW is calculated by transforming the input matrix X to X' by $D = \text{diag}(X^T X e)$ and $X' = XD^{-1/2}$, with $e = [1, 1, ..., 1]^T$.
- *LTC.* Adapting the LTC variant [Salton and Buckley, 1988] used in [Knees et al., 2004], considering the concatenated web pages of an artist as one document [Pohle et al., 2007a]. This variant ranked high in the experiments presented in Chapter 3 when vectors are compared by cosine similarity, Dice formulation or Jaccard formulation (denoted *sum.tf_ c3.idf_ b2* in Appendix A.1).
- *"Robust".* Finally, we take what appeared to be the most robust approach against the change of the similarity function, which is variant *numPagesRel.tf_ f.idf_ b2* in Appendix A.1.

5.2.2. Approaches to Obtain Terms and Term Weights

We compare two methods to define term dictionaries and estimate the associated term weights. We use a subset of the artist names contained in the *C19959g* collection [Schedl, 2008].

1. *TD-EXA* The first method uses the search engine Exalead and the music dictionary from [Schedl, 2008], which consists of about 1400 music-related terms and was compiled from various sources [Schedl, 2008]. The corresponding full inverted index from [Schedl, 2008] created using

Lucene[27] was used. Only the artist names from the *C19959g* collection that were contained in this index were kept.

2. *AS-GO* The second method is the one followed in [Pohle et al., 2007a], where the search engine Google[28] is used, and web pages are searched for terms obtained from Audioscrobbler[29].

For comparability, the same set of artists was considered in the two cases. 292 terms are the same in both dictionaries (i.e., exact matches). The data was cleaned up by removing terms that occur for less than 2 artists and for more than 95% of the artists. For method 1, 1281 terms remained after this step. For method 2, 2045 terms remained.

5.2.3. Dimensionality Reduction Methods

Finally, three methods to reduce the vector length are compared against each other by comparing their classification performance, and by comparing it to the classification performance as obtained by the full-length TFxIDF vectors. For all these comparisons, cosine similarity is used as the similarity measure between vectors. The compared methods are clustering by linkage, PCA and NMF, as discussed next. All three methods are applied to compress the TFxIDF vectors to a length of 16.

5.2.3.1. Clustering by Linkage. The first (and probably most obvious) method to reduce the length of the TFxIDF vectors is to use a clustering algorithm. Each term is represented by its TFxIDF values over all artists. Based on this information, the clustering algorithm determines 16 clusters of terms and assigns each term to one of these clusters. An artist TFxIDF vector is projected to 16 dimensions by calculating for each cluster the mean of the TFxIDF values of terms belonging to the given cluster. To find the clusters, a deterministic procedure was chosen to avoid multiple runs. Based on preliminary experiments conducted for NMF initialization, the following procedure was adapted. A hierarchical cluster tree was created[30] and clusters were combined bottom-up until only 16 clusters remained. This procedure was repeated several times with different similarity measures and cluster methods, and the result was selected that produced clusters with highest entropy of cluster sizes

[27] http://lucene.apache.org/java/
[28] http://www.google.com
[29] http://www.audioscrobbler.net/
[30] we used the Matlab function `linkage`

(i.e., the optimum is obtained when cluster sizes are of same size). The applied similarity measures were *cosine*, *Euclidean*, and *correlation*. Cluster methods were *complete link*, *average link*, *single link*, and *weighted average distance*.

5.2.3.2. PCA PCA was performed as usual, using the 16 eigenvectors with highest eigenvalues for projection of the data after mean removal. Each of the kept eigenvectors represents a basis in the low-dimensional term space. (Transposing the input matrix before PCA did not consistently increase or decrease classification accuracies.)

5.2.3.3. NMF To calculate NMF a number of algorithms have been proposed [Lee and Seung, 1999, Lee and Seung, 2000]. For the experiments, we use the implementation associated with [Hoyer, 2004] without sparseness constraints (i.e., the usual multiplicative update rules with square of the Euclidean distance as cost function). Based on the observation that when initializing H and W in a random manner, classification accuracies have a rather large variance, we replace the initialization method. Both W and H are initialised with the deterministic clustering method described above (Section 5.2.3.1), i.e., for initialization, H is initialised by clustering term vectors (containing the values of a given term over all artists), and W is initialised by clustering artist vectors (containing the values of a given artist over all terms). This initialization seems to improve the obtained results, although the final cost was not always lower than the cost obtained with random initialization. In most cases, the combination of correlation distance measure and complete linkage was the method that yielded highest entropy of clusters sizes both during initialization of H and W. In [Xu et al., 2003], each codebook vector is normalised to have a Euclidean length of 1.0. We adapt this normalization step. (In general, instead of normalising each codebook vector, normalizing each basis of the low-dimensional artist space to a Euclidean length of one did decrease accuracies in our experimental setup.)

5.2.4. Classification Results

In Table 10, results obtained based on the TD-EXA data are listed. For better visibility, the highest accuracy obtained for each method to compute the TFxIDF vector is printed in bold letters. The corresponding values obtained for AS-GO are listed in Table 11.

	Full TFxIDF	Linkage	PCA	NMF
Raw TF	**66.8%**	59.0% (56.2%)	55.7% (64.9%)	52.3% (62.8%)
like [Xu et al., 2003]	67.6%	65.5%	78.8% 82.5% (NCW)	77.5% **83.3% (NCW)**
"Robust"	82.8%	76.3% (77.4%)	89.4% **(89.7%)**	88.2% (89.6%)
LTC	87.2%	80.6% (82.1%)	89.5% (89.7%)	89.7% **(90.0%)**

Table 10: TD-EXA: 1NN classification accuracy for various methods to calculate TFxIDF vectors and various projection methods. Vectors are compared with cosine similarity after projection to 16 clusters (full TFxIDF: no projection). In parentheses: normalizing each (unprojected) TFxIDF vector to have a Euclidean length of 1.0.

5.2.4.1. Comparison of Methods to Calculate TFxIDF. When comparing Tables 10 and 11, it can be seen that using the raw TF generally yields the lowest accuracies, and using the LTC like variant yields the highest. When using the AS-GO data, after projection the variant like in [Xu et al., 2003] produces higher accuracies than the "robust" variant, while this is vice versa when using the TD-EXA data.

Additionally, the impact a change of the term dictionary and of the used search engine have seem to be different for the different methods. In all cases, accuracies are higher when calculated on the AS-GO data instead of the TD-EXA data. However, the raw TF method seems most affected (i.e., there is a larger difference of classification accuracy when the data is changed from AS-GO to TD-EXA), and the "robust" and the LTC like variant seem least affected. These results indicate that the LTC variant is a good choice for this kind of experiments.

5.2.4.2. Comparison of Clustering Methods. It can be seen that when reducing the length of the TFxIDF vectors to 16 by applying bottom-up clustering by linkage, the accuracies obtained typically are constantly below those obtained when using full TFxIDF vectors on the TD-EXA data. The difference is in the range from 2.1 to 10.6 percentage points, with an average decrease of 6.1 percentage points. However, the corresponding results for AS-GO show that this is changed when using a different search engine and term dictionary. In the latter case, accuracies are increased in two cases, with the increase be-

	Full TFxIDF	Linkage	PCA	NMF
Raw TF	87.8%	85.5% (84.7%)	86.1% **(88.2%)**	85.6% (86.7%)
like [Xu et al., 2003]	86.0%	87.9%	91.2% 91.3% (NCW)	90.6% **91.5% (NCW)**
"Robust"	87.0%	87.8% (86.4%)	90.2% **(90.9%)**	90.1% (90.5%)
LTC	91.7%	90.8% (89.4%)	92.6% (93.1%)	**93.5%** (92.9%)

Table 11: AS-GO: 1NN classification accuracy obtained when vectors are compared with cosine similarity after projection (full TFxIDF: no projection). 16 cluster. In parentheses: normalizing each TFxIDF vector to have a Euclidean length of 1.0 before calculating the projection. Normalization does not have an influence on the full TFxIDF vectors as cosine similarity is used.

ing 0.8 and 1.9 percentage points, respectively. Also, the maximum decrease is lower, i.e., 3.1 percentage points.

For PCA and NMF, results are more conclusive. Except for the "Raw TF" method, accuracies are increased when similarities are calculated on the 16 dimensional vectors instead of the high-dimensional TFxIDF vectors. We see this as an indication that the projection can provide a "meaningful" abstraction from the original data by combining related terms. One might tend to explain the low performance after projection of raw TF vectors by an inappropriate scaling of raw term frequencies, i.e., this scaling might be not suited for methods that model data points as linear combinations of a set of bases. Without examining this in detail, this notion is *not* supported by a simple experiment. If instead of the raw term frequency, the logarithm is used, accuracy obtained by the full vectors is 88.9%, while it decreases to 86.6% after PCA projection when using the TD-EXA data and artist vectors are normalised to 1.0.

5.2.5. Assessing the Found Concepts

In the experiments presented above, 56 different combinations are evaluated. As it is infeasible to show all concepts found by these different methods, here only some of them that are likely to be among the most interesting ones are shown. To evaluate which methods produce the most meaningful concepts, a user study could be of interest. While a user study is not conducted in the course of this thesis as it is time and cost intensive, we argue that those

projections that produce highest classification accuracies are those that are of most interest. Thus, in the following, the concepts found by the various projection methods for the LTC-like variant on the AS-GO data are shown and discussed.

0.90	muddy waters	0.91	columbia	0.90	death metal
0.87	delta blues	0.90	1950s	0.89	thrash
0.87	electric blues	0.90	new orleans	0.88	metal bands
0.86	slide guitar	0.89	1960s	0.88	power metal
0.85	blues guitar	0.88	ballads	0.87	pantera
0.86	hardcore	0.93	ny	0.89	traditional country
0.82	punk rock	0.92	french	0.88	bluegrass
0.81	weird	0.92	european	0.87	alan jackson
0.80	emo	0.91	research	0.87	garth brooks
0.80	progressive	0.91	nights	0.87	americana
0.82	my stuff	0.93	oldies	0.86	techno
0.82	favourite	0.90	gospel	0.85	remix
0.80	playlist	0.89	60s	0.84	electro
0.79	australian	0.89	christmas	0.84	beats
0.73	britain	0.88	favorites	0.81	electronica
0.83	classic rock	0.95	the jazz	0.84	james brown
0.82	sucks	0.95	saxophone	0.83	motown
0.80	420	0.94	pianist	0.79	funk
0.80	berlin	0.94	trumpet	0.77	soulful
0.79	get more	0.94	bop	0.77	funky
0.80	black death metal	0.94	melodic	0.85	disco
0.77	death thrash	0.93	experimental	0.83	ct
0.76	melodic death metal	0.93	complex	0.83	worldwide
0.76	progressive metal	0.93	noise	0.83	bookmark
0.75	dark metal	0.92	intense	0.80	bt
0.82	dub				
0.77	ska				
0.74	dancehall				
0.74	jamaica				
0.70	conscious				

Table 12: Clusters found by linkage algorithm (correlation / complete link). Top five terms with highest cosine similarity to cluster centers shown. Cluster centers are means of all term vectors that make up a cluster.

5.2.5.1. Low-dimensional Representation found by Linkage. Table 12 lists the concepts that are found by the linkage clustering algorithm. The actual algorithm was to use correlation as a distance measure and combining clusters by complete link, as this produced the highest entropy of cluster sizes. Some of the terms closest to the cluster centers represent a music-related se-

mantic concept. For example, the terms at (1, 1) are related to blues music, and those at (1, 3) and (5, 1) are related to metal music. Other term lists however seem not to be linked to consistent musical concepts. Cluster (2, 2) seems to be linked to geographic terms (which is a phenomenon that was already observed in [Pohle et al., 2007a]), while clusters (4, 1) and (5, 3) might represent no clear concepts. Thus, there may be room for improving upon this method.

5.2.5.2. Low-dimensional Representation found by PCA. The low-dimensional bases found by PCA are illustrated in Table 13. As each basis also contains negative values, the three terms with highest positive and the three terms with highest negative weight are shown for each basis. Looking at the terms in the table, it becomes apparent that while some term combinations are clearly related to musical concepts, in general this way of representing concepts does not support understandability. In particular, when assessing the effect of combining bases, one has to consider that positive and negative weights may cancel out to a certain extent when they represent similar concepts.

5.2.5.3. Low-dimensional Representation found by NMF. In Table 14, the factors found by NMF based on the LTC variant are listed. Most term groups are closely linked to a musical concept. In many cases, the terms are genre names, or names of bands that are associated with a certain style of music. In two cases, this seems not immediately obvious. The terms ranked two to five for basis (3, 3) are bands that are commercially successful. A hypothetical explanation for the term "class a1" is that such artists might be considered first-class artists. The terms for basis (4, 1) are genres at least weakly linked to punk rock and punk metal[31]. Overall, the bases found by NMF seem to be more clearly linked to musical concepts than those found by linkage or PCA.

5.2.5.4. Comparison to Other NMF Variants. The variants discussed above are all based on AS-GO data and the LTC-like variant. To get some insight into what concepts are found by NMF on different TFxIDF variants, four additional term tables are listed in Appendix A.2, Tables 16 to 19. These are the NCW variant calculated like [Xu et al., 2003] on AS-GO, NMF on

[31] cf. http://en.wikipedia.org/wiki/Punk_metal and http://en.wikipedia.org/wiki/Punk_rock, both pages retrieved 31.07.2009

0.50	alan jackson	0.33	black metal	1.00	real country
0.48	real country	0.32	power metal	0.92	alan jackson
0.48	power metal	0.28	death metal	0.80	honkytonk
−0.85	keith jarrett	−0.83	garth brooks	−0.84	northern soul
−0.88	blue note records	−0.98	real country	−0.84	rasta
−1.00	hard bop	−1.00	alan jackson	−0.91	roots reggae
1.00	delta blues	1.00	roots reggae	1.00	roots reggae
0.98	electric blues	0.93	rasta	0.88	rasta
0.92	jump blues	0.84	rastafari	0.80	rastafari
−0.46	garth brooks	−0.67	northern soul	−0.30	paul oakenfold
−0.57	alan jackson	−0.80	quiet storm	−0.30	detroit techno
−0.67	real country	−0.82	slow jams	−0.30	deep house
1.00	northern soul	1.00	east coast rap	0.93	progressive rock
0.86	prog	0.91	jamband	0.88	whitesnake
0.82	slow jams	0.85	contemporary classical	0.83	dokken
−0.59	30s	−0.88	grp records	−0.68	real country
−0.65	hardcore rap	−0.95	deep house	−0.71	hardcore rap
−0.94	east coast rap	−0.97	progressive house	−1.00	east coast rap
0.52	blue note records	1.00	peter white	1.00	dokken
0.49	jump blues	0.94	jeff lorber	0.89	ratt
0.44	real country	0.91	jazz-fusion	0.82	whitesnake
−0.36	medieval	−0.56	girl groups	−0.66	auf der maur
−0.90	classic folk	−0.57	oldies but goodies	−0.76	70s music
−1.00	folk revival	−0.65	beach music	−0.91	transplants
1.00	grp records	1.00	folk revival	0.97	essential jazz
0.86	east coast rap	0.80	classic folk	0.63	progressive rock
0.82	classic folk	0.25	american roots	0.58	hard-bop
−0.77	ashanti	−0.47	prog	−0.56	dave grusin
−0.84	transplants	−0.49	jump blues	−0.69	peter white
−0.90	class a1	−0.50	progressive rock	−1.00	xd
1.00	xd				
0.49	beach music				
0.20	pat metheny group				
−0.32	russ freeman				
−0.36	grp records				
−0.37	jazz instrumental				

Table 13: Components found by PCA based on LTC variant. Artist vectors are normalized before PCA. Term dictionary from Audioscrobbler. For display purposes, each factor is normalized to have a maximum absolute value of 1.0. Only the three terms with highest and the three terms with lowest weight are shown. Components are sorted by descending importance from left to right and from top to bottom.

