Content-Based Music Recommender Systems: Beyond simple Frame-Level Audio Similarity

Dissertation

zur Erlangung des akademischen Grades

Doktor

im Doktoratsstudium der

Technischen Wissenschaften

Eingereicht von:
Dipl.-Ing. Klaus Seyerlehner

Angefertigt am:
Institut für Computational Perception

Beurteilung:
Univ.-Prof. Dipl.-Ing. Dr. Gerhard Widmer (Betreuung)
Dr.-Eng. Geoffroy Peeters

Linz, Dezember 2010

Acknowledgments

First of all I want to thank Gerhard Widmer. He was the one to organize all those research projects and allowed me to participate in many interesting conferences all over the world, where I got to know many interesting people. He also gave me the possibility to contribute to very interesting industrial projects and as the supervisor of my thesis he gave me the opportunity and freedom to focus on research subjects based on my own interests.

Furthermore, I want to thank all my colleagues of our research team. Most of the time we had a perfect friendly, and also a very productive atmosphere, which made working there a pleasure. My special thanks go to Peter Knees, Markus Schedl, Dominik Schnitzer and Tim Pohle for all those fruitful discussions and their collaboration.

Further thanks go to Claudia Kindermann for all those relaxing coffee breaks and especially for her famous grill events, to Josef Scharinger for his friendship and the great fun we had during our weekly after work session on Friday and to Nora Chelbat from the Bioinformatics Institute next door for all the liveliness she brought to the 7th floor and her unrestricted openness. Furthermore, I want to thank my colleagues at OFAI (The Austrian Research Institute for Artificial Intelligence), especially Arthur Flexer and Martin Gasser, for many productive as well as jolly meetings and many interesting discussions and their collaboration.

From the economical side I want to thank the FWF (Fonds zur Förderung der wissenschaftlichen Forschung) for funding my research work.

Finally, I want to thank my whole family: my parents and my brothers, my both grandmothers, my little sister and of course my lovely girlfriend, for always being there for me and for supporting me.

Kurzfassung

Zielsetzung dieser Dissertation ist es, inhaltsbasierte Musikempfehlungssysteme zu verbessern. Neben einer allgemeinen Einführung in das Thema Musikempfehlungs-syssteme und einer intensiven Diskussion von Evaluierungsmethoden für inhaltsbasierte Musikempfehlungssysteme werden Verbesserungen auf zwei verschiedenen Abstraktionsebenen in dieser Dissertation behandelt:

Der erste offensichtliche Weg inhaltsbasierte Musikempfehlungssysteme zu verbessern ist das zugrundeliegenden Musik-Ähnlichkeitsmaß zu verbessern. Daher werden aktuelle Methoden Ähnlichkeiten zwischen Musikstücken abzuschätzen analysiert und Verbesserungsmöglichkeiten und Einschränkungen aufgezeigt. Dann wird ein neuartiges Framework vorgestellt, das im Gegensatz zu aktuellen Methoden das Spektrum nicht in einzelne Frames zerlegt verarbeitet sondern ganze Blöcke von Frames betrachtet. Für dieses Verarbeitungsframework werden mehrere neuartige Features definiert, die als Block-Features bezeichnet werden. Um auf Basis dieser neuen Features Musikempfehlungen generieren zu können, werden zwei Ansätze vorgestellt. Der eine Ansatz ist ein Musikähnlichkeitsmaß direkt auf Basis dieser Block-Features zu definieren, der andere zuerst eine Menge beschreibender Schlagwörter vorherzusagen, um dann auf deren Basis paarweise Ähnlichkeiten zwischen Musikstücken abzuschätzen. Schlußendlich wird vorgeschlagen beide Ansätze zu kombinieren. Dieser Kombinationsansatz belegte den ersten Rang im heurigen Audio-Musikähnlichkeitswettbewerb (MIREX 2010).

Der zweite Ansatz, inhaltsbasierte Musikempfehlungssysteme zu verbessern, betrachtet Empfehlungssysteme auf einer abstrakteren Ebene. Auf dieser Abstraktionsebene wird ein Musikempfehlungssystem als ein Empfehlungsnetzwerk gesehen. Eine Analyse dieser Empfehlungsnetzwerke zeigt, dass der einfache Ansatz, die N ähnlichsten Musiktitel zu empfehlen, die Erreichbarkeit von Musiktiteln in diesem Netzwerk stark vermindern kann. Dies kann den Nutzen von Musikempfehlungssystemen deutlich reduzieren. Um dieses Problem zu beheben, werden zwei Strategien vorgestellt und untersucht. Es zeigt sich, dass beide Methoden die Erreichbarkeit von Musiktiteln innerhalb eines Empfehlungsnetzwerkes signifikant verbessern können.

Abstract

This thesis aims at improving content-based music recommender systems. Besides a general introduction to music recommendation and an in-depth discussion of evaluation methods of content-based music recommender systems, improvements on two different abstraction levels are considered in this thesis:

The first and most obvious way to improve a content-based music recommender system is to improve the underlying music similarity measure that is used to generate the recommendations. State-of-the-art frame-level audio similarity algorithms are analyzed and improvements and limitations are discussed. Then a novel block-level feature extraction framework and a set of novel block-level features are introduced. To generate recommendations based on these block-level features two approaches are presented. One approach is based on directly estimating similarities based on the block-level features, and the other approach performs a mapping onto a semantic tag space before estimating pairwise song similarities. Finally, it is proposed to combine these two approaches. This combination approach ranked first in the MIREX 2010 Audio Music Similarity and Retrieval task.

The second way of improving content-based music recommenders considers a more abstract level of a music recommendation system. At this higher level of abstraction a music recommender system is interpreted as a recommendation network. Based on the analysis of the emerging recommendation network it will be shown that a straightforward top-N recommendation approach can significantly impact the reachability of songs in a music recommendation network. This can radically reduce the usability of music recommender systems. Two strategies to alleviate this issue are presented and evaluated. It will be shown that both strategies can significantly improve the reachability of songs within a music recommendation network.