0.40	roots reggae	0.37	progressive rock	0.30	east coast rap
0.36	rasta	0.33	prog	0.21	hardcore rap
0.32	rastafari	0.23	progressive metal	0.20	west coast rap
0.25	rocksteady	0.23	avant-prog	0.17	old school rap
0.24	ragga	0.22	space rock	0.17	gangsta rap
0.24	dokken	0.18	black metal	0.27	jeff lorber
0.20	ratt	0.16	power metal	0.25	peter white
0.19	tesla	0.16	century media	0.25	larry carlton
0.19	whitesnake	0.15	morbid angel	0.23	david sanborn
0.18	hair metal	0.15	doom metal	0.23	spyro gyra
0.49	folk revival	0.26	hard bop	0.21	class a1
0.34	classic folk	0.26	blue note records	0.21	transplants
0.19	american roots	0.23	hard-bop	0.19	ashanti
0.14	folk rock	0.20	post-bop	0.16	boyz ii men
0.13	folk blues	0.19	hardbop	0.16	snoop dogg
0.14	pop punk	0.28	electric blues	0.33	real country
0.14	metalcore	0.27	delta blues	0.32	alan jackson
0.14	math rock	0.25	blues guitar	0.27	garth brooks
0.13	emo	0.25	british blues	0.26	classic country
0.12	death metal	0.24	classic blues	0.25	honkytonk
0.19	deep house	0.27	northern soul	0.18	vocal jazz
0.18	progressive house	0.22	slow jams	0.18	30s
0.18	detroit techno	0.21	southern soul	0.18	grp records
0.17	acid house	0.21	beach music	0.18	classic jazz
0.17	breakbeat	0.20	classic soul	0.16	jump blues
0.20	sun ra				
0.19	contemporary classical				
0.18	ecm				
0.17	free jazz				
0.17	jamband				

Table 14: Factors found by NMF based on LTC variant (with normalization of artist vectors). Term dictionary from audioscrobbler, search engine is Google.

"robust" variant on AS-GO, NMF on LTC-like variant on TD-EXA, and NMF on "robust" variant on TD-EXA.

Table 16 shows the bases calculated by our adaptation of the NCW variant (with which the highest classification accuracy based on our implementation of the variants from [Xu et al., 2003] is obtained in the results reported in Tables 10 and 11). While in general the terms form meaningful groups, it becomes obvious that two bases contain only one term: "ihs" and "tag 1". This may be an artifact resulting from our way of initializing the NMF algorithm.

Replacing AS-GO with TD-EXA, the terms associated with the bases found by NMF on the LTC-like variant become less easily interpretable, as can be seen in Table 18. This shows that the robustness against changing the term dictionary and/or search engine is limited. The assessment of the influence each of the two factors has on the quality is left for future work.

A comparison of the results from calculating the NMF on the LTC-like variant and the "robust" variant (Table 17) shows that when using the "robust" variant, also easily understandable concepts are found. Considering that the obtained accuracies are only slightly below those obtained with the LTC-like variant, the "robust" variant may be also a good choice. However, another consideration should be kept in mind: As the "robust" variant is based on the relative number of web pages a term occurs on, the algorithm used by the search engine might have an even larger influence on the results than when using the LTC-like variant, which is based on the number of occurrences of a given term on all web pages retrieved for an artist.

5.3. Comparison to Other Work

In [Pohle et al., 2007a], we applied NMF on the *AS-GO* data and the LTC like variant. In [Pohle et al., 2007a], genre classification accuracies after projection are lower than for full TFxIDF vectors. We see a likely reason for this in the different initialization step in the NMF algorithm (random instead of clustering as done here). Based on the found low-dimensional artist representations, we presented an application for artist recommendation (cf. Chapter 7, [Pohle et al., 2007b]).

[Levy and Sandler, 2008] apply Latent Semantic Analysis (LSA) and Probabilistic Latent Semantic Analysis (PLSA) to TFxIDF vectors constructed from last.fm tags and the last.fm tag frequencies on the *track* level. A detailed comparison of the differences arising from track level vs. artist level, and setting

term frequencies by last.fm frequencies vs. web text counts is yet to be done. While [Gaussier and Goutte, 2005] point out that NMF is PLSA with KL Divergence, the effect of the change of distance function in the update rule is another unexamined parameter.

5.4. Conclusions

Not considered in the experiments above is the parameter "text processing" (e.g., stemming), and the experiments are only based on one set of artists. Thus, there are some parameters that have not been examined, and that may have a certain influence. Keeping this in mind, a number of observations can be made from our experiments.

- In a number of cases, classification accuracy improved when calculating similarity based on vectors projected by PCA and NMF, instead of using full-length TFxIDF vectors. We see this as an indication that the found clusters support a meaningful abstraction from the original data.
- Normalizing each TFxIDF vector to an Euclidean length of 1.0 before dimensionality reduction constantly led to improved classification accuracy for PCA, and in most cases also of NMF.
- The choice of term dictionary and way to estimate term weights has a clearer impact on some of the TFxIDF approaches (raw TF, like [Xu et al., 2003]), while the difference in classification accuracy is lower for other TFxIDF approaches ("robust", "LTC").

For the better-performing methods, accuracies are quite high, and the found concepts seem reasonable. Although this renders further improvements questionable, for completeness it would be possible to conduct experiment series as done in Chapter 2 to optimize different parameter settings yet unexplored.

6. Combining Audio and Web Based Data

In the previous chapters, methods to obtain relevant information from the audio signal and from textual artist descriptions have been discussed. Obviously, it is of interest to asses whether these two sources of information can be combined to obtain a more robust overall recommendation technique.

Following the previously presented reasoning, in this chapter we assess if an improved *artist* similarity measure can be created by applying simple methods to combine a text based and an audio based artist similarity measure. In these experiments, we take into consideration a possible difficulty: we assume that for each considered artist, there are music tracks available, but that for some of them, text-based information may be not available. This reflects a user collection that contains both music from well-known artists, and music by unknown artists. Obviously, for the latter it may be difficult to obtain a sufficient amount of textual information from the Web.

6.1. Introduction

As discussed in the previous chapters, artist similarity measures are an important part of many MIR systems. In this chapter, a method to combine two commonly used sources to automatically compute artist similarity is discussed and evaluated. The first source are texts (e.g., web pages, tags, and song lyrics), as described in Chapters 3 and 5. One problem that may occur with such data is that the data is not available for all artists (or tracks, respectively). There usually is not much information about very unknown artists available on the Internet: the artist is only mentioned on few (if any) web pages, and there are likely not sufficiently many human-assigned tags available.

The second source to compute artist similarity is the audio signal of music by the artists. To do so, the techniques used to compute song-to-song similarity (see Chapters 2 and 4) are extended to the artist level. For example, in [Berenzweig et al., 2003], an artist is represented by building a common model over all pieces. The similarity of two artists is calculated by comparing their models.

Here, we investigate if these two sources for artist similarity measures – audio and text data – can be combined to create an improved measure. [Li and Ogihara, 2004] present an approach to train artist classifiers with such heterogeneous data from lyrics on the one hand and sound on the other hand.

In contrast, in this chapter, we present a way to build a combined similarity measure (not a classifier), and evaluate it with partly missing data.

The remainder of this chapter is organized as follows. Section 6.2 presents the basic idea. The data and similarity measures investigated are described in Section 6.3. Results for the combination of both measures with full data coverage are shown in Section 6.4, while the method is extended to partly missing data in Section 6.5. Finally, Section 6.6 concludes the chapter.

6.2. Basic Idea

The basic idea for combining the two different similarity measures is to treat each of them as a black box, and only combine their output. Assuming that both measures reflect a different aspect of similarity, and that both of these aspects are basically of same importance to an overall measure, the main task is to create a common "vote" for the best-matching artists. Several methods are evaluated: the simple ones are taking the minimum, mean, and maximum of both measures.

Based on the idea that the two measures produce differently scaled outputs, but the ranking of their output seems of more importance, the output of both similarity measures is then replaced by the respective ranking (details will be explained in Section 6.3.3). Using these rankings as a basis makes the combination of the measures more intuitively meaningful.

In particular, a main motivation for this work is the following idea. By taking the maximum of the two rank-based distance measures[32], it becomes possible to filter out "bad" matches from one measure M_1 even if the other measure M_2 does not cover all artists, but rather is only calculated on a subset of the artists covered by M_1. This could be done, for instance, by taking the maximum distance calculated by the two measures, where both measures are available. Distances involving artists only covered by M_1 could be left unchanged.

Although the idea is based on taking the maximum rank of the two rank based measures, the minimum and mean functions are also evaluated for completeness. The detailed setup is presented in the next section.

[32] The terms *distance measure* and *similarity measure* are used interchangeably here. The term *similarity measure* reflects that only the top-ranked artists are of importance, while for the computation it is of importance that *distances* are used in the rank-based method.

6.3. Experimental Setup

We used a collection of music by 3818 artists assigned to 45 genres by expert opinion. The largest genres *pop*, *r'n'b* and *pop/rock* contained 182, 178 and 172 artists, respectively, while the smallest genre *gangsta rap* contained 4 artists. Songs by all artists are given as 30 second excerpts. There are 37,110 excerpts in the collection, with a median value of 3 excerpts per artist.

In our experimental setup, we use the genre labels as ground truth. Assuming that for a better similarity measure, more of those artists that are ranked closely are in the same genre as the seed artist than for an inferior similarity measure, we use k-NN genre classification as a quality indication. On our data, the baseline accuracy for this method is 4.77% for an algorithm that classifies all items as belonging to the largest class.

6.3.1. Audio Similarity

For determining the audio similarity of music, we apply the widely used MFCCs. All MFCCs of a piece (or all MFCCs of all pieces by a particular artist) are combined into a single Gaussian model. Two models can be compared by the KL distance. The resulting values are scaled, thus the resulting distances are in the unit interval [Mandel and Ellis, 2005a].

We tried different methods to compute the distance between two artists. On the one hand, we tried methods based on the pairwise distance between tracks by two artists, and on the other hand, we evaluated building a combined MFCC model for each artist. For combining the pairwise track distances of tracks by two artists into one common measure, we tried min, mean, median and max functions on all pairwise distances (cf. [Logan, 2004]), and a number of slightly more complicated asymmetric methods. We found the method to build one common GMM for an artist to work best. Probably this is due to the short duration of 30 seconds for each individual clip. Thus, in the experiments presented here, we use this method. It turns out that about 61% of all pairwise audio based artist distances have a value of 0.99 to 1.0. Classification accuracies obtained with this measure can be found in Table 15 (denoted *Audio*).

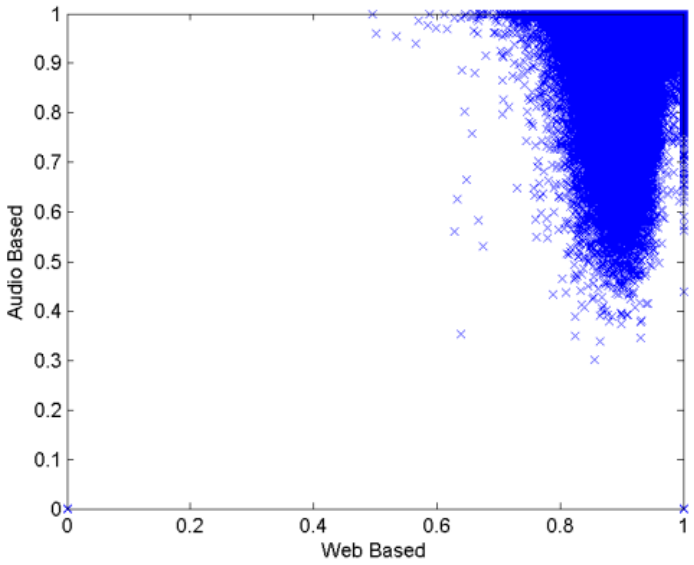

Figure 34: Values of web-based artist distance vs. audio-based artist distance for 1 million randomly chosen artist pairs.

6.3.2. Web-Based Similarity

To compute the text based artist similarity, the search engine Exalead[33] was queried for web pages containing the artist names. Returned documents were retrieved from the web and concatenated into one text per artist. After stemming, these texts were scanned for a dictionary of 7570 music-related stemmed tags. Tags were retrieved from the Ausdioscrobbler[34] web service. The occurrences were counted and used to construct a *TFxIDF* vector for each artist. The similarity between two artists is computed by calculating the cosine similarity between their vectors. Classification accuracies obtained with this measure are also given in Table 15 (denoted *Web*).

6.3.3. Combined Similarity Measure

In Figure 34, audio and web based distances are plotted against each other for 1 million randomly chosen artist pairs. It can be seen that the two measures are not highly interdependent (i.e., knowing the value of one measure does

[33] www.exalead.com
[34] www.audioscrobbler.net

not give very much information about the value of the other). Also, for both measures, there are only few values that are small (i.e., artists that have a high similarity value).