Contents

1 Introduction

1.1 Motivation and Vision

While just a few years ago people were mainly looking for and buying music from local CD stores, the way people search for music and the way people consume music has radically changed. Thanks to portable music players like the *iPod* and mobile Internet access music can be listened to everywhere and at any time. It is not only the way people consume music that has completely changed, but also the variety of songs that is available. Today even the most obscure type of music is available for download from modern music online platforms. Thus, the problem is no longer to store or transmit digital music, but to assist users and help them find music they like. Assisting users in their music search is the task of a **music recommender system**.

Many of the popular music services today have a music recommender system attached. However, the majority of these systems is based on conventional collaborative filtering algorithms that analyze for example user ratings. This thesis, in contrast, focuses on another well-known approach in Music Information Retrieval (MIR), namely **content-based music recommendation**. The idea of a content-based music recommender systems is that the machine itself can *"listen"* and understand music on its own. The ultimate goal would be that the machine can then act as a musical expert and assist the user in her search task and point her to new interesting songs like a good friend would do. Furthermore, the machine could then help to automatically organize and visualize a music collection according to the user's taste or mood. From a technical point of view such content-based music recommender systems significantly differ from other recommendation techniques. Instead of accumulating e.g. user ratings or information from the web, they directly analyze the music audio signals and generate recommendations based on the information extracted from the signals themselves.

While content-based music recommenders have been a hot topic in academia for some years now, recently the first commercial applications and products popped up on the market. One of the first companies that introduced a real commercial product were *Bang & Olufsen* who presented the *BeoSound5*, the first commercial home entertainment center with automatic playlist generation based on content analysis inside. Many start-up companies working on content-based recommendation like for example *The Echo Nest*[1], *mufin*[2] or *sonarflow*[3] have been founded. Another example for the growing interest in content-based techniques is the *FM4 Soundpark*[4], an online music platform of an Austrian radio station. The main drawback of content-based music recommender systems is that these systems still lack in quality. This essentially means that the machine is recommending items that according to humans are not appropriate recommendations in a given context.

The **goal of this thesis** is to further improve the usability of content-based music recommenders. Improvements on two different abstraction levels are considered in this thesis:

- The first and most obvious way to improve a content-based music recommender system is to improve the underlying music similarity measure that is used to generate the recommendations. State-of-the-art frame-level audio similarity algorithms are analyzed and improvements and limitations are discussed. Then a novel block-level feature extraction framework and a set of novel block-level features is introduced. To generate recommendations based on these block-level features two approaches are presented. One approach is based on directly estimating similarities based on the block-level features, and the other approach performs a mapping onto a semantic tag space before estimating pairwise song similarities. Finally, it is proposed to combine

[1]http://the.echonest.com/
[2]http://www.mufin.com/
[3]http://www.sonarflow.at/
[4]http://fm4.orf.at/soundpark/

these two approaches.

- The second approach considers a more abstract level of a music recommendation system. At this higher level of abstraction a music recommender system is interpreted as a recommendation network. Based on the analysis of the emerging recommendation network it will be shown that a straight forward top-N recommendation approach can significantly impact the usability of music recommendation systems. Two strategies to alleviate this issue are presented and evaluated.

The next section presents the outline of this thesis and summarizes the major contributions presented in this thesis.

1.2 Organization & Contributions

This thesis is organized in seven chapters. **Chapter 1** contains an introduction to the field of music recommendation by giving an overview on basic recommendation approaches and introducing some fundamental terms that will be used throughout this thesis. Furthermore, the important relation between *music similarity* and *music recommendation* is discussed.

Then in **Chapter 2** basic digital signal processing techniques are introduced. This brief introduction specifically focuses on music signal processing techniques that are then used in the descriptions of the music processing algorithms in the consecutive chapters.

Chapter 3 of this thesis focuses on evaluation methods and metrics for music recommender systems. First, direct evaluation methods of assessing the quality of music recommender systems are discussed. Then the evaluation via genre classification, which is a common workaround to evaluate content-based systems, is presented. Genre classification is a practical and cost efficient evaluation method, which makes this evaluation method especially suitable for academic prototyping. However, in the context of music recommendation, genre-classification-based quality measures can be extremely over-optimistic because of *album*, *artist* and *portfolio* *effects*. The influence and impact of artist and album effects on the evaluation measures that are used in this thesis are investigated. The conducted experiments show that the application of an *artist filter* to remove album and artist effects is a must in the context of music recommendation. Furthermore, artist and album effects not only require to carefully design genre classification based evaluation methods, but also require to compile specific evaluation collections containing songs by many different artists. Therefore, one further contribution of this thesis to the evaluation of music recommender systems are three genre classification datasets that are made publicly available. Some other contributions to the evaluation of music recommender systems via genre classification are based on the results of a conducted listening experiment. In this listening experiment humans were asked to classify a set of songs according to a predefined genre taxonomy. The collected data is then used to make the automatic classification results comparable to human classification results, which is typically not possible on standard datasets as for these datasets typically no human reference classification results have been collected. Additionally, automatic and human classification results are not only compared to each other but for the first time are also compared to two collaborative-filtering based algorithms (see 1.3). The results from this comparison indicate that the collaborative approach outperforms both the automatic approach and individual humans. This finding strongly supports the idea of hybrid music recommender systems combining a content-based and a collaborative-filtering based algorithm.

Chapter 4 is the second of the four main chapters and presents the author's contributions to the analysis and the improvement of *frame-level audio similarity algorithms*. This chapter starts with an introduction to frame-level algorithms, also often called Bag-Of-Frames (BoF) approaches, and gives an overview of different variants. Then two novel variants to compute frame-level similarities are presented. These two novel variants are used to analyze two specific aspects of the BoF approach. Using the first variant, called Multi-Level Vector Quantization (MLVQ), it is investigated if the distribution model itself, which can be either *parametric* or *non-parametric*, does have an influence on the resulting similarity estimates. The second variant of the BoF approach that is based on Nearest Neighbour Density Estimation (NNDE) will then be used to identify exactly those audio frames that make two songs sound the same from a machine's point of view. The insights gained from the analysis of the NNDE approach resulted in a novel frame selection strategy that improves the quality of frame-level similarity algorithms. Still, the main conclusion of this chapter is that frame-level similarity algorithms can only identify simple timbral song relations, but are difficult to extend or improve and are rather limited in identifying interesting musical similarities as needed for music recommender systems.

One possible limiting factor of frame-level audio similarity algorithms is that they are not able to capture *temporal information*, neither short term temporal information like the spectral evolution of a played piano tone, nor long term temporal information like rhythm or beat. To overcome these limiting factors a novel *block-processing framework* is proposed which will be presented in **Chapter 5**. This block-processing framework is inspired by the feature extraction process of the so-called *Fluctuation Patterns* proposed by Pampalk et al. [Pampalk et al., 2002]. A set of six novel block-level features has been developed that can be extracted within the proposed block-processing framework. Because of its vector space representation the proposed feature set is not only useful to visualize a song's model, but can also be used for music classification in a straight-forward way. The submitted algorithm based on the block-level feature set ranked first in the *MIREX 2010 Audio Classification (Train/Test) Tasks*, which is a clear indicator for the high descriptive power of the proposed feature set. Furthermore, based on these block-level features two novel *audio similarity measures* are defined. The first approach directly combines the distance estimates of the individual block-level features into an overall similarity measure. The second approach performs a mapping over a semantic space by predicting tag affinities using the extracted block-level features. Both approaches are evaluated and compared to state-of-the-art algorithms. Furthermore, numerous combinations of state-of-the-art approaches are evaluated to identify any potential for further improvements. One of these combinations, fusing the block-level similarity and the tag-based similarity measure, was submitted to the *MIREX 2010 Audio Music Similarity and Retrieval Task* and ranked first in this competition. Thus, the proposed algorithms can be considered to be state-of-the-art in content-based music recommendation.

Chapter 6 then focuses on improvements of music recommender systems on a more abstract level by analyzing the emerging music recommendation network. Although music recommender systems are often evaluated based on standard quality measures like music genre classification accuracies only, recent research work is more and more focusing on the analysis of the resulting recommendation networks [Celma and Cano, 2008, Celma and Herrera, 2008]. This thesis contributes to this research direction by analyzing music recommendation networks with respect to their navigability. More precisely, it is shown that *hubs* (highly reachable graph nodes) and *sources* (not reachable graph nodes) naturally exist in any top-N recommendation network. Unfortunately, both hubs and sources can decrease the usability of recommendation networks. Furthermore, it is empirically shown for two real world datasets (a music recommendation network and a movie recommendation network) that the number of sources (the number of items that are not reachable via browsing the recommendations) can be a serious problem within recommendation networks as it can be a non-negligible portion of all items. Finally, two approaches to improve the browsability of recommendation networks are presented and evaluated.

In **Chapter 7** the author's conclusions on the obtained results are presented and an outlook on future research directions for music recommender systems is given.

The next section starts with a brief introduction to the field of music recommendation.

1.3 Music Recommenders: Types of Systems and Focus of Thesis

Today recommendation is a hot and widespread topic in various application areas and not restriced to music at all. Some very well-known recommender systems are those used at *Netflix.com*[5] to recommend movies and the recommender system

[5]http://www.netflix.com/

used at *Amazon.com*[6] to recommend a large variety of products. There also exist some not that well-known application areas of recommender systems like news recommendation or blog recommendation. And the number of services that make use of recommendation engines is steadily growing.

However, independent of the application area and the products that are recommended, the basic *recommendation scenario* [Sarwar et al., 2001, Celma, 2008] is in principle the same. Recommending basically means to predict a subset I_{u_a} of items out of all items $I = \{i_1, ..., i_m\}$ that are available through a service such that the items in I_{u_a} are **interesting** for a specific *active user* $u_a \in U$ out of all users $U = \{u_1, ..., u_n\}$. Furthermore, there exists some knowledge about the user u_i's interest in form of a list of items I_{u_i}, which for example have been rated by the user. In this thesis we focus on a special recommendation scenario, where the user is currently focusing on a specific item i_a out of all items, the *active item*. The active item is typically the current song the user is listening to. Naturally, the number of recommended items $|I_{u_a}|$ should be far smaller than the total number of items m. For practical reasons (e.g. the display space) the number of recommended items is often restricted to contain at most N items. In this case one speaks of *top-N recommendation* [Sarwar et al., 2001, Karypis, 2001, Lam. and Riedl, 2004]. Based on this formal definition of a recommendation scenario the important question is how to generate recommendations.

With respect to music recommender systems one can identify three principal recommendation approaches [Barrington et al., 2009, Celma, 2008]: the *Metadata-based Approach*, the *Content-based Approach* and *Hybrid Approaches*. Metadata-based approaches can be further subdivided according to the type of metadata that is used and the way it is collected. Figure 1.1 visualizes a categorization of music recommender systems.[7] In the following these types of recommender systems are briefly reviewed and the relation of this thesis to the different types of systems is

[6]http://www.amazon.com/

[7]It is worth mentioning that there exist three further general recommendation approaches (the Demographic Filtering-based Approach, Utility-based Approach and Knowledge-based Approach)[Burke, 2002], but in the context of music recommendation these approaches have not yet received any attention so far and are therefore not considered in this thesis.

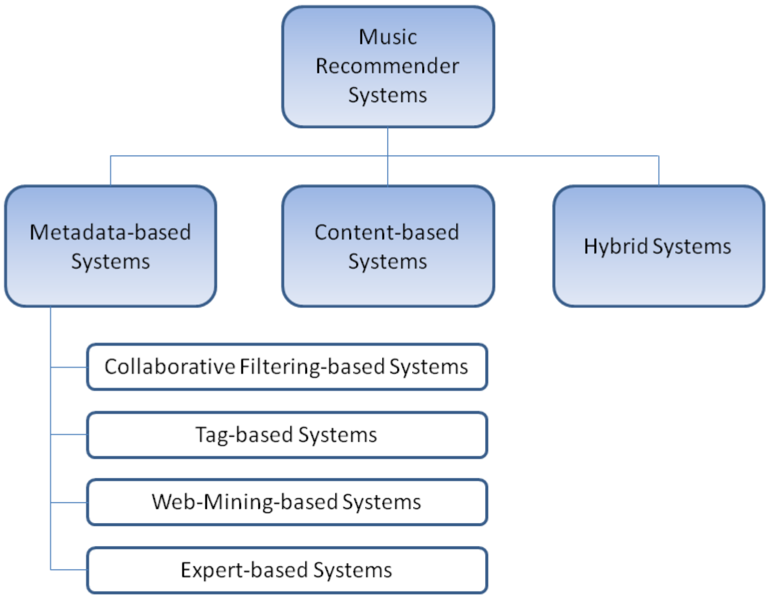


Figure 1.1: Categorization of music recommender systems.

emphasized.