To combine both measures, in a first experiment simply their output values are combined by taking their minimum, mean and maximum values, respectively. However, as indicated in Figure 34, the distributions the output values take differ, thus such a direct combination might be not the best combination method. As in the scenario discussed here, it is rather of importance which artists are ranked highest by a measure, we replace each of the calculated distances with the rank. For this operation, for a seed artist, all distances to all other artists are calculated and sorted. Then the distance to each artist is replaced by the resulting rank. This is done for both measures independently. The n-th closest artist is assigned number n. This way, the output of a measure is transformed into a more directly interpretable value. Most importantly, the outputs of both measures can be combined based on this new measure. Once again, they are combined into one common measure by taking their minimum, mean and maximum value. We did not try a parametric method due to potential overfitting to the used music collection. Results are given in the next section.

	1-NN	3-NN	5-NN	10-NN	20-NN
Audio	15.8%	14.6%	13.6%	12.4%	11.2%
Web	17.1%	15.5%	14.6%	13.5%	12.1%
Min	19.0%	16.9%	16.0%	14.6%	13.0%
Mean	19.0%	17.1%	15.8%	14.8%	**13.4**%
Max	15.7%	13.5%	12.5%	11.2%	9.9%
Min_r	16.2%	15.6%	15.1%	14.0%	12.8%
Mean_r	**21.5**%	**18.4**%	**17.0**%	**15.0**%	13.0%
Max_r	20.4%	17.7%	16.2%	14.4%	12.3%

Table 15: Leave-one-out classification accuracies for the various approaches when both audio and web data is available for all artists.

6.4. Results

Table 15 shows the accuracies obtained for the basic audio and web based distance measures, as well as the various evaluated combinations of both. The obtained accuracies are low compared to those found in the literature, which may be due to the high number of 45 genres (resulting in a baseline of 4.77%;

a purely random distance measure produces about 3% accuracy). As measure of accuracy, $\overline{\text{acc}_\text{A}}$ (Equation 3, page 28, i.e., the fraction of nearest neighbors considered that was in the same class as the test item) was taken as a measure for accuracy, because we consider this as a more exact measure than a majority vote.

It can be seen that taking the simple minimum and mean of the two basic measures yields an improved measure when considering the closest neighbors. This may be corresponding to the assumption "if one measure outputs a very small distance, then the artist is indeed a good match". In line with this, using the maximum decreases performance. For the rank-based measures (denoted as Min_r, Mean_r and Max_r, respectively), it can be seen that Mean_r yields the best overall results. From Table 15 it can be seen that this common measure produces better results for up to 10 NN than each of the underlying measures alone, and also better results than the reported non-rank-based combination variants.

6.5. Missing Data

If a music collection contains music by very unknown artists, for the unknown artists there likely is only the music available but not community-based meta-data or lyrics. To see if the method also can be used to handle such missing data, we conducted experiments with some of the artist web data intentionally left out. While audio-based distances are available for all artists, for left out artists no web-based distance data is available. Consequently, when computing the rank-based distances between artists, the maximum rank between artists is lower for the web-based measure than for the audio based measure.

Measures were combined by taking again the minimum, mean and maximum of both distances where both distances are available, and simply taking the audio based distance where only the audio based distance is available. During the experiments, it was assumed that a different fraction of the web-based artist data was available, ranging from 0% (corresponding to the purely audio-based measure) and 100% (corresponding to the results presented in Section 6.4). Missing artists were chosen randomly.

The results presented in Figure 35 indicate that such a combination of partial data results in a decreased accuracy for the maximum and the rankbased min methods when only 10% of the artists have web data associated. However, somewhat surprisingly, when using the other methods for such a combination,

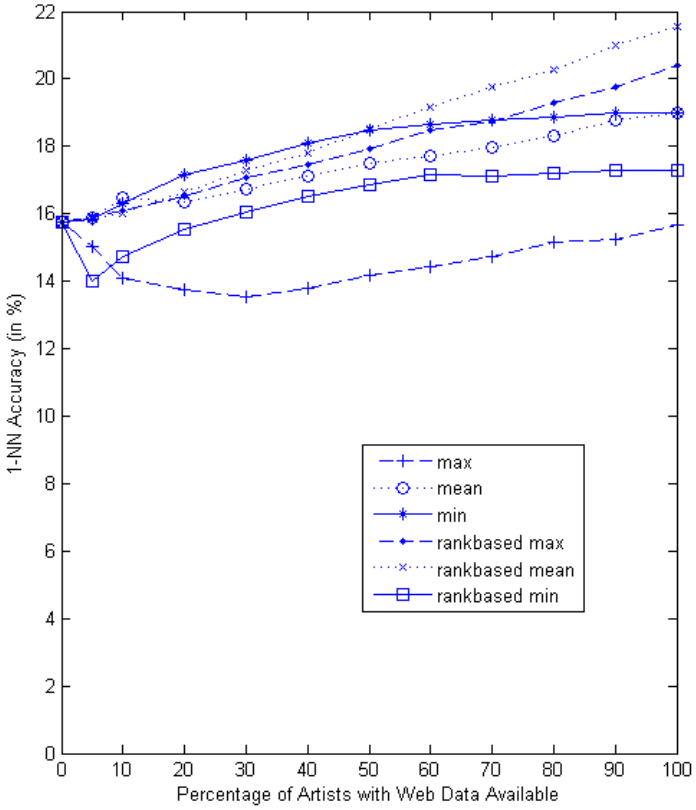

Figure 35: Partly missing artist data. Artists with missing data are chosen randomly. Results averaged over 10 runs.

the average performance of the audio-based measure is increased with all parameter settings examined here. However, considering each of the ten runs separately, it turns out that when less than 30 per cent of the artists had web data associated, performance decreased slightly in some cases for all methods. From the results, it follows that using the rank-based mean combination seems the best method. It is particularly interesting that the fraction of available web based artist distances seems to be almost linearly associated with an increased overall performance.

6.6. Conclusion

In this chapter, we presented a method to combine a web based and an audio based artist similarity measure into an improved common similarity measure. The experimental results indicate that the method can also be applied to obtain an improved measure when web data is missing for some of the artists.

These experiments point the direction to future extensions. For example, there is an obvious method to combine this method with the "topic" based search discussed in Chapter 5. To do so, the artist representations can be replaced by the low-dimensional representation, and the artist ranking produced by the text-based part is replaced by the low-dimensional similarity to the (possibly user-modified) low-dimensional query vector. Such an integration can even be made in a more seamless manner by allowing the user to adjust the weight of the topic based artist measure and the audio based artist similarity measure by determining the relative weights of the two measures, which then are combined by a weighted sum.

7. Applications

In this chapter, a number of example applications is outlined that can make use of the techniques described and evaluated in this thesis. The first scenario (Section 7.1) is the enrichment of existing audio players with functionality based on audio features. The second scenario (Section 7.2) comprises a distinct concept for audio players, where music is accessed by turning a wheel. A third application scenario (Section 7.3) is the use of higher-level artists concepts to offer controllable artist recommendations. The chapter concludes by drafting further possible application scenarios.

7.1. Using Linear Descriptors by Functionality Injection

Considering the results presented in Chapters 2 and 4, it becomes apparent that the most intuitively understandable audio descriptors developed in the course of this work are *Harmonicness* and *Attackness*. They aim to describe the amount of harmonic and percussive structures in a cent-scaled spectrogram that is level-adapted to better match human loudness perception. While horizontal and vertical structure in principle are not necessarily related to each other (e.g., a high Harmonicness value does not imply a low Attackness value), combining them into the one common value H2A Ratio (Equation 36) yields an intuitively understandable single scalar descriptor. In general, a low H2A Ratio indicates dominance of drum-like elements, while a high H2A Ratio usually is assigned to pieces with mainly sustained sounds and few percussive elements.

A simple yet effective way to use such linear audio descriptors in conjunction with existing audio players (software or hardware based) is to write the descriptor value as a string at the beginning of the file name, or into a metadata field of the track. Depending on the actual player used, such data can then be processed by the player's built-in functionality. We refer to this method as *Functionality Injection*.

For example, the H2A ratio can be written into the *Comment* field of each music file. To facilitate string comparison and sorting, the descriptor value can be represented by three digits in the range from 000 to 999. When loading this collection into a software music player that supports the sorting of the currently selected tracks according to the string written into the comment field, it is possible to sort the current tracks according to H2A Ratio without

having modified the player application, by simply sorting them according to the text written into the comment field.

In principle, this basic concept can be also used on mobile players. The actual usage depends on the capabilities of the player. For example, if a player does not support reading of mp3 tags, but allows for sorting according to the file name, file names can be renamed to start with the respective feature name and feature value.

7.2. "The Wheel Player"

While Functionality Injection allows to use linear audio descriptors in conjunction with existing audio players, the methods described in this thesis can also be applied to create novel music player concepts. Such a concept is the wheel player which we first introduced in [Pohle et al., 2005b].

7.2.1. Basic Idea

The basic idea behind the wheel player is to arrange all the music into one circular playlist containing regions of similar music, and to offer the user a way to quickly jump into any position of the playlist. Once the user knows in what region to expect which music, it becomes easier to select a certain type of music. Furthermore, once a piece is selected, the player goes on by playing similar music that is next in the playlist. Such an application can be useful when there should only be little attention paid to the audio player, e.g., during workout.

To quickly jump to a region of the playlist, an interface consisting of a wheel can be used, as drafted in Figure 36. Grouping of similar music can be achieved by a number of different approaches. The most simple one is to use only audio similarity [Pohle et al., 2005b]. More advanced methods also take into consideration web based information [Schedl et al., 2006, Knees et al., 2006a, Pohle et al., 2007, Schnitzer et al., 2007], as discussed next.

7.2.2. Audio Based Algorithms

As presented in [Pohle et al., 2005b], the wheel player's playlist can be created by calculating the audio similarity between each pair of tracks in the collection, converting these into distances, and then calculating a Traveling Salesman

Figure 36: Idea for an audio player. All available music is arranged around the wheel. The aim is to arrange the music in such a way that similar music forms clustered regions around the wheel, so that a music style can easily be selected (cf. [Pohle et al., 2005b,Pohle et al., 2007]).

Problem (TSP) algorithm on these distances. A number of TSP algorithms have been evaluated in [Pohle et al., 2005b] by considering genre labels of songs as an evaluation criterion. Regions of similar songs are assumed to contain only a small number of different genres. In this evaluation it turned out that a SOM-based algorithm and a Minimum Spanning Tree algorithm seem to be better suited for the task than a simple greedy method and an elaborated algorithm. We assume that improvements of the underlying audio similarity (cf. Chapter 2) measure will result in improved playlists.

7.2.3. Improvement with Web Based Data

When constructing the playlist based only on audio data, it can be observed that in some cases various regions that are apart contain similar music. Two methods to use web based data have been proposed to reduce this.

7.2.3.1. Distance penalties. In [Pohle et al., 2007] it is proposed to use web-based artist data for modifying the pairwise track distances so that the TSP algorithm is forced to preferably use transitions between tracks of similar artists. To this end, similar artists (as determined by a similarity measure based on data obtained from the web, also cf. Chapter 3) are clustered by using a Self-Organizing Map (SOM). Distances between tracks of artists that are not in the same cluster are increased by a penalty term before the TSP algorithm is computed.

7.2.3.2. Labeled clusters. To offer faster insight into which music is contained in which region, techniques such as the one discussed in Chapter 5 can be used to build and label the playlist. To this end, as described in [Schnitzer et al., 2007], a number r of clusters is created by running a NMF algorithm with r factors on those artists in the collection that have term vectors associated. An artist cluster is created from a factor by assigning all artists to it that have the largest association to this factor. The cluster is described by the n top weighted terms associated with the factor. Tracks by artists that have no TFxIDF vectors associated are assigned to the clusters by searching for the track in the clusters that has the minimum audio distance to the given unlabeled song. An overall playlist is created from this cluster data in a two-step procedure. First, all tracks within a cluster are arranged in a circular manner by calculating a TSP on them. Second, these playlists are combined by breaking each playlist up at the longest distance, and combining the (now linear) playlists in a greedy manner into one long circular playlist. This long playlist can be displayed in an iconic way (e.g., as a bar) along with the cluster labels at appropriate positions. Such a demonstration application is presented in [Schnitzer et al., 2007].

While the results in [Schnitzer et al., 2007] show that NMF can be used for this algorithm, it also turns out that a more simple binning method performs comparably to the more advanced NMF approach. A reason for that the simpler algorithm performs comparably may be that the description of the cluster centers is of interest, and not the degree to which instances are associated with individual clusters. An application that makes use of such information is presented next.

7.3. Artist Browser

In Chapter 5 it is discussed how NMF can be used to compress long TFxIDF vectors to few (e.g., $r = 16$), ideally meaningful concepts. This allows to represent each artist by the degree to which the artist is associated with each of these concepts. The low-dimensionality artist representations then can be used to define (or refine) queries for artists that have certain characteristics. Two example applications are introduced in [Pohle et al., 2007a] and [Pohle et al., 2007b]. Both work by the same basic principle. The user first selects a seed artist out of a drop-down list. This triggers a lookup of the selected artist's representation in the compressed domain. The corresponding values (i.e., degree to which this artist is associated with each of the r concepts) are transferred to r sliders that are displayed in the user interface, and labeled with descriptions of the corresponding concepts. These labels can either simply be the n (e.g., $n = 3$) top terms in each cluster [Pohle et al., 2007a], or assigned manually during program design [Pohle et al., 2007b].

Figure 37 shows this state in the application from [Pohle et al., 2007a] for the artist Miles Davis. It can be seen that this artist is most associated with the concept that has the most highly weighted terms *hard bop, blue note, modern jazz*, but Miles Davis is here also associated with the concepts *funky soul (...)* and *jazz vocals (...)*.

Internally, the slider positions are linked with a query vector of length r that is used to determine the most similar artists in the collection by determining the cosine similarity of each compressed artist vector and the query vector. The most similar artists are shown in the list at the right hand along with the cosine similarity value with respect to the query vector.

When the slider positions are changed, a new query with the modified query vector is triggered, and the artist list is updated with the artists most similar to the modified query vector. An example of updating the concepts is given for the application presented in [Pohle et al., 2007b], which is a similar application but also displays the last.fm page of the best-matching artist. Figure 38 shows the command section of this artist browser.