1.3.1 Metadata-based Approaches

Metadata-based approaches use information *about* music that is not directly derivable from the audio signal itself, but has been associated with a specific song by a human subject. This can be any *explicit metadata* like genre, tags, artist, song names or user ratings or any kind of *implicit metadata* like purchase information, play counts or skip rates. All metadata-based approaches are *collaborative approaches* as no single person can ever annotate (neither implicitly nor explicitly) the complete music universe. Thus, in some publications this category is denoted collaborative approaches instead of metadata-based approaches [Adomavicius and Tuzhilin, 2005]. Here we further distinguish four types of metadata-based approaches, namely *Collaborative Filtering-based Approaches*, *Web-Mining-based Approaches*, *Tag-based Approaches* and *Expert-based Approaches*. Metadata-based approaches are not in the main focus of this thesis, but in chapter 3, which compares the genre classification performance of automatic and human classification, also two collaborative filtering based systems are evaluated, to cover a broader range of recommendation approaches (see section 3.4.3.3). Furthermore, in **??** a music recommendation system based on automatically estimated tags is presented. Thus, the research work presented in this thesis also has a weak relation to tag-based music recommendation systems. The following subsections briefly introduce the four subtypes of metadata-based systems. Wherever commercial solutions exist in the area of music, one of these is also briefly discussed.

1.3.1.1 Collaborative Filtering Approaches

The collaborative filtering approach has already been intensively studied in academic research and has been successfully used in many commercial recommendation applications. A classical collaborative filtering system allows users to rate items e.g. on a five star scale. For example many modern music players like Apple's *iTunes* or the *Microsoft Media Player* allow to collect explicit ratings typically on a five star rating scale. Besides *explicit* ratings, many of these systems also collect a lot of *implicit* user feedbacks based on the user's actions. For example stopping or skipping songs could indicate that a user dislikes a specific song. Furthermore, one typically distinguishes two different variants of collaborative filtering: *item-based* and *user-based* systems [Wang et al., 2006, Sarwar et al., 2001, Celma, 2008]. User-based collaborative filtering systems first estimate similarities among users based on the collected ratings and then recommend items that have been bought by similar users. In item-based collaborative filtering first similarities among items are estimated. Then items that are similar to items the user has been interested in are recommended.

One well-known example of a real world music recommender system that is most likely based on collaborative filtering is Apple's Genius. Although the details of the music recommendation engine are kept secret, according to Barrington et al.

[Barrington et al., 2009] some informal experiments give some clues that Apple is using collaborative filtering to compare the seed song's metadata to iTunes' massive database of music sales (over 50 million customers who have purchased over 5 billion songs). Unfortunately, it is unknown if and how the Genius engine has affected Apple's music sales.

1.3.1.2 Tag-based Approaches

The tag-based approach is also known as collaborative tagging, social classification, social indexing, social tagging or Folksonomy. In tag-based systems users assign descriptive terms, so-called tags, to each song. All terms assigned to a specific song together form a song description. Songs or artists with similar descriptions are considered to be similar too. One example of such a popular music folksonomy is *last.fm*. One advantage of tag-based systems is that the visualizations of a song's tag model, so-called tag clouds, can be used to immediately explain to users why two artists are considered similar or related by the system. However, one disadvantage is that tags are often assigned at the artist level. At the track level data is very sparse and tags are often not available. Additionally, there exist a number of issues related to the tags themselves. For example synonyms like *"Hip-Hop"* and *"HipHop"* and personal tags like *"My Favorite"* pose a problem.

1.3.1.3 Web-Mining-based Approaches

Web-mining-based approaches are quite related to tag-based approaches. In contrast to tag-based systems, where the tags are explicitly assigned by users, web-mining based systems rely on descriptive terms extracted from web pages. Such systems are based on the traditional methods of information retrieval. Typically either term profiles are extracted from a set of web pages, or simple page counts or web co-occurrences are used [Schedl and Knees, 2009]. Another important difference to tag-based systems, where users can potentially use any term to tag a song, is that web-mining based systems often filter the document terms with a controlled dictionary of musical terms. However, web-mining-based systems are often not reliable and can typically not provide recommendations at the song level. With respect to music recommendation, web-based systems have so far only been used in academic research.

1.3.1.4 Expert-based Approaches

Expert-based systems do not collect information about music from a crowd of users, but rather rely on explicit metadata assigned by musical experts. Expert-based approaches are special in the sense that only trustworthy and well-trained experts collaborate in the annotation process. Of course manually annotating songs is costly and does not scale well. Furthermore, although the musical experts are typically well-trained there can still be inconsistencies in the annotations. Probably the only music service that is based on meta information assigned by music experts is *Pandora*[8]. The music experts at Pandora manually quantify properties of songs that are related to melody, harmony and rhythm, to instrumentation, orchestration, arrangement, lyrics, singing and vocal harmony. Over 400 different musical attributes are quantified for each song. Based on this information the Pandora Music Service then generates personalized Internet Radio Streams and offers the ability to buy the songs or albums at various online retailers. Since 2008 Pandora is not only available on PC platforms but also on many mobile devices. Currently Pandora has 700,000 tracks in its library and 48 million users who listen for 11.6 hours per month on average. While Pandora was already founded back in 2000, the company reported its first profitable quarter at the end of 2009 with about $50 million in annual revenue. This is another indicator for the growing interest in music recommendation services.

The expert-based approach is the last of the four important types of metadata-based approaches. The next section now presents a fundamentally different category of approaches: the *content-based approach*.

[8]http://www.pandora.com/

1.3.2 Content-based Recommendation Approaches

In contrast to metadata-based approaches, which are based on user created metadata, content-based approaches instead try to analyze the content of digital objects directly. The content-based approaches are basically restricted to digital objects like images, videos, music or e-books. The basic approach is to extract meaningful features out of the digital objects and to generate comprehensive object representations from these. Thus, there exists a functional or mathematical dependency between the content of a digital object and its representation (see figure 1.2).

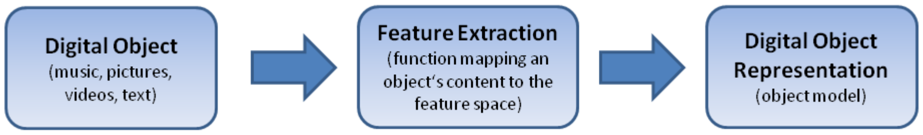

Digital Object
(music, pictures, videos, text)
Feature Extraction
(function mapping an object's content to the feature space)
Digital Object Representation
(object model)

Figure 1.2: Feature Extraction process of content-based recommender systems.

Content-based recommendation approaches belong to the category of item-based recommendation approaches, where recommendations are generated based on item-to-item similarities. Content-based recommender systems estimate similarities among items by matching the extracted object representations. Hence, each content-based recommendation algorithm basically has to perform two steps:

- First, extract features out of the content of a digital object and combine these features into a meaningful object representation.
- Second, define a similarity function among the object representations that corresponds to the human perception of the item-to-item similarity.

For many domains like video, text, images and audio, how to define a set of appropriate features and the corresponding similarity function is still an open research question. Within this thesis the definition of feature sets and similarity functions for digital music signals are investigated. Particularly, this thesis will concentrate on the analysis of standard music similarity algorithms and on the definition of novel meaningful song representations and the corresponding similarity functions (in Chapters 4 and 5).

The clear advantage of content-based methods is that they do not require any metadata, they are perfectly scalable and can generate recommendations for completely unknown and unrated songs. Furthermore, the generated recommendations are to some extent *"objective"* and are consequently not biased towards well-known or popular items. However, the content-based approach also has some major drawbacks. For example content-based approaches are not able to capture any contextual information or cultural information. Only information that can be derived from the content of a digital object can be extracted and used to generate recommendations. In practice, however, it is often hard to extract and adequately map information that is inside the content of a digital object onto a perceptually meaningful descriptor. Thus, the main problem of content-based techniques is that they are often not satisfactory with respect to recommendation quality. Improving the recommendation quality of content-based music recommender systems is the main goal of this thesis.

Some of the research results of this thesis have already been used in a commercial environment. One example for the application of the results of this thesis is the frame-selection strategy proposed in section 4.3.3. This strategy is used in *Bang & Olufsen*'s *BeoSound5*, the first commercial available home entertainment center with automatic playlist generation inside. Another example is the *Continuous Frequency Activation* (CFA) feature (see section 5.2.5), which has been used to implement a speech-music discrimination system for a large Chinese telecom solutions provider. Furthermore, the analysis of the recommendation network of the *FM4 Soundpark* (see section 6.2.2) helped to identify weaknesses related to the browsability of the recommendations and so contributed to the improvement of the recommender system that is attached to the *FM4 Soundpark*. These examples indicate that content-based techniques are already mature enough in some areas to be used in commercial applications. Especially with respect to content-based music recommender systems is seems that such systems will get another boost in the context of *hybrid music recommender systems*. Hybrid recommender systems are typically combinations of content-based and metadata-based systems (see section 1.3.4) that in combination solve many of the problems of individual recommendation approaches. Some of these well-known problems will be discussed in the next section.

1.3.3 Well-Known Problems of Recommender Systems

This chapter presents some well-known weaknesses of many recommendation approaches. Although some of these issues are frequently discussed in the context of collaborative-filtering-based systems only, most of them also apply to other recommendation approaches presented so far. In the following the most prominent issues are presented and then in table 1.1 it is indicated which recommendation approach is affected by the aforementioned problems.

- **The Sparsity Problem (SP)**
  Recommender systems often return poor recommendations, when they do not have much meta information on either the items or the users [Burke, 2002, Claypool et al., 1999]. In the context of music recommendation, for example, there typically exists little meta-information for unknown artists. Another problem is that even if metadata is available, it is often only available at the artist level and not at the song level.

- **The Cold-Start Problem (CSP)**
  The *Cold-Start Problem* can be interpreted as a special case of the *Sparsity Problem* [Schein et al., 2002]. For newly added items or new users that recently joined an online platform, recommendations are hard to generate because the system has no meta information at all about the new user or the new item.

- **The Gray-Sheep Problem (GSP)**
  For individuals with a significantly different opinion from the group opinion most recommender systems cannot generate adequate recommendations, simply because they do not consistently agree or disagree with any group of people [Claypool et al., 1999]. Such individuals will rarely, if ever, receive accurate predictions.

- **The Popularity-Bias Problem (PBP)**
  Many recommender systems, especially metadata-based systems, tend to have a popularity bias, which means that popular items are recommended very frequently, while unpopular or unknown items are not recommended at all [Celma and Cano, 2008] and consequently stay hidden in the so-called *long tail* [Brynjolfsson et al., 2003, Anderson, 2006, Brynjolfsson et al., 2006].

- **No Explanations Problem (NEP)**
  Another drawback of many recommendation approaches is that they follow a black box paradigm. Such systems provide recommendations, but do not provide user understandable explanations for the generated recommendations. Explanations help to make recommendations *transparent* and *scrutable* for the user [Kay et al., 2001, Barneveld and Setten, 2004]. Furthermore, explanations can be beneficial to help users understand and accept recommendation errors, but can also help alleviate fears related to perfect or too personal recommendation. For example many users feel observed (*"how could the machine know that"*), which can reduce the user's trust in the system [Tintarev and Masthoff, 2007].