In the example, first with the "Quick Selection" drop-down list the seed artist *Beatles* was selected. The system then transferred the internal representation of the Beatles (i.e., the concept vector of length 16) to the sliders. All sliders are left untouched in this example, except for the slider labeled "Punk". From the picture, it can be seen that the system associates the Beatles mainly with "Classic Oldies", but also – to a much lesser extent – they are associated with

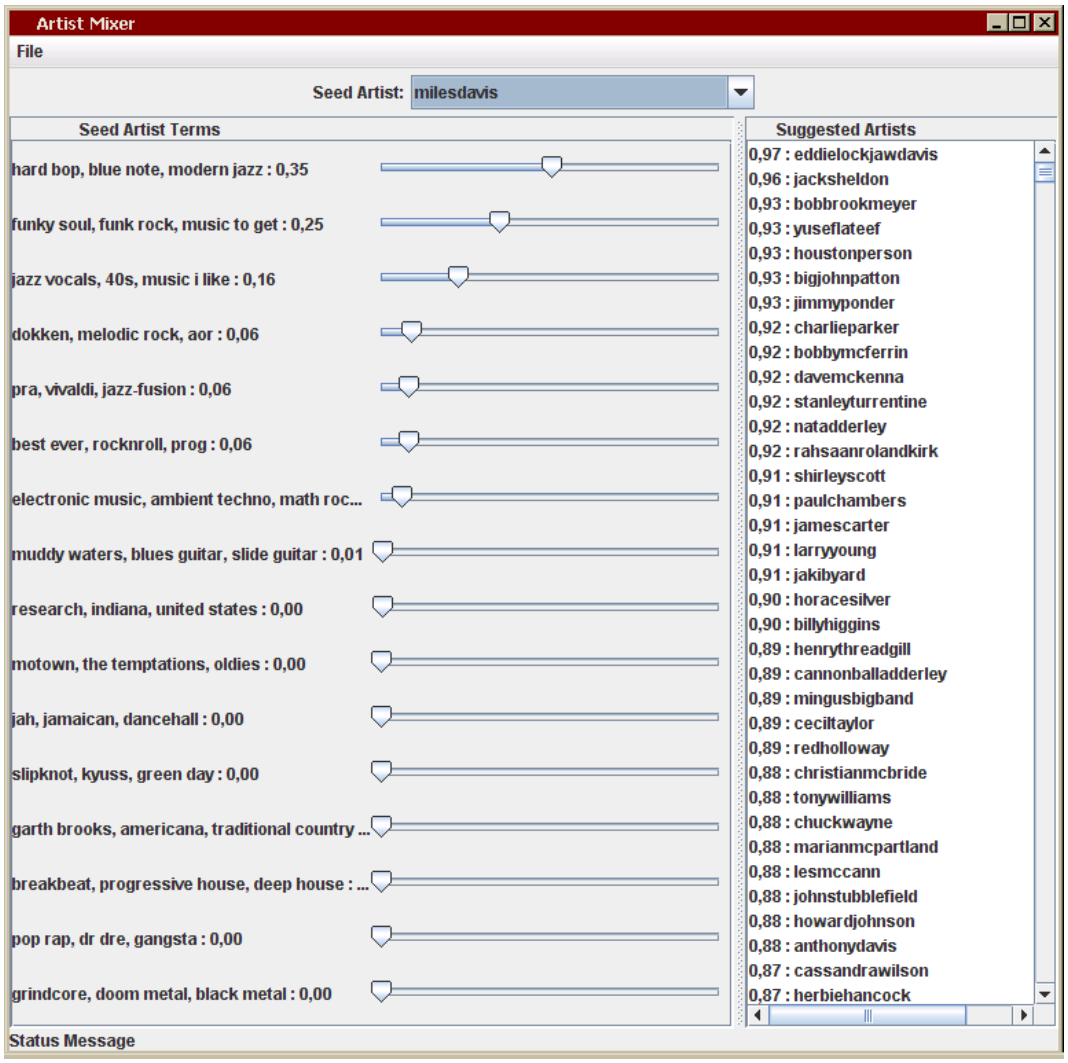

Figure 37: Artist Browser [Pohle et al., 2007a]

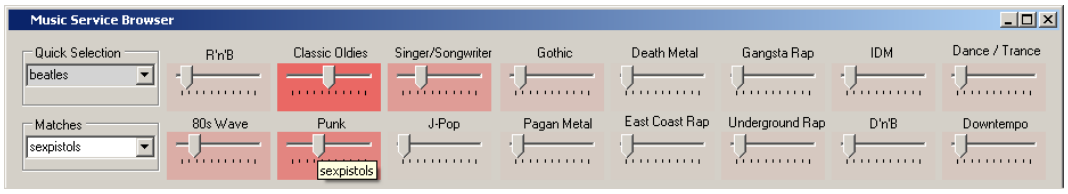

Figure 38: Music Service Browser [Pohle et al., 2007b]

the "Singer / Songwriter" and "80s Wave" categories. When the user increases the value of the "Punk" slider slightly, "Iggy Pop" is suggested. Increasing it more (as shown in the picture), the Sex Pistols are the artist whose internal description best matches the modified query vector.

7.4. Further Potential Applications

The presented example applications motivate that the methods developed in the course of this thesis might be useful for improving existing music retrieval systems to allow for a more convenient and more intuitive interaction. To conclude this chapter, two further potential application scenarios are drafted here, without presenting actual implementations of the concepts. These are a constrained random play function called *DendrogRandom* and a method to automatically update the music on a mobile player according to the user's taste.

7.4.1. DendrogRandom: Constrained Random Play Function

The method discussed in Chapter 5 is designed to find topics associated with the music. In the Artist Browser application discussed in Section 7.3 these topics are collocated in a nonhierarchical manner. As each topic is represented by a vector, it is possible to define a similarity between topic vectors by e.g. calculating their cosine similarity. Based on the distances between topics, a dendrogram can be calculated with bottom-up clustering. Figures 39 and 40 show the dendrograms based on the data used in the artist browser applications.

The nodes of the resulting dendrogram are automatically annotated in a bottom-up manner. Leaf nodes are assigned the vectors of the respective factors. Each node is annotated with the three terms with highest weight associated, with terms scaled according to their weights. The vectors associated with inner nodes are created by normalizing the vectors associated with the two next lower nodes, and summing them up. It can be seen that in many cases the overall hierarchy reflects intuitive aspects. We give two examples how such a dendrogram can be used in an audio player, *dendrogram zooming* and a constrained random play function we call *DendrogRandom*.

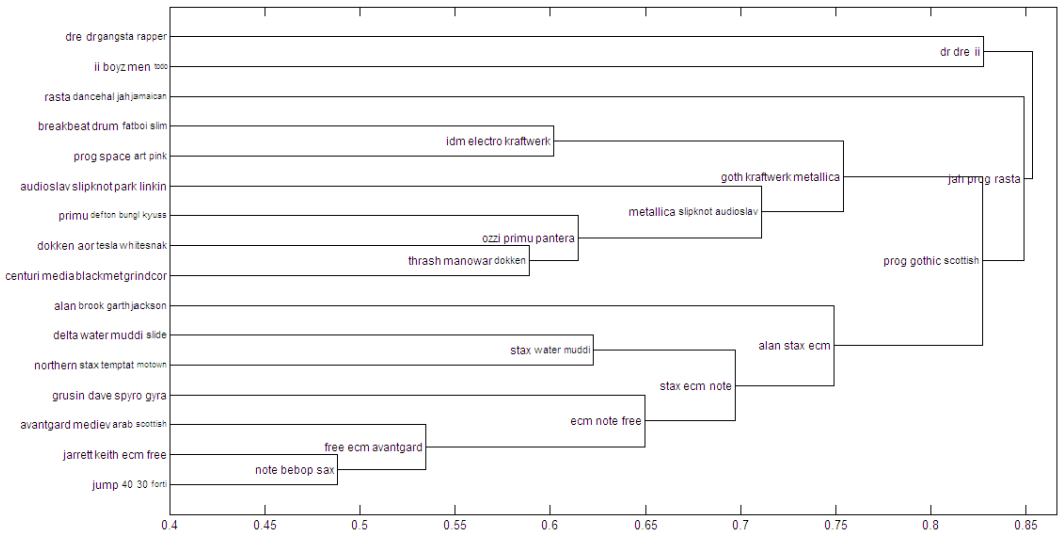

Figure 39: Dendrogram based on the artist data from [Pohle et al., 2007a], cosine similarity and average linkage algorithm.

7.4.1.1. Dendrogram zooming. The hierarchy reflected in a dendrogram can be used in user interfaces to reduce the information displayed when zooming out of visual representations of the music collections. For example, in an application context like [Schnitzer et al., 2007], this method allows to give only coarse descriptions of the regions of the bar representing the music on the player, together with a color representation of their (relative) sizes. When zooming into a particular region, this region can be split up and more fine-grained labels can be shown. This zooming can also be done seamlessly, as two levels of abstraction may be shown simultaneously.

7.4.1.2. Constrained Random Play Function. On mobile devices having displays with limited screen resolution, the hierarchical dendrogram structure may be used to quickly browse the collection based on a meaningful organization. I.e., instead of an artist / album / track browsing procedure, the selection would take place top-down, narrowing the kind of music the listener wants to listen to at the moment. She can stop the procedure at any time, and all tracks that are contained under the current position in the tree may be played, e.g. in a random manner. For example, in a hierarchy as in Figure 40, the user may choose to play just pop music by browsing to the branch labelled with *"80, pop, classic"*. Obviously, this method to access only a particular kind of music can be also used in software players, and combined with other search and access methods. For example, the selected music can be further narrowed

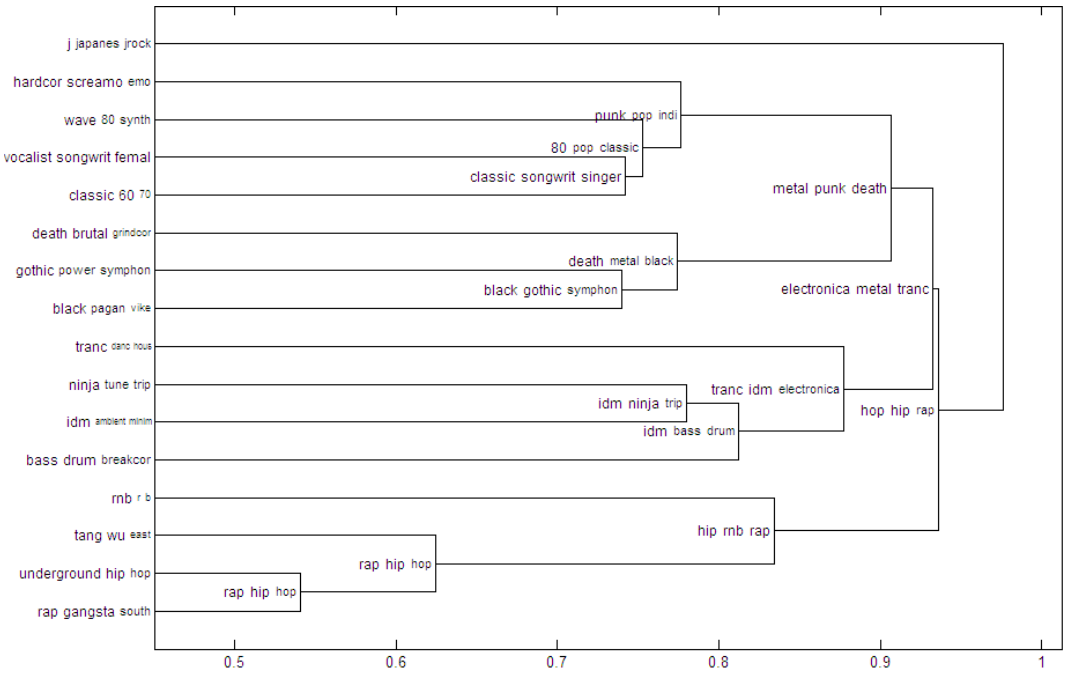

Figure 40: Dendrogram based on the artist data from [Pohle et al., 2007b], cosine similarity and average linkage algorithm.

by a metadata search for only allowing music from a given decade.

7.4.2. Automatically Keeping the Collection Up-To-Date

Another potential application scenario is assessed in [Pohle et al., 2008]. The idea in this scenario is to exploit information about the user's listening behavior to automatically update the music she has on her mobile player. It is assumed that the number of times a track is listened to or skipped contains information about the user's preferences. While such a scenario is difficult to evaluate due to the large number of rather unpredictable influences on the actual usage behavior, the conducted experiments show that combining statistics about listening and skipping songs with an audio similarity measure can potentially be used to support the user in finding new music she likes. As such a recommender system is independent of metadata and artist popularity this is an example of a system that is able to deliver music from the long tail.

7.5. Conclusions

In the final chapter of this thesis, a number of example applications has been presented that can make use of the methods discussed and evaluated in the previous chapters. They motivate a number of alternative ways to interact with music collections, and maybe, these example applications even give a first taste of the way music is consumed in the future.

A. Experimental Results

A.1. Sorted List of TFxIDF Approaches

Below, the sorted list of TFxIDF approaches discussed in Section 3.3 is listed. The number gives the maximum rank of the approach based on the 323a set and the 3000a set. Entries have the form <PageAggregationFunction>.<TF-Approach>.<IDF-Approach>.<SimilarityMeasure>.