- **Portfolio Effect (PE)**
  The term *portfolio effect* [Burke, 2002] describes the undesired situation, where identical or near identical items are recommended. For example the portfolio effect [Tintarev and Masthoff, 2006] is well-known in news recommendation, but also huge recommendation engines like *Amazon.com* have to deal with this problem. For instance, customers that have purchased many books by a specific author may get recommendation lists where all top-5 entries are books by that author [Ziegler et al., 2005]. Such recommendation sets clearly provide marginal utility for a user. With respect to music especially content-based recommender systems suffer from a related effect, the *artist effect* (see 3.3.1). The artist effect describes recommendation situations where only songs by the same artist as the query song are recommended.

In addition to these well-known problems also the *scalability* and the *recommendation quality* are important criteria for recommender systems. A compact overview which of the presented recommendation approaches are affected by which issues, is given in table 1.1. *Scalability (SA)* and *Recommendation Quality (RQ)* have been rated by the author on a three level scale (1=good, 2=medium, 3=poor) and reflect the author's personal opinion.

Taking a look at this comparison it seems that expert-based systems are a good solution. However, in practice these systems are rarely used because of their poor scalability. Furthermore, one can observe that the gray sheep problem is a fundamental one and cannot be solved by any recommender system. For *"gray-sheep"* the only solution is personal recommendation. Another general issue is the portfolio effect. Only tag-based systems seem not to be much affected by the portfolio effect, which is indicated by the conducted experiments in section 5.5, where automatically estimated tags are used. In contrast to the gray-sheep problem the portfolio effect can be solved technically by using topic diversification techniques. One straight-forward solution to resolve some of the issues related to a specific recommendation approach (especially the cold-start problem and the sparsity problem) is to combine two or more recommendation approaches. Such combinations are then called *hybrid recommendation systems*.

Approach	**SP**	**CSP**	**GSP**	**PBP**	**NEP**	**PE**	**SA**	**RQ**
Collaborative Filtering	✗	✗	✗	✗	✗	✗	1	1
Tag-based	✗	✗	✗	✗	✓	✓	1	1
Web-Mining-based	✗	✗	✗	✗	✓	✗	1	3
Expert-based	✓	✓	✗	✓	✓	✗	3	1
Content-based	✓	✓	✗	✓	✗	✗	1	2

Table 1.1: Recommendation approaches and their issues (a check indicates that the approach is not affected by the specific problem, whereas a cross indicates that the approach is affected).

1.3.4 Hybrid Recommendation Approaches

The idea of hybrid recommender systems is to combine two or more recommendation techniques in order to gain better performance and simultaneously eliminate specific issues of the underlying recommendation approaches [Burke, 2002]. In the literature many, often rather complex, ways of combining recommender systems have been proposed. Burke distinguishes seven different combination approaches (Weighted, Switching, Mixed, Feature Combination, Cascade, Feature Augmentation and Meta-level).

In music recommendation, hybrid systems have not yet received much research attention. In the simplest case, a metadata-based approach and a content-based approach are combined to improve the recommendation quality and to simultaneously solve the cold-start, the sparsity or the popularity-bias problem of the metadata based approach. In [Knees et al., 2008, Knees, 2007] Knees et al. make use of content-based item similarities to infer tag models for songs where no tag models are available. Another simple *mixing* approach has been proposed in [Tiemann and Pauws, 2007]. More advanced combinations have been proposed in [Donaldson, 2007] and [Yoshii et al., 2008]. Although the results obtained via these first hybrid algorithms look promising, to the best of the author's knowledge there do not yet exist any commercial solutions that are based on a hybrid recommendation approach. Still, hybrid recommender systems seem to be the future in music recommendation, as they can alleviate the weaknesses of the individual approaches.

The last section of this chapter points out the relation of music similarity algorithms and music recommendation, because similarity-based recommendation will be the focus of this thesis.

1.4 From *Music Similarity* to *Music Recommendation*

The most widespread *application* of a content-based music similarity measure is that one can easily build a music recommendation engine simply by recommending the songs that sound most similar to a given query song according to the similarity measure. Thus, many publications focusing on *Music Similarity* implicitly assume that the actual application scenario will be *Music Recommendation.* Often these terms are even used interchangeably and some important differences between music similarity and music recommendation are left out.

The most important difference between *Music Similarity* and *Music Recommendation* is that not all songs that *"sound similar"* are also good recommendations. For example it is quite obvious that users would be annoyed if just songs by the same artist as the query song or by just one other artist appear in the recommendations. Unfortunately, this is often the case if the recommendations are generated with a content-based similarity measure: songs by one artist are of course often very similar to each other, because of the very specific style and the specific recording conditions. From an acoustic point of view the characteristic style of an artist is related to the instrumentation, the singing voice and the type of music being played, which are all in some way reflected in a good similarity measure. Thus, it is relatively easy to identify songs by one and the same artist using audio similarity algorithms. This effect is known as *artist effect.* In some cases even album-specific production effects are reflected in the spectral representation of songs, which is respectively called *album effect.* Artist and album effects are not a problem *per se*, but they do become one in the context of recommendation, as such recommendations provide only marginal utility for users and could be easily generated by searching for the artist's name[9]. A simple and straight-forward solution is to simply filter out songs from the same artist as the query song, which is called artist filtering (see 3.3.1).

[9]Interestingly, similar effects are also known from other recommendation domains like, for example, news recommendation, under the term *portfolio effect* (see 1.3.3).

Post-filtering the music recommendation lists can, however, significantly impact the typically accuracy-based quality criteria. Thus, the main differences between content-based music recommendation and music similarity is the way how such systems must be evaluated. Especially using an artist filter is a must with respect to the evaluation of music recommendation. This will also be experimentally shown in Chapter 3.3.1. This also raises the need for novel evaluation datasets that contain songs from a large number of different artists. Furthermore, in the context of recommendation it is no longer enough to just focus on accuracy-based evaluation criteria only, but the research has to focus on other aspects of the user experience. Other evaluation criteria beyond accuracy like topic diversification [Ziegler et al., 2005] and navigability of recommendation networks [Seyerlehner et al., 2009a] and, most of all, user satisfaction become important criteria that clearly differentiate content-based music recommendation from music similarity estimation algorithms.

2 Music Signal Processing Fundamentals

This chapter briefly introduces the fundamentals of digital audio signal processing with a special focus on music signal processing. These fundamentals are then used in Chapters 4 and 5 to compactly describe the feature extraction process of the presented algorithms. For a more detailed introduction to general digital audio processing the interested reader is referred to [Zölzer, 2002, Oppenheim et al., 2004].

2.1 Digital Audio Signals

An analog audio signal $x(t)$ can be transformed into a digital signal $x[n]$ representation by an analog-to-digital converter (ADC). An ADC *samples* the analog signal in equidistant time intervals of duration T_s, the *sampling period*, and *quantizes* each amplitude sample by approximating the continuous amplitude values by a relatively small set of discrete symbols of typically 16-bit integer values. The *sampling rate* $f_s = 1/T_s$ of the digital audio signal is the reciprocal of the sampling interval and is given in *Hertz* (Hz).

From a practical point of view with respect to Music Information Retrieval the music audio signals to be analyzed are all digital audio signals that are most frequently stored encoded for example as *mp3*-files. Therefore *decoding* and *resampling* of audio files plays an important role. Decoding is a necessary but time consuming step to reconstruct the raw digital audio signal in Pulse Code Modulation (PCM) from the compressed audio files. Resampling is necessary to ensure that all input signals of the feature extraction stage have the same sampling rate. Typically, a low sample rate like 11kHz or 22kHz is used in many MIR applications. This is a first step to reduce the enormous amount of audio data. Although there exist some audio features, for example the *Zero-Crossing Rate*, that can be computed from the time-domain representation of an audio signal, the majority of well-known audio features [Peeters, 2004] including those investigated in this thesis are derived from the frequency domain representation of an audio signal.

2.2 Spectrum Analysis

The spectrum of an audio signal describes the distribution of energy over the frequency range and can be generated via a *Fourier Transform* of the signal. For digital signals the discrete variant, the *Discrete Fourier Transform* (DFT), is used.

$$X[k] = \mathbf{DFT}(x[n])] = \sum_{n=0}^{N-1} x[n]e^{-j2\pi nk/N} \qquad k = 0, 1, \ldots, N-1. \tag{2.1}$$

The DFT can be computed for any discrete and finite signal $x[n]$ of length N and is based on the assumption that the finite signal $x[n]$ of length N is one period of a periodic signal $\tilde{x}[n]$ with a period of length N. The Fourier Coefficients $X[k]$ then correspond to the Fourier Coefficients $\tilde{X}[k]$ of the periodic signal $\tilde{x}[n]$. Together the N complex-valued Fourier Coefficients $X[k]$[1] form the discrete Fourier spectrum of the discrete signal $x[n]$.

As the counterpart to the DFT there also exists the *inverse Discrete Fourier Transform* (iDFT), which transforms a spectrum back from the discrete frequency domain to the discrete time domain.

$$x[n] = \mathbf{iDFT}(X[k]) = \frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{j2\pi nk/N} \qquad n = 0, 1, \ldots, N-1. \tag{2.2}$$

[1]The square brackets are used to differentiate the discrete spectrum from the continuous spectrum.

The iDFT can be useful to re-synthesize a spectrum e.g. in the case the spectrum has been modified by an algorithm to be able to listen to the result of the modification.

Based on the discrete Fourier spectrum one can compute the **magnitude spectrum** $|X[k]|$ and the **phase spectrum** $\varphi[k]$.

$$|X[k]| = \sqrt{\mathrm{Re}(X[k])^2 + \mathrm{Im}(X[k])^2} \tag{2.3}$$

$$\varphi[k] = \arctan \frac{\mathrm{Im}(X[k])}{\mathrm{Re}(X[k])} \tag{2.4}$$

Often logarithmic amplitude scale is used to plot the magnitude spectrum, which can be obtained according to equation 2.5.

$$X[k]_{dB} = 20 \log_{10} (|X[k]|) \tag{2.5}$$

The frequency resolution of the discrete Fourier transform is linear. Hence, the Fourier coefficients are equidistant samples of the continuous spectrum. Given that the signal $x(t)$ was sampled at a sampling rate f_s the continuous spectrum is sampled at the frequency points $k\frac{f_s}{N}$, where k is running from 0 to $N-1$. In practice the DFT is computed by the fast Fourier Transform (FFT), an efficient algorithmic implementation of the DFT and the Fourier Coefficients are often called *frequency bins* or *FFT bins*.

Figure 2.1 shows a real world audio signal $x[n]$ taken from a Wagner Opera and the magnitude spectrum thereof. In the lowest plot the frequency is given in Hertz. The audio signal was sampled at 22 kHz. The magnitude spectrum is mirrored around the *Nyquist frequency* (11 kHz in this example) as the input signal is a real-valued signal and the resulting Fourier Coefficients around the Nyquist frequency are conjugated. For audio signals this is always the case as they are always real-valued. In MIR the spectrum is therefore typically only considered up to the Nyquist frequency.