6. numPagesAbs.tf_c3.idf_h.cosSim
7. numPagesAbs.tf_c2.idf_h.cosSim
10. mean.tf_f.idf_e.jeff
11. mean.tf_c2.idf_e.jeff
11. numPagesAbs.tf_c.idf_h.cosSim
12. sum.tf_c2.idf_b2.cosSim
15. sum.tf_c3.idf_b2.cosSim
31. numPagesAbs.tf_c3.idf_b2.cosSim
32. sum.tf_c2.idf_h.cosSim
42. mean.tf_f.idf_b2.jeff
43. sum.tf_c3.idf_h.cosSim
44. mean.tf_c2.idf_b2.jeff
44. numPagesRel.tf_f.idf_b2.cosSim
45. numPagesRel.tf_f.idf_b2.dice
45. sum.tf_c3.idf_b2.dice
46. numPagesRel.tf_f.idf_b2.jacc
46. sum.tf_c3.idf_b2.jacc
47. numPagesRel.tf_f.idf_h.dice
48. numPagesRel.tf_f.idf_h.jacc
48. sum.tf_c.idf_b2.cosSim
49. numPagesRel.tf_f.idf_i.dice
50. numPagesRel.tf_f.idf_i.jacc
51. numPagesRel.tf_c2.idf_b2.cosSim
54. numPagesAbs.tf_b.idf_h.cosSim
55. numPagesAbs.tf_d.idf_h.cosSim
57. sum.tf_c2.idf_b2.jacc
58. sum.tf_c2.idf_b2.dice
60. sum.tf_c.idf_b2.dice
61. sum.tf_c.idf_b2.jacc
62. sum.tf_c.idf_h.cosSim
67. numPagesAbs.tf_b.idf_e.jeff
68. numPagesAbs.tf_d.idf_e.jeff
68. sum.tf_c2.idf_i.cosSim
69. numPagesRel.tf_b.idf_e.jeff
70. numPagesRel.tf_d.idf_e.jeff
71. numPagesAbs.tf_c2.idf_b2.cosSim
71. numPagesRel.tf_f.idf_h.cosSim
72. numPagesRel.tf_c2.idf_e.jeff
75. numPagesAbs.tf_b.idf_b2.cosSim
76. numPagesAbs.tf_d.idf_b2.cosSim
76. numPagesRel.tf_c2.idf_h.dice
77. numPagesRel.tf_b.idf_b2.cosSim
77. numPagesRel.tf_c2.idf_h.jacc
78. numPagesRel.tf_d.idf_b2.cosSim
78. numPagesRel.tf_f.idf_i.cosSim
80. numPagesRel.tf_b.idf_i.cosSim
81. numPagesRel.tf_c2.idf_i.dice
82. numPagesRel.tf_c2.idf_i.jacc
83. numPagesRel.tf_d.idf_i.cosSim
84. numPagesRel.tf_b.idf_h.cosSim
85. numPagesAbs.tf_b.idf_h.jeff
85. numPagesRel.tf_d.idf_h.cosSim
86. numPagesAbs.tf_d.idf_h.jeff
87. numPagesRel.tf_c2.idf_i.cosSim
88. numPagesRel.tf_c2.idf_h.cosSim
89. sum.tf_c3.idf_i.cosSim
90. numPagesAbs.tf_c2.idf_i.cosSim
91. numPagesRel.tf_f.idf_e.jeff
98. mean.tf_f.idf_h.jeff
99. numPagesAbs.tf_d.idf_h.dice
100. numPagesAbs.tf_d.idf_h.jacc
101. numPagesRel.tf_c2.idf_b2.dice
102. numPagesRel.tf_c2.idf_b2.jacc
103. sum.tf_c.idf_i.cosSim
105. mean.tf_f.idf_b2.cosSim
105. numPagesAbs.tf_c.idf_i.cosSim

109. numPagesAbs.tf_c3.idf_i.cosSim
110. numPagesRel.tf_d.idf_h.dice
111. numPagesRel.tf_d.idf_h.jacc
112. numPagesRel.tf_d.idf_i.dice
113. numPagesRel.tf_d.idf_i.jacc
115. numPagesAbs.tf_c.idf_h.dice
116. numPagesAbs.tf_c.idf_h.jacc
118. numPagesRel.tf_b.idf_i.dice
119. numPagesAbs.tf_c3.idf_h.dice
119. numPagesRel.tf_b.idf_i.jacc
120. numPagesAbs.tf_c3.idf_h.jacc
120. numPagesRel.tf_b.idf_h.dice
121. numPagesAbs.tf_d.idf_b2.jeff
121. numPagesRel.tf_b.idf_h.jacc
122. mean.tf_f.idf_i.jeff
122. numPagesRel.tf_b.idf_b2.jeff
123. numPagesAbs.tf_c.idf_b2.cosSim
123. numPagesRel.tf_d.idf_b2.jeff
124. numPagesAbs.tf_b.idf_b2.jeff
126. mean.tf_f.idf_b2.dice
127. mean.tf_f.idf_b2.jacc
128. sum.tf_c3.idf_e.dice
129. numPagesAbs.tf_c2.idf_h.dice
129. sum.tf_c3.idf_e.jacc
130. numPagesAbs.tf_c2.idf_h.jacc
130. sum.tf_c2.idf_e.dice
131. sum.tf_c2.idf_e.jacc
134. sum.tf_c2.idf_e.cosSim
135. sum.tf_c3.idf_e.cosSim
136. mean.tf_b.idf_e.jeff
137. mean.tf_d.idf_e.jeff
137. sum.tf_c2.idf_h.dice
138. sum.tf_b.idf_e.jeff
138. sum.tf_c2.idf_h.jacc
139. sum.tf_d.idf_e.jeff
142. numPagesRel.tf_b.idf_b2.dice
143. numPagesRel.tf_b.idf_b2.jacc
143. sum.tf_c3.idf_h.dice
144. sum.tf_c3.idf_h.jacc
145. sum.tf_c.idf_h.dice
146. numPagesAbs.tf_d.idf_b2.dice
146. sum.tf_c.idf_h.jacc
147. numPagesAbs.tf_d.idf_b2.jacc
148. numPagesAbs.tf_c2.idf_b2.dice
148. numPagesRel.tf_d.idf_b2.dice
149. numPagesAbs.tf_c2.idf_b2.jacc
149. numPagesRel.tf_d.idf_b2.jacc
150. numPagesAbs.tf_c3.idf_e.cosSim
151. sum.tf_c.idf_e.dice
152. numPagesAbs.tf_c3.idf_b2.dice
152. sum.tf_c.idf_e.jacc
153. numPagesAbs.tf_c3.idf_b2.jacc
153. sum.tf_c.idf_e.cosSim
154. numPagesRel.tf_c2.idf_b2.jeff
154. numPagesRel.tf_f.idf_e.cosSim
155. mean.tf_c2.idf_h.jeff
155. numPagesAbs.tf_c.idf_b2.dice
156. numPagesAbs.tf_c.idf_b2.jacc
156. numPagesRel.tf_f.idf_e.dice
157. numPagesRel.tf_f.idf_b2.jeff
157. numPagesRel.tf_f.idf_e.jacc
158. numPagesAbs.tf_c3.idf_e.dice
159. numPagesAbs.tf_c3.idf_e.jacc
162. numPagesAbs.tf_c2.idf_e.cosSim
163. numPagesRel.tf_c2.idf_e.cosSim
164. mean.tf_c2.idf_i.jeff
165. numPagesRel.tf_b.idf_i.jeff
165. sum.tf_c3.idf_e.jeff
166. numPagesAbs.tf_c2.idf_e.dice
166. numPagesRel.tf_d.idf_i.jeff
167. numPagesAbs.tf_c2.idf_e.jacc
168. numPagesRel.tf_c2.idf_e.dice
169. numPagesRel.tf_b.idf_h.jeff
169. numPagesRel.tf_c2.idf_e.jacc
170. numPagesAbs.tf_c.idf_e.cosSim
170. numPagesRel.tf_d.idf_h.jeff
171. sum.tf_c2.idf_e.jeff
175. numPagesAbs.tf_c.idf_e.dice
175. numPagesRel.tf_c2.idf_i.jeff
176. numPagesAbs.tf_c.idf_e.jacc
176. numPagesRel.tf_c2.idf_h.jeff
177. mean.tf_f.idf_h.cosSim
178. mean.tf_f.idf_i.cosSim
178. sum.tf_c.idf_i.dice
179. numPagesAbs.tf_d.idf_e.cosSim
179. sum.tf_c.idf_i.jacc

180. numPagesRel.tf_b.idf_e.cosSim
180. numPagesRel.tf_f.idf_h.jeff
181. numPagesRel.tf_d.idf_e.cosSim
181. numPagesRel.tf_f.idf_i.jeff
182. numPagesAbs.tf_b.idf_e.cosSim
182. sum.tf_c2.idf_i.dice
183. sum.tf_c2.idf_i.jacc
187. sum.tf_c3.idf_i.dice
188. sum.tf_c3.idf_i.jacc
190. sum.tf_c3.idf_b2.jeff
191. numPagesAbs.tf_c3.idf_e.jeff
191. numPagesAbs.tf_d.idf_e.dice
192. numPagesAbs.tf_d.idf_e.jacc
192. numPagesAbs.tf_f.idf_h.cosSim
193. numPagesRel.tf_d.idf_e.dice
194. numPagesRel.tf_d.idf_e.jacc
195. numPagesRel.tf_b.idf_e.dice
195. sum.tf_c2.idf_b2.jeff
196. numPagesRel.tf_b.idf_e.jacc
197. mean.tf_f.idf_i.dice
198. mean.tf_f.idf_i.jacc
198. sum.tf_c2.idf_h.jeff
199. mean.tf_f.idf_h.dice
199. sum.tf_c3.idf_h.jeff
200. mean.tf_f.idf_h.jacc
203. sum.tf_c.idf_e.jeff
205. numPagesAbs.tf_b.idf_h.dice
206. numPagesAbs.tf_b.idf_h.jacc
206. numPagesAbs.tf_c3.idf_h.jeff
207. numPagesAbs.tf_f.idf_i.cosSim
208. numPagesAbs.tf_f.idf_h.dice
209. mean.tf_f.idf_e.cosSim
209. numPagesAbs.tf_f.idf_h.jacc
211. numPagesAbs.tf_c3.idf_b2.jeff
212. numPagesAbs.tf_c2.idf_h.jeff
214. numPagesAbs.tf_c2.idf_e.jeff
216. sum.tf_c.idf_b2.jeff
218. sum.tf_c.idf_h.jeff
219. mean.tf_c2.idf_b2.cosSim
219. numPagesAbs.tf_c2.idf_b2.jeff
220. numPagesAbs.tf_c.idf_e.jeff
223. numPagesAbs.tf_c.idf_h.jeff
226. sum.tf_f.idf_h.cosSim
227. mean.tf_b.idf_b2.jeff
227. numPagesAbs.tf_f.idf_i.dice
228. mean.tf_d.idf_b2.jeff
228. numPagesAbs.tf_f.idf_i.jacc
229. numPagesAbs.tf_c.idf_b2.jeff
229. sum.tf_b.idf_b2.jeff
230. numPagesAbs.tf_d.idf_i.dice
230. sum.tf_d.idf_b2.jeff
231. numPagesAbs.tf_d.idf_i.jacc
233. mean.tf_c2.idf_b2.dice
234. mean.tf_c2.idf_b2.jacc
234. sum.tf_f.idf_i.cosSim
237. numPagesAbs.tf_c.idf_i.dice
238. numPagesAbs.tf_c.idf_i.jacc
239. sum.tf_c3.idf_i.jeff
239. sum.tf_f.idf_b2.dice
240. sum.tf_c2.idf_i.jeff
240. sum.tf_f.idf_b2.jacc
241. numPagesAbs.tf_c2.idf_i.dice
241. numPagesAbs.tf_f.idf_b2.dice
242. numPagesAbs.tf_c2.idf_i.jacc
242. numPagesAbs.tf_f.idf_b2.jacc
243. mean.tf_f.idf_e.jacc
243. sum.tf_f.idf_b2.cosSim
244. mean.tf_f.idf_e.dice
244. numPagesAbs.tf_f.idf_b2.cosSim
245. numPagesAbs.tf_b.idf_b2.dice
245. sum.tf_f.idf_e.dice
246. numPagesAbs.tf_b.idf_b2.jacc
246. sum.tf_f.idf_e.jacc
247. sum.tf_f.idf_e.cosSim
247. sum.tf_f.idf_h.dice
248. numPagesAbs.tf_f.idf_e.dice
248. sum.tf_f.idf_h.jacc
249. numPagesAbs.tf_f.idf_e.jacc
250. numPagesAbs.tf_f.idf_e.cosSim
253. numPagesAbs.tf_d.idf_i.jeff
254. numPagesRel.tf_c.idf_i.dice
255. numPagesRel.tf_c.idf_h.dice
255. numPagesRel.tf_c.idf_i.jacc
256. numPagesAbs.tf_f.idf_e.jeff
256. numPagesRel.tf_c.idf_h.jacc
257. numPagesAbs.tf_b.idf_i.jeff

267. numPagesAbs.tf_b.idf_e.dice
268. numPagesAbs.tf_b.idf_e.jacc
269. numPagesAbs.tf_f.idf_h.jeff
270. numPagesAbs.tf_c3.idf_i.dice
271. numPagesAbs.tf_c3.idf_i.jacc
272. numPagesAbs.tf_d.idf_i.cosSim
273. numPagesAbs.tf_b.idf_i.cosSim
274. sum.tf_c.idf_i.jeff
275. numPagesRel.tf_c.idf_h.cosSim
276. numPagesRel.tf_c.idf_i.cosSim
277. mean.tf_c2.idf_h.cosSim
278. mean.tf_c2.idf_i.cosSim
279. mean.tf_c2.idf_h.dice
280. mean.tf_c2.idf_h.jacc
280. numPagesAbs.tf_f.idf_b2.jeff
281. mean.tf_c.idf_h.jeff
281. sum.tf_f.idf_i.dice
282. mean.tf_c2.idf_i.dice
282. sum.tf_f.idf_i.jacc
283. mean.tf_c2.idf_i.jacc
283. sum.tf_f.idf_e.jeff
288. mean.tf_c.idf_i.jeff
288. numPagesAbs.tf_c2.idf_i.jeff
289. mean.tf_b.idf_h.jeff
289. numPagesAbs.tf_c3.idf_i.jeff
290. mean.tf_d.idf_h.jeff
291. sum.tf_b.idf_h.jeff
292. mean.tf_c2.idf_e.cosSim
292. sum.tf_d.idf_h.jeff
293. mean.tf_c.idf_b2.jeff
293. numPagesAbs.tf_c.idf_i.jeff
296. mean.tf_b.idf_i.jeff
296. sum.tf_f.idf_b2.jeff
297. mean.tf_d.idf_i.jeff
297. numPagesRel.tf_c3.idf_h.cosSim
298. numPagesRel.tf_c3.idf_i.cosSim
300. mean.tf_c3.idf_h.jeff
301. sum.tf_b.idf_i.jeff
302. sum.tf_d.idf_i.jeff
303. mean.tf_c.idf_h.cosSim
304. mean.tf_b.idf_b2.cosSim
305. mean.tf_d.idf_b2.cosSim
305. numPagesAbs.tf_f.idf_i.jeff
306. sum.tf_b.idf_b2.cosSim
307. sum.tf_d.idf_b2.cosSim
307. sum.tf_f.idf_h.jeff
308. mean.tf_c2.idf_e.dice
308. mean.tf_d.idf_b2.dice
309. mean.tf_c2.idf_e.jacc
309. mean.tf_d.idf_b2.jacc
310. sum.tf_d.idf_b2.dice
311. sum.tf_d.idf_b2.jacc
312. mean.tf_b.idf_b2.dice
313. mean.tf_b.idf_b2.jacc
314. mean.tf_c.idf_h.dice
315. mean.tf_c.idf_h.jacc
315. sum.tf_f.idf_i.jeff
316. mean.tf_c3.idf_i.jeff
317. mean.tf_c.idf_i.cosSim
319. numPagesRel.tf_c3.idf_h.dice
320. mean.tf_c.idf_i.dice
320. numPagesRel.tf_c3.idf_h.jacc
321. mean.tf_c.idf_i.jacc
321. numPagesRel.tf_c3.idf_i.dice
322. mean.tf_c3.idf_h.cosSim
322. numPagesRel.tf_c3.idf_i.jacc
323. mean.tf_c3.idf_b2.jeff
324. mean.tf_c3.idf_h.dice
325. mean.tf_c3.idf_h.jacc
325. numPagesRel.tf_c.idf_i.jeff
326. mean.tf_c3.idf_i.cosSim
326. numPagesRel.tf_c.idf_h.jeff
327. mean.tf_c3.idf_i.dice
327. sum.tf_b.idf_e.cosSim
328. mean.tf_b.idf_e.cosSim
328. mean.tf_c3.idf_i.jacc
329. mean.tf_d.idf_e.cosSim
330. sum.tf_d.idf_e.cosSim
331. numPagesRel.tf_c.idf_b2.dice
332. numPagesRel.tf_c.idf_b2.jacc
332. sum.tf_b.idf_b2.dice
333. numPagesRel.tf_c.idf_b2.jeff
333. sum.tf_b.idf_b2.jacc
334. mean.tf_c.idf_b2.dice
335. mean.tf_c.idf_b2.jacc
336. mean.tf_b.idf_e.dice

336. numPagesRel.tf_c3.idf_b2.dice
337. mean.tf_b.idf_e.jacc
337. numPagesRel.tf_c3.idf_b2.jacc
338. mean.tf_c.idf_b2.cosSim
338. mean.tf_d.idf_e.dice
339. mean.tf_d.idf_e.jacc
339. numPagesRel.tf_c.idf_b2.cosSim
340. sum.tf_d.idf_e.dice
341. mean.tf_c3.idf_b2.dice
341. sum.tf_d.idf_e.jacc
342. mean.tf_c3.idf_b2.jacc
343. mean.tf_c3.idf_b2.cosSim
344. numPagesAbs.tf_b.idf_i.dice
344. numPagesRel.tf_c3.idf_b2.cosSim
345. numPagesAbs.tf_b.idf_i.jacc
346. numPagesRel.tf_c3.idf_h.jeff
347. numPagesRel.tf_c3.idf_i.jeff
347. sum.tf_d.idf_h.dice
348. numPagesRel.tf_c3.idf_b2.jeff
348. sum.tf_d.idf_h.jacc
349. mean.tf_b.idf_h.cosSim
349. sum.tf_b.idf_e.dice
350. mean.tf_d.idf_h.cosSim
350. sum.tf_b.idf_e.jacc
351. sum.tf_b.idf_h.cosSim
352. sum.tf_d.idf_h.cosSim
353. mean.tf_b.idf_h.dice
354. mean.tf_b.idf_h.jacc
355. mean.tf_d.idf_h.dice
356. mean.tf_d.idf_h.jacc
357. sum.tf_b.idf_h.dice
358. sum.tf_b.idf_h.jacc
359. mean.tf_d.idf_i.jacc
360. mean.tf_d.idf_i.dice
361. mean.tf_b.idf_i.cosSim
362. mean.tf_d.idf_i.cosSim
366. sum.tf_b.idf_i.cosSim
367. mean.tf_b.idf_i.dice
367. sum.tf_d.idf_i.cosSim
368. mean.tf_b.idf_i.jacc
369. sum.tf_d.idf_i.dice
370. sum.tf_d.idf_i.jacc
374. mean.tf_c3.idf_e.cosSim
374. mean.tf_c3.idf_e.dice
375. mean.tf_c3.idf_e.jacc
375. numPagesRel.tf_c3.idf_e.dice
376. mean.tf_c.idf_e.cosSim
376. numPagesRel.tf_c3.idf_e.jacc
377. mean.tf_c.idf_e.dice
377. numPagesRel.tf_c3.idf_e.cosSim
378. mean.tf_c.idf_e.jacc
378. numPagesRel.tf_c.idf_e.dice
379. numPagesRel.tf_c.idf_e.jacc
381. mean.tf_c.idf_e.jeff
381. numPagesRel.tf_c.idf_e.cosSim
382. mean.tf_c3.idf_e.jeff
383. numPagesRel.tf_c.idf_e.jeff
383. sum.tf_b.idf_i.dice
384. numPagesRel.tf_c3.idf_e.jeff
384. sum.tf_b.idf_i.jacc

A.2. Term Clusters

Here, some more concept decompositions as created by the methods discussed in Chapter 5 are listed.

0.85	progressive rock	0.32	real country	0.24	kyuss
0.26	prog	0.30	alan jackson	0.19	danzig
0.13	avant-prog	0.28	garth brooks	0.16	stonerrock
0.11	progressive metal	0.24	alt country	0.15	pantera
0.09	dream theater	0.23	classic country	0.15	sevendust
0.25	northern soul	0.27	dokken	0.98	noisepop
0.23	slow jams	0.26	aor	0.14	swedish music
0.22	beach music	0.24	ratt	0.07	delta blues
0.21	motown	0.21	hair metal	0.05	tribal metal
0.19	philly soul	0.19	melodic rock	0.04	r n b
0.33	black metal	0.20	blue note	1.00	ihs
0.27	death metal	0.17	big band	0.03	reggie
0.22	blackmetal	0.17	bebop	0.02	westcoast
0.18	century media	0.14	hard bop	0.02	muusika
0.18	power metal	0.14	tenor sax	0.01	jackson family
0.30	free jazz	0.88	classic folk	0.46	rasta
0.26	sun ra	0.30	folk revival	0.40	roots reggae
0.24	ecm	0.10	celtic	0.29	jamaican
0.18	avant-garde	0.08	american roots	0.29	jah
0.16	archie	0.08	folk rock	0.25	dancehall
0.98	tag 1	0.27	tresor	0.34	muddy waters
0.08	fucking awesome	0.25	detroit techno	0.31	delta blues
0.05	non-japanese	0.21	deep house	0.29	blues guitar
0.05	current 93	0.18	progressive house	0.27	electric blues
0.05	favourite jazz	0.17	techno	0.19	slide guitar
0.34	east coast rap				
0.22	classic hip hop				
0.20	old school rap				
0.19	hardcore rap				
0.17	big daddy kane				

Table 16: AS-GO: Concepts found by the method like [Xu et al., 2003], NCW.

0.18	funkmaster flex	0.27	classic reggae	0.14	progressive house
0.16	lox	0.24	rastafari	0.14	detroit techno
0.15	method man	0.23	roots reggae	0.13	deep house
0.14	gza	0.23	contemporary reggae	0.13	acid house
0.14	east coast rap	0.22	conscious reggae	0.13	dnb
0.24	real country	0.10	team sleep	0.23	british blues
0.21	hot country	0.10	cky	0.21	soul-blues
0.21	honkytonk	0.10	melodic rock	0.20	downhome
0.18	modern country	0.09	hair band	0.18	old blues
0.18	outlaw country	0.09	josh homme	0.18	classic blues
0.25	indie rap	0.17	contemporary classical	0.29	classic folk
0.24	good rap	0.15	jamband	0.27	folk revival
0.24	hip hop and rap	0.15	avantgarde	0.16	new riders of the purple sage
0.24	music to download	0.14	mike patton	0.12	woman singer
0.23	russian rock	0.13	achim	0.11	traditional pop
0.18	grp records	0.10	candlemass	0.21	the vibrators
0.16	jazz vocals	0.10	blackmetal	0.20	atmos
0.16	the forties	0.10	extreme metal	0.20	outlawz
0.15	rat pack	0.09	goth metal	0.20	t power
0.13	jazz trumpet	0.09	doom metal	0.19	shanice
0.20	essential jazz	0.20	philly soul	0.33	avant-prog
0.20	jazz master	0.19	pop-soul	0.28	kraut
0.19	great jazz musicians	0.18	smooth soul	0.24	progressive metal
0.15	good jazz	0.17	soul artists	0.24	space rock
0.14	hard-bop	0.17	psychedelic soul	0.18	dream theater
0.23	jeff lorber				
0.18	peter white				
0.18	spyro gyra				
0.17	dave grusin				
0.17	jazz-funk				

Table 17: AS-GO: "Robust" method, NMF, each artist vector normalized to have an Euclidean length of 1.0 before NMF calculation.

0.34	scientific	0.42	recorder	0.31	garth
0.25	organic	0.40	oboe	0.30	honky
0.19	echo	0.32	harmonica	0.30	tonk
0.18	industrial	0.26	trombone	0.26	nashville
0.16	sampler	0.25	clarinet	0.22	brooks
0.24	aphex	0.30	dancehall	0.27	slayer
0.22	daft	0.29	ska	0.25	sabbath
0.22	downtempo	0.26	calypso	0.21	sepultura
0.21	breakbeat	0.24	marley	0.20	anthrax
0.19	kraftwerk	0.24	tosh	0.19	megadeth
0.28	synthpop	0.33	passive	0.30	gaye
0.28	eurodance	0.31	archaic	0.30	temptations
0.26	goa	0.26	blocks	0.30	aretha
0.25	gothic	0.26	silly	0.25	cooke
0.24	grunge	0.25	trad	0.24	redding
0.29	accordion	0.52	seeger	0.34	dre
0.26	choral	0.36	baez	0.33	gangsta
0.24	tuba	0.24	townes	0.28	notorious
0.24	patriotic	0.23	zandt	0.21	missy
0.21	xylophone	0.21	joan	0.20	ghetto
0.30	brahms	0.24	bebop	0.58	evergreens
0.29	mozart	0.24	bop	0.55	hardrock
0.26	beethoven	0.22	sextet	0.49	cabaret
0.24	haydn	0.21	alto	0.13	disco
0.23	bach	0.19	afro	0.11	techno
0.37	mayall				
0.30	muddy				
0.30	hooker				
0.29	broonzy				
0.25	otis				

Table 18: TD-EXA: LTC variant, NMF after normalizing each artist's TFxIDF vector to an Euclidean length of 1.0.

0.39	recorder	0.56	hong	0.38	muddy
0.38	oboe	0.56	kong	0.31	broonzy
0.31	harmonica	0.41	worldwide	0.29	otis
0.28	trombone	0.14	dynamics	0.28	mayall
0.27	clarinet	0.11	ideal	0.25	hooker
0.49	bop	0.27	eurodance	0.35	gangsta
0.37	bebop	0.27	grunge	0.34	dre
0.26	tenor	0.27	synthpop	0.31	notorious
0.20	alto	0.25	goa	0.24	ghetto
0.18	sextet	0.23	rave	0.20	enemy
0.72	seeger	0.41	honky	0.54	dixieland
0.33	baez	0.39	tonk	0.31	orleans
0.20	dylan	0.35	nashville	0.26	armstrong
0.17	joan	0.26	hank	0.22	ellington
0.15	revival	0.25	garth	0.21	ragtime
0.58	evergreens	0.31	wop	0.13	joseph
0.55	hardrock	0.30	gaye	0.13	fusion
0.49	cabaret	0.30	cooke	0.12	improvisation
0.17	disco	0.29	temptations	0.12	college
0.14	techno	0.28	aretha	0.12	improvised
0.32	sabbath	0.58	archaic	0.45	dancehall
0.27	slayer	0.46	silly	0.40	ska
0.25	thrash	0.37	blocks	0.31	marley
0.23	priest	0.23	garage	0.30	calypso
0.20	megadeth	0.19	randy	0.25	dub
0.21	breakbeat				
0.21	aphex				
0.21	downtempo				
0.20	ambient				
0.19	experimental				

Table 19: TD-EXA: "Robust" method with NMF, after normalizing each artist's TFxIDF vector to an Euclidean length of 1.0.

B. References

[mpe, a] ISO/IEC 15938 Information technology – Mpeg7 Multimedia content description interface – Part 4: Audio.

[Abdallah, 2002] Abdallah, S. A. (2002). Towards Music Perception by Redundancy Reduction and Unsupervised Learning in Probabilistic Models. PhD thesis, Department of Electronic Engineering, King's College London.

[Abdallah and Plumbley, 2001] Abdallah, S. A. and Plumbley, M. D. (2001). If The Independent Components Of Natural Images Are Edges, What Are The Independent Components Of Natural Sounds? In Proceedings of International Conference on Independent Component Analysis and Signal Separation (ICA2001).

[Ahn and Dabbish, 2004] Ahn, L. v. and Dabbish, L. (2004). Labeling Images with a Computer Game. In ACM Conference on Human Factors in Computing Systems (CHI).

[Ahrendt, 2005] Ahrendt, P. (2005). The Multivariate Gaussian Probability Distribution. Technical report IMM, Technical University of Denmark.

[Aucouturier, 2006] Aucouturier, J.-J. (2006). Ten Experiments on the Modelling of Polyphonic Timbre. PhD thesis, University of Paris 6, France.

[Aucouturier and Pachet, 2004] Aucouturier, J.-J. and Pachet, F. (2004). Improving Timbre Similarity: How High is the Sky? Journal of Negative Results in Speech and Audio Sciences *1*.

[Aucouturier et al., 2007] Aucouturier, J.-J., Pachet, F., Roy, P. and Beurivé, A. (2007). Signal + Context = Better Classification. In Proceedings of the 8^{th} International Conference on Music Information Retrieval (ISMIR'07).

[Avery and Zeckhauser, 1997] Avery, C. and Zeckhauser, R. (1997). Recommender Systems for Evaluating Computer Messages. Communications of the ACM *40*, 88–89.

[Barrington et al., 2007] Barrington, L., Turnbull, D., Torres, D. and Lanckriet, G. (2007). Semantic Similarity for Music Retrieval. Submission to MIREX 2007.

[Barrington et al., 2009] Barrington, L., Turnbull, D., Yazdani, M. and Lanckriet, G. (2009). Combining Audio Content and Social Context for Semantic Music Discovery. In Proceedings of the 32nd Annual ACM SIGIR Conference.

[Baumann and Hummel, 2003] Baumann, S. and Hummel, O. (2003). Using Cultural Metadata for Artist Recommendation. In Proceedings of the Conference on Web Delivering of Music (Wedelmusic 2002).

[Berenzweig et al., 2003] Berenzweig, A., Logan, B., Ellis, D. P. W. and Whitman, B. (2003). A Large-Scale Evaluation of Acoustic and Subjective Music Similarity Measures. In Proceedings of the 4^{th} International Conference on Music Information Retrieval (ISMIR'03).

[Bosteels and Kerre, 2007a] Bosteels, K. and Kerre, E. E. (2007a). Combining Audio Similarity Measures Using Generalizations of Logical Connectives. In 4th Annual Music Information Retrieval Evaluation Exchange.

[Bosteels and Kerre, 2007b] Bosteels, K. and Kerre, E. E. (2007b). Fuzzy Audio Similarity Measures Based on Spectrum Histograms and Fluctuation Patterns. In Proceedings of the 2007 International Conference on Multimedia and Ubiquitous Engineering (MUE07).

[Bouvrie et al., 2008] Bouvrie, J., Ezzat, T. and Poggio, T. (2008). Localized Spectro-Temporal Cepstral Analysis of Speech. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2008).

[Briët and Harremoës, 2009] Briët, J. and Harremoës, P. (2009). Properties of Classical and Quantum Jensen-Shannon Divergence. Physical Review A (Atomic, Molecular, and Optical Physics) *79*, 052311.

[Buckley and Voorhees, 2000] Buckley, C. and Voorhees, E. (2000). Evaluating Evaluation Measure Stability. In Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.

[Celma, 2008] Celma, O. (2008). Music Recommendation and Discovery in the Long Tail. PhD thesis, Universitat Pompeu Fabra.

[Celma et al., 2006] Celma, O., Cano, P. and Herrera, P. (2006). Search Sounds: An Audio Crawler Focused on Weblogs. In Proceedings of the 7^{th} International Conference on Music Information Retrieval (ISMIR'06).

[Celma et al., 2005] Celma, O., Ramírez, M. and Herrera, P. (2005). Foafing the Music: A Music Recommendation System Based on RSS Feeds and User Preferences. In Proceedings of the 6^{th} International Conference on Music Information Retrieval (ISMIR'05).

[Cho et al., 2003] Cho, Y.-C., Choi, S. and Bang, S.-Y. (2003). Non-Negative Component Parts of Sound for Classification. In Proceedings of the 3rd IEEE International Symposium on Signal Processing and Information Technology (ISSPIT 2003).

[Ciaccia et al., 1997] Ciaccia, P., Patella, M. and Zezula, P. (1997). M-tree: An Efficient Access Method for Similarity Search in Metric Spaces. In Proceedings of the 23^{rd} International Conference on Very Large Databases.

[Cohen and Fan, 2000] Cohen, W. W. and Fan, W. (2000). Web-Collaborative Filtering: Recommending Music by Crawling The Web. Computer Networks *33*, 685–698.

[Comaniciu et al., 2003] Comaniciu, D., Ramesh, V. and Meer, P. (2003). Kernel-Based Object Tracking. Transactions on Pattern Analysis and Machine Intelligence *25*, 564–577.

[Cross and Sudkamp, 2002] Cross, V. and Sudkamp, T. (2002). Similarity and Compatibility in Fuzzy Set Theory. Physica-Verlag.

[Deshpande et al., 2001] Deshpande, H., Singh, R. and Nam, U. (2001). Classification of Music Signals in the Visual Domain. In Proceedings of the COST G-6 Conference on Digital Audio Effects (DAFX-01).

[Diaconis and Zabell, 1982] Diaconis, P. and Zabell, S. L. (1982). Updating Subjective Probability. Journal of the American Statistical Association *77*, 822–830.

[Downie, 2006] Downie, J. S. (2006). The Music Information Retrieval Evaluation eXchange (MIREX). D-Lib Magazine *12*.

[Eck et al., 2007] Eck, D., Bertin-Mahieux, T. and Lamere, P. (2007). Autotagging Music Using Supervised Machine Learning. In Proceedings of the 8^{th} International Conference on Music Information Retrieval (ISMIR'07).

[Ellis, 2006] Ellis, T. (2006). Beat Tracking with Dynamic Programming. In 3rd Annual Music Information Retrieval Evaluation Exchange.

[Faloutsos and Lin, 1995] Faloutsos, C. and Lin, K.-I. D. (1995). FastMap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets. In Proceedings of the 1995 ACM SIGMOD international conference on Management of data.

[Fastl and Zwicker, 2007] Fastl, H. and Zwicker, E. (2007). Psychoacoustics. Springer Series in Information Sciences, third edition, Springer.

[Fischer et al., 2002] Fischer, S., Klinkenberg, R., Mierswa, I. and Ritthoff, O. (2002). Yale: Yet Another Learning Environment – Tutorial. Technical report Collaborative Research Center 531, University of Dortmund.

[Friedman, 1937] Friedman, M. (1937). The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance. Journal of the American Statistical Association *32*, 675–701.

[Fukunaga, 1997] Fukunaga, K. (1997). Introduction to Statistical Pattern Recognition. Academic Press, Inc.

[Gaussier and Goutte, 2005] Gaussier, E. and Goutte, C. (2005). Relation Between PLSA and NMF and Implications. In Proceedings of the 28*th* Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.

[Goto, 2006] Goto, M. (2006). A Chorus-Section Detection Method for Musical Audio Signals and Its Application to a Music Listening Station. IEEE Transactions on Audio, Speech and Language Processing *14*, 1783–1794.

[Gouyon, 2005] Gouyon, F. (2005). A Computational Approach to Rhythm Description Audio Features for the Computation of Rhythm Periodicity Functions and Their Use in Tempo Induction and Music Content Processing. PhD thesis, University Pompeu Fabra.

[Gouyon and Dixon, 2004] Gouyon, F. and Dixon, S. (2004). Dance Music Classification: A Tempo-Based Approach. In Proceedings of the 5^{th} International Conference on Music Information Retrieval (ISMIR'04).

[Gouyon et al., 2006] Gouyon, F., Klapuri, A., Dixon, S., Alonso, M., Tzanetakis, G., Uhle, C. and Cano, P. (2006). An Experimental Comparison of Audio Tempo Induction Algorithms. IEEE Transactions on Speech and Audio Processing *14*.

[Homburg et al., 2005] Homburg, H., Mierswa, I., Möller, B., Morik, K. and Wurst, M. (2005). A Benchmark Dataset for Audio Classification and Clustering. In Proceedings of the 6^{th} International Conference on Music Information Retrieval (ISMIR'05).

[Hoyer, 2004] Hoyer, P. (2004). Non-Negative Matrix Factorization with Sparseness Constraints. Journal of Machine Learning Research *5*, 1457–1469.

[Hu et al., 2005] Hu, X., Downie, J. S., West, K. and Ehmann, A. (2005). Mining Music Reviews: Promising Preliminary Results. In Proceedings of the 6^{th} International Conference on Music Information Retrieval (ISMIR'05).

[Huber et al., 2008] Huber, M., Bailey, T., Durrant-Whyte, H. and Hanebeck, U. (2008). On Entropy Approximation for Gaussian Mixture Random Vectors. In IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems.

[Jebara and Kondor, 2003] Jebara, T. and Kondor, R. (2003). Bhattacharyya and Expected Liklihood Kernels. In Learning Theory and Kernel Machines: COLT 2003: Annual Conference on Learning Theory No. 16.

[Jensen et al., 2007] Jensen, J. H., Ellis, D. P., Christensen, M. G. and Jensen, S. H. (2007). Evaluation of Distance Measures Between Gaussian Mixture Models of MFCCS. In Proceedings of the 8^{th} International Conference on Music Information Retrieval (ISMIR'07).

[Jiang et al., 2002] Jiang, D.-N. J., Lu, L., Zhang, H.-J., Tao, J.-H. and Cai, L.-H. (2002). Music Type Classification by Spectral Contrast Feature. In Proceedings of the IEEE International Conference on Multimedia and Expo (ICME).

[Kailath, 1967] Kailath, T. (1967). The Divergence and Bhattacharyya Distance Measures in Signal Selection. IEEE Transactions on Communication Technology *15*, 52–60.

[Knees et al., 2004] Knees, P., Pampalk, E. and Widmer, G. (2004). Artist Classification with Web-based Data. In Proceedings of the 5^{th} International Conference on Music Information Retrieval (ISMIR'04).

[Knees et al., 2006a] Knees, P., Pohle, T., Schedl, M. and Widmer, G. (2006a). Combining Audio-based Similarity with Web-based Data to Accelerate Automatic Music Playlist Generation. In Proceedings of the 8th ACM SIGMM International Workshop on Multimedia Information Retrieval (MIR'06).

[Knees et al., 2006b] Knees, P., Schedl, M., Pohle, T. and Widmer, G. (2006b). An Innovative Three-Dimensional User Interface for Exploring Music Collections Enriched with Meta-Information from the Web. In Proceedings of the ACM Multimedia 2006.

[Knees and Pohle, 2008] Knees, P., S. M. and Pohle, T. . (2008). A Deeper Look into Web-based Classification of Music Artists. In Proceedings of the 2nd Workshop on Learning the Semantics of Audio Signals (LSAS 2008).

[Kraskov et al., 2003] Kraskov, A., Stögbauer, H., Andrzejak, R. G. and Grassberger, P. (2003). Hierarchical Clustering Based on Mutual Information. Technical report John-von-Neumann Institute for Computing, Forschungszentrum Jülich.

[Lamberti and Majtey, 2003] Lamberti, P. W. and Majtey, A. P. (2003). Non-logarithmic Jensen-Shannon divergence. Physica A *329*, 81–90.

[Lamere and Maillet, 2008] Lamere, P. and Maillet, F. (2008). Creating Transparent, Steerable Recommendations. In Late–breaking Proceedings of the 8th International Conference on Music Information Retrieval.

[Law et al., 2007] Law, E. L. M., von Ahn, L., Dannenberg, R. B. and Crawford, M. (2007). TAGATUNE: A Game for Music and Sound Annotation. In Proceedings of the 8^{th} International Conference on Music Information Retrieval (ISMIR'07).

[Le Borgne et al., 2004] Le Borgne, H., Anne, G.-D. and Antoniadis, A. (2004). Representation of Images for Classification with Independent Features. Pattern Recognition Letters *25*, 141–154.

[Lee and Seung, 2000] Lee, D. and Seung, H. (2000). Algorithms for Non-Negative Matrix Factorization. In Advances in Neural Information Processing Systems 13 (Proc. NIPS*2000).

[Lee and Seung, 1999] Lee, D. D. and Seung, H. S. (1999). Learning the Parts of Objects by Non-Negative Matrix Factorization. Nature *401*, 788–791.

[Legg et al., 2007] Legg, P. A., Rosin, P. L., Marshall, D. and Morgan, J. E. (2007). Improving Accuracy and Efficiency of Registration by Mutual Information using Sturges Histogram Rule. In Proceedings of the 11^{th} Medical Image Understanding and Analysis (MIUA) 2007 conference.

[Levy and Sandler, 2006] Levy, M. and Sandler, M. (2006). Lightweight Measures for Timbral Similarity of Musical Audio. In AMCMM '06: Proceedings of the 1st ACM Workshop on Audio and Music Computing Multimedia.

[Levy and Sandler, 2008] Levy, M. and Sandler, M. (2008). Learning Latent Semantic Models for Music from Social Tags. Journal of New Music Research *13(2)*, 137–150.

[Li et al., 2001] Li, D., Sethi, I. K., Dimitrova, N. and McGee, T. (2001). Classification of General Audio Data for Content-Based Retrieval. Pattern Recogn. Lett., *22(5)*, 533–544.

[Li and Ogihara, 2004] Li, T. and Ogihara, M. (2004). Music Artist Style Identification by Semisupervised Learning from both Lyrics and Content. In Proceedings of ACM Conference on Multimedia.

[Lin, 1991] Lin, J. (1991). Divergence Measures Based on the Shannon Entropy. IEEE Transactions on Information Theory *37*, 145–151.

[Logan, 2000] Logan, B. (2000). Mel Frequency Cepstral Coefficients for Music Modeling. In Proceedings of the First International Symposium on Music Information Retrieval (ISMIR).

[Logan, 2004] Logan, B. (2004). Music Recommendation From Song Sets. In Proceedings of the 5^{th} International Conference on Music Information Retrieval (ISMIR'04).

[Logan et al., 2003] Logan, B., Ellis, D. P. W. and Berenzweig, A. (2003). Toward Evaluation Techniques for Music Similarity. In Workshop on the Evaluation of Music Information Retrieval (MIR) Systems at SIGIR 2003. Keynote address.

[Logan and Salomon, 2001] Logan, B. and Salomon, A. (2001). A Music Similarity Function Based on Signal Analysis. In IEEE International Conference on Multimedia and Expo (ICME 2001).

[Ma and He, 2005] Ma, J. and He, Q. (2005). A Dynamic Merge-or-Split Learning Algorithm on Gaussian Mixture for Automated Model Selection. In Proceedings of 6th International Conference on Intelligent Data Engineering and Automated Learning - IDEAL.

[Mahamud and Hebert, 2003] Mahamud, S. and Hebert, M. (2003). Minimum Risk Distance Measure for Object Recognition. In Proceedings of the Ninth IEEE International Conference on Computer Vision (ICCV).

[Mandel and Ellis, 2005a] Mandel, M. and Ellis, D. (2005a). Song-Level Features and Support Vector Machines for Music Classification. In Proceedings of the 6^{th} International Conference on Music Information Retrieval (ISMIR'05).

[Mandel and Ellis, 2005b] Mandel, M. and Ellis, D. (2005b). Song-Level Features and SVMs for Music Classification. In 2nd Annual Music Information Retrieval Evaluation Exchange.

[Mandel and Ellis, 2007] Mandel, M. I. and Ellis, D. P. W. (2007). A Web-Based Game for Collecting Music Metadata. In Proceedings of the 8^{th} International Conference on Music Information Retrieval (ISMIR'07).

[McKay et al., 2005] McKay, C., Fiebrink, R., McEnnis, D., Li, B. and Fujinaga, I. (2005). ACE: A Framework for Optimizing Music Classification. In Proceedings of the 6^{th} International Conference on Music Information Retrieval (ISMIR'05).

[Mierswa and Morik, 2005] Mierswa, I. and Morik, K. (2005). Automatic Feature Extraction for Classifying Audio Data. Machine Learning Journal *58*, 127–149.

[Moerchen et al., 2006] Moerchen, F., Mierswa, I. and Ultsch, A. (2006). Understandable Models of Music Collections Based on Exhaustive Feature Generation with Temporal Statistics. In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

[Niblack et al., 1993] Niblack, C. W., Barber, R., Equitz, W., Flickner, M. D., Glasman, E. H., Petkovic, D., Yanker, P., Faloutsos, C. and Taubin, G. (1993). QBIC Project: Querying Images by Content, Using Color, Texture, and Shape. In Proceedings of the SPIE Conference on Storage and Retrieval for Image and Video Databases.

[Novello et al., 2006] Novello, A., McKinney, M. F. and Kohlrausch, A. (2006). Perceptual Evaluation of Music Similarity. In Proceedings of the 7^{th} International Conference on Music Information Retrieval (ISMIR'06).

[Ono et al., 2008] Ono, N., Miyamoto, K., Kameoka, H. and Sagayama, S. (2008). A Real-Time Equalizer of Harmonic and Percussive Components in Music Signals. In Proceedings of the 9^{th} International Conference on Music Information Retrieval (ISMIR'08).

[O'Shaughnessy, 1987] O'Shaughnessy, D. (1987). Speech Communication: Human and Machine. Addison-Wesley Publishing Co.

[Pachet et al., 2001] Pachet, F., Westermann, G. and Laigre, D. (2001). Musical Data Mining for Electronic Music Distribution. In Proceedings of the First International Conference on WEB Delivering of Music (WEDELMUSIC'01).

[Pampalk, 2001] Pampalk, E. (2001). Islands of Music: Analysis, Organization, and Visualization of Music Archives. Master's thesis Department of Software Technology and Interactive Systems, Vienna University of Technology.

[Pampalk, 2004] Pampalk, E. (2004). A Matlab Toolbox to Compute Music Similarity from Audio. In Proceedings of the 5^{th} International Conference on Music Information Retrieval (ISMIR'04).

[Pampalk, 2005] Pampalk, E. (2005). Speeding Up Music Similarity. In Implementation submitted to the 2nd Annual Music Information Retrieval eXchange (MIREX'05).

[Pampalk, 2006a] Pampalk, E. (2006a). Audio-Based Music Similarity and Retrieval: Combining a Spectral Similarity Model with Information Extracted from Fluctuation Patterns. In Implementation submitted to the 3rd Annual Music Information Retrieval eXchange (MIREX'06).

[Pampalk, 2006b] Pampalk, E. (2006b). Computational Models of Music Similarity and their Application in Music Information Retrieval. PhD thesis, Vienna University of Technology.

[Pampalk et al., 2003] Pampalk, E., Dixon, S. and Widmer, G. (2003). On the Evaluation of Perceptual Similarity Measures for Music. In Proceedings of the 6th International Conference on Digital Audio Effects (DAFx-03).

[Pampalk et al., 2004] Pampalk, E., Dixon, S. and Widmer, G. (2004). Exploring Music Collections by Browsing Different Views. Computer Music Journal *28*, 49–62.

[Pampalk et al., 2005a] Pampalk, E., Flexer, A. and Widmer, G. (2005a). Hierarchical Organization and Description of Music Collections at the Artist Level. In Proceedings of the 9^{th} European Conference on Research and Advanced Technology for Digital Libraries (ECDL'05).

[Pampalk et al., 2005b] Pampalk, E., Flexer, A. and Widmer, G. (2005b). Improvements of Audio-Based Music Similarity and Genre Classification. In Proceedings of the 6^{th} International Conference on Music Information Retrieval (ISMIR'05).

[Pampalk et al., 2005c] Pampalk, E., Pohle, T. and Widmer, G. (2005c). Dynamic Playlist Generation Based on Skipping Behaviour. In Proceedings of the 6^{th} International Conference on Music Information Retrieval (ISMIR'05).

[Pampalk et al., 2002] Pampalk, E., Rauber, A. and Merkl, D. (2002). Content-based Organization and Visualization of Music Archives. In Proceedings of ACM Multimedia 2002.

[Penny, 2001] Penny, W. (2001). KL-Divergences of Normal, Gamma, Dirichlet and Wishart Densities. Technical report Wellcome Dpt of Cognitive Neurology, University College London.

[Pohle, 2005] Pohle, T. (2005). Extraction of Audio Descriptors and Their Evaluation in Music Classification Tasks. Master's thesis Diploma thesis, TU Kaiserslautern, OFAI, DFKI.

[Pohle et al., 2007] Pohle, T., Knees, P., Schedl, M., Pampalk, E. and Widmer, G. (2007). "Reinventing the Wheel": A Novel Approach to Music Player Interfaces. IEEE Transactions on Multimedia *9(3)*, 567–575.

[Pohle et al., 2006a] Pohle, T., Knees, P., Schedl, M. and Widmer, G. (2006a). Independent Component Analysis for Music Similarity Computation. In Proceedings of the 7^{th} International Conference on Music Information Retrieval (ISMIR'06).

[Pohle et al., 2006b] Pohle, T., Knees, P., Schedl, M. and Widmer, G. (2006b). Automatically Adapting the Structure of Audio Similarity Spaces. In Proceedings of the 1^{st} Workshop on Learning the Semantics of Audio Signals (LSAS 2006), 1^{st} International Conference on Semantics and Digital Media Technology (SAMT 2006).

[Pohle et al., 2007a] Pohle, T., Knees, P., Schedl, M. and Widmer, G. (2007a). Building an Interactive Next-Generation Artist Recommender Based on Automatically Derived High-Level Concepts. In Proceedings of the 5th International Workshop on Content Based Multimedia Indexing (CBMI 2007).

[Pohle et al., 2007b] Pohle, T., Knees, P., Schedl, M. and Widmer, G. (2007b). Meaningfully Browsing Music Services. In Proceedings of the 8^{th} International Conference on Music Information Retrieval (ISMIR'07).

[Pohle et al., 2005a] Pohle, T., Pampalk, E. and Widmer, G. (2005a). Evaluation of Frequently Used Audio Features for Classification of Music into Perceptual Categories. In Proceedings of the 4^{th} International Workshop on Content Based Multimedia Indexing (CBMI 2005).

[Pohle et al., 2005b] Pohle, T., Pampalk, E. and Widmer, G. (2005b). Generating Similarity-Based Playlists Using Traveling Salesman Algorithms. In Proceedings of the 8th International Conference on Digital Audio Effects (DAFx-05).

[Pohle and Schnitzer, 2007] Pohle, T. and Schnitzer, D. (2007). Striving for an Improved Audio Similarity Measure. In 4th Annual Music Information Retrieval Evaluation Exchange.

[Pohle et al., 2009] Pohle, T., Schnitzer, D., Schedl, M., Knees, P. and Widmer, G. (2009). On Rhythm and General Music Similarity. In Proceedings

of the 10^{th} International Conference on Music Information Retrieval (ISMIR'09).

[Pohle et al., 2008] Pohle, T., Seyerlehner, K. and Widmer, G. (2008). An Approach to Automatically Tracking Music Preference on Mobile Players. In Proceedings of the 6th International Workshop on Adaptive Multimedia Retrieval (AMR'2008).

[Rubner et al., 2000] Rubner, Y., Tomasi, C. and Guibas, L. J. (2000). The Earth Mover's Distance as a Metric for Image Retrieval. International Journal of Computer Vision *40*, 99–121.

[Salton and Buckley, 1988] Salton, G. and Buckley, C. (1988). Term-Weighting Approaches in Automatic Text Retrieval. Information Processing and Management *24*, 513–523.

[Sanderson and Zobel, 2005] Sanderson, M. and Zobel, J. (2005). Information Retrieval System Evaluation: Effort, Sensitivity, and Reliability. In Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2005).

[Schedl, 2008] Schedl, M. (2008). Automatically Extracting, Analyzing, and Visualizing Information on Music Artists from the World Wide Web. PhD thesis, Department of Computational Perception, Johannes Kepler University Linz, Austria.

[Schedl et al., 2007] Schedl, M., Knees, P., Seyerlehner, K. and Pohle, T. (2007). The CoMIRVA Toolkit for Visualizing Music-Related Data. In Proceedings of the 9th Eurographics/IEEE VGTC Symposium on Visualization (EuroVis'07).

[Schedl et al., 2006] Schedl, M., Pohle, T., Knees, P. and Widmer, G. (2006). Assigning and Visualizing Music Genres by Web-based Co-occurrence Analysis. In Proceedings of the 7^{th} International Conference on Music Information Retrieval (ISMIR'06).

[Schedl et al., 2007] Schedl, M., Widmer, G., Pohle, T. and Seyerlehner, K. (2007). Web-Based Detection of Music Band Members and Line-Up. In Proceedings of the 8^{th} International Conference on Music Information Retrieval (ISMIR'07).

[Scheirer, 1998] Scheirer, E. D. (1998). Tempo and Beat Analysis of Acoustic Musical Signals. The Journal of the Acoustical Society of America *103(1)*, 588–601.

[Schnitzer et al., 2007] Schnitzer, D., Pohle, T., Knees, P. and Widmer, G. (2007). One-Touch Access to Music on Mobile Devices. In Proceedings of the 6^{th} International Conference on Mobile and Ubiquitous Multimedia (MUM 2007).

[Seyerlehner and Schnitzer, 2007] Seyerlehner, K. and Schnitzer, D. (2007). From Rhythm Patterns to Perceived Tempo. In Proceedings of the 8^{th} International Conference on Music Information Retrieval (ISMIR'07).

[Seyerlehner et al., 2008] Seyerlehner, K., Widmer, G. and Knees, P. (2008). Frame Level Audio Similarity - A Codebook Approach. In Proceedings of the 11th International Conference on Digital Audio Effects (DAFx-08).

[Shi and Malik, 2000] Shi, J. and Malik, J. (2000). Normalized Cuts and Image Segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence *22(8)*, 888–905.

[Smaragdis, 2001] Smaragdis, P. (2001). Redundancy Reduction for Computational Audition, a Unifying Approach. PhD thesis, Massachusetts Institute of Technology.

[Srivastava and Gupta, 2008] Srivastava, S. and Gupta, M. R. (2008). Bayesian Estimation of the Entropy of the Multivariate Gaussian. In Proceedings of the IEEE International Symposium on Information Theory (ISIT 2008).

[Stenzel and Kamps, 2005] Stenzel, R. and Kamps, T. (2005). Improving Content-Based Similarity Measures by Training a Collaborative Model. In Proceedings of the 6^{th} International Conference on Music Information Retrieval (ISMIR'05).

[Swain and Ballard, 1991] Swain, M. and Ballard, D. (1991). Color indexing. International Journal of Computer Vision *7*, 11–32.

[Torres et al., 2007] Torres, D., Turnbull, D., Barrington, L. and Lanckriet, G. (2007). Identifying Words that are Musically Meaningful. In Proceedings of the 8^{th} International Conference on Music Information Retrieval (ISMIR'07).

[Turnbull et al., 2006] Turnbull, D., Barrington, L. and Lanckriet, G. (2006). Modeling Music and Words Using a Multi-Class Naive Bayes Approach. In Proceedings of the 7^{th} International Conference on Music Information Retrieval (ISMIR'06).

[Turnbull et al., 2008] Turnbull, D., Barrington, L. and Lanckriet, G. (2008). Five Approaches to Collecting Tags for Music. In Proceedings of the 9^{th} International Conference on Music Information Retrieval (ISMIR'08).

[Turnbull et al., 2007] Turnbull, D., Liu, R., Barrington, L. and Lanckriet, G. (2007). A Game-Based Approach for Collecting Semantic Annotations of Music. In Proceedings of the 8^{th} International Conference on Music Information Retrieval (ISMIR'07).

[Tzanetakis and Cook, 1999a] Tzanetakis, G. and Cook, P. (1999a). MARSYAS: A Framework for Audio Analysis. Organized Sound *4(3)*, 169–175.

[Tzanetakis and Cook, 1999b] Tzanetakis, G. and Cook, P. (1999b). Multifeature Audio Segmentation for Browsing and Annotation. In Proceedings of the 1999 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA99).

[Tzanetakis and Cook, 2002] Tzanetakis, G. and Cook, P. (2002). Musical Genre Classification of Audio Signals. IEEE Transactions on Speech and Audio Processing *10*, 293–302.

[Tzanetakis et al., 2001] Tzanetakis, G., Essl, G. and Cook, P. (2001). Automatic Musical Genre Classification Of Audio Signals. In Proceedings of the 2^{nd} International Conference on Music Information Retrieval (ISMIR'01).

[Tzanetakis et al., 2007] Tzanetakis, G., Jones, R. and McNally, K. (2007). Stereo Panning Features for Classifying Recording Production Style. In Proceedings of the 8^{th} International Conference on Music Information Retrieval (ISMIR'07).

[West, 2008] West, K. (2008). Novel Techniques for Audio Music Classification and Search. PhD thesis, School of Computing Sciences, University of East Anglia.

[West and Cox, 2005] West, K. and Cox, S. (2005). Finding an Optimal Segmentation for Audio Genre Classification. In Proceedings of the 6^{th} International Conference on Music Information Retrieval (ISMIR'05).

[West and Lamere, 2007] West, K. and Lamere, P. (2007). A Model-Based Approach to Constructing Music Similarity Functions. EURASIP Journal on Applied Signal Processing *2007*, 149–149.

[Whitman, 2005] Whitman, B. (2005). Learning the Meaning of Music. PhD thesis, School of Architecture and Planning, Massachusetts Institute of Technology.

[Whitman and Ellis, 2004] Whitman, B. and Ellis, D. P. W. (2004). Automatic Record Reviews. In Proceedings of the 5^{th} International Conference on Music Information Retrieval (ISMIR'04).

[Whitman and Lawrence, 2002] Whitman, B. and Lawrence, S. (2002). Inferring Descriptions and Similarity for Music from Community Metadata. In Proceedings of the 2002 International Computer Music Conference (ICMC).

[Whitman and Smaragdis, 2002] Whitman, B. and Smaragdis, P. (2002). Combining Musical and Cultural Features for Intelligent Style Detection. In Proceedings of the 3^{rd} International Conference on Music Information Retrieval (ISMIR'02).

[Xu et al., 1998] Xu, W., Duchateau, J., Demuynck, K. and Dologlou, I. (1998). A New Approach to Merging Gaussian Densities in Large Vocabulary Continuous Speech Recognition. In Proceedings of the IEEE Benelux Signal Processing Symposium.

[Xu et al., 2003] Xu, W., Liu, X. and Gong, Y. (2003). Document Clustering Based On Non-negative Matrix Factorization. In Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval.

[Yu and Slotine, 2009] Yu, G. and Slotine, J.-J. E. (2009). Audio Classification from Time-Frequency Texture. In Proceedings of the 2009 IEEE International Conference on Acoustics, Speech and Signal Processing.

[Zobel and Moffat, 1998] Zobel, J. and Moffat, A. (1998). Exploring the Similarity Space. ACM SIGIR Forum *32*, 18–34.