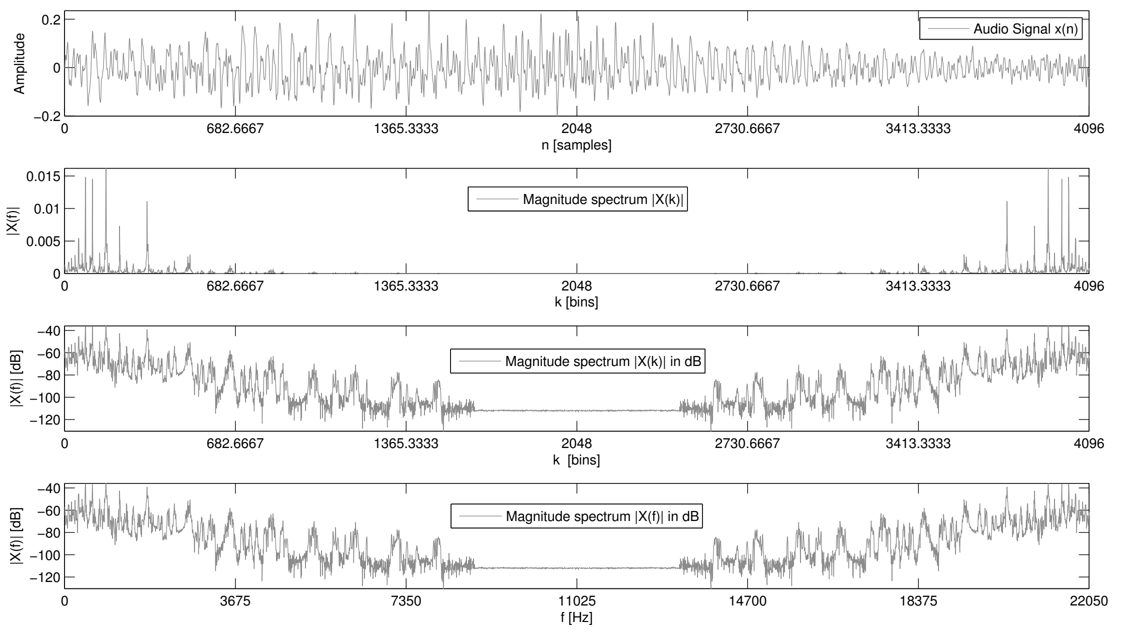


Figure 2.1: Visualization of an audio signal, the magnitude spectrum and the magnitude spectrum in dB.

With respect to MIR the linear frequency resolution of the DFT is not ideal, as a linear frequency resolution does not correspond to the human perception of sound. In the next section four acoustic scales are presented that better correspond to the logarithmic human frequency perception.

2.3 Auditory Scales

In many audio applications it can be useful to mimic the human listening process. One example is lossy audio compression. Such codecs make use of the limitations of the human auditory system by simply removing parts of an audio signal that are not perceived by humans anyway. In content-based MIR the analysis of audio signals should account for the human perception of sound so that just the perceptually relevant information of an audio signal is further processed. For this reason

MIR and Psychoacoustics, the study of subjective human perception of sound, are closely related research fields.

One aspect of the human perception of sound that is well-known is the fact that the frequency resolution of the human auditory system is not linear, but logarithmic. To account for the logarithmic human perception of sound in MIR, the linear frequency spectrum of the DFT is typically compressed according to an auditory scale. The most popular auditory scales that are used to perform this mapping from linear to logarithmic frequency resolution are the Mel-Scale, the Bark-Scale, the ERB-Scale and the Cent-Scale. For all four scales the relation between linear frequency scale (f_{Hz})and logarithmic frequency scale is defined by an equation, which is summarized in the following.

2.3.1 Mel-Scale

The Mel-Scale is a perceptual scale that has been defined in [Stanley et al., 1937]. The scale is based on the results of a pitch comparison listening experiment. The Mel-Scale is frequently used in MIR applications and also in many speech recognition applications, as the Mel-scaled spectrum is the basis to compute the widely used Mel Frequency Cepstral Coefficients (MFCCs), a compact representation of the spectral envelope of an audio frame (see 2.4.4).

$$f_{\mathrm{mel}} = 2595 \log_{10}\left(\frac{f_{\mathrm{Hz}}}{700} + 1\right) \tag{2.6}$$

2.3.2 Bark-Scale

The Bark-Scale is another psychoacoustic scale proposed by Eberhard Zwicker in 1961 [Zwicker, 1961]. He proposed to subdivide the audible frequency range into *critical bands.* The definition of critical bands is based on the observation that humans can no longer distinguish two tones, whenever their frequency difference is smaller than the *critical bandwidth.* In such a case the inability of the auditory frequency-analysis mechanism to resolve the two tones is linked to a beating or roughness sensation. Today often the ERB-Scale is used instead of the Bark-Scale.

$$f_{\text{bark}} = 13 \arctan(0.00076 f_{\text{Hz}}) + 3.5 \arctan((f_{\text{Hz}/7500})^2) \tag{2.7}$$

2.3.3 ERB-Scale

The human auditory system can be modeled as an array of overlapping band-pass filters known as *auditory filters*. Each auditory filter typically corresponds to a critical band. Using the equation 2.8 of the Equivalent Rectangular Bandwidth (ERB) one can estimate the bandwidth BW of an auditory filter at center frequency f_c in Hertz.[Moore, 1998].

$$BW_{\text{Hz}} = 24.7\,(0.00437 f_c + 1) \tag{2.8}$$

It is important to note that the formula above converts a frequency (in Hz) to a bandwidth (also in Hz). To convert a frequency in Hz to a frequency in units of ERB-bands, the following formula is used

$$f_{\text{ERB}} = 21.4 \log(0.00437 f_{\text{Hz}} + 1). \tag{2.9}$$

2.3.4 Cent-Scale

The Cent-Scale is a logarithmic scale for musical intervals e.g. the interval of a semitone corresponds to 100 cent and an octave corresponds to 1200 cent. Given the frequencies of two notes f_a and f_b the interval Δf_{cent} in cent is

$$\Delta f_{\text{cent}} = 1200 \log_2(\frac{f_a}{f_b}). \tag{2.10}$$

The relation of the linear frequency f_{Hz} to the frequency in cent f_{cent} is defined via a reference frequency. As reference frequency $f_b = 440 * (\sqrt[1200]{2})^{-5700})$ is used [Goto, 2003]. Consequently, the reference frequency is 5700 cent below the concert pitch of 440 Hz. The conversion from Hertz to Cent is defined by

$$f_{\mathrm{cent}} = 1200 \log_2 \left(f_{\mathrm{Hz}} / (440 * (\sqrt[1200]{2})^{-5700}) \right). \tag{2.11}$$

In contrast to Mel, Bark and ERB the Cent-Scale is not a psychoacoustic scale, but a musical scale. In the next section the presented auditory scales are compared.

2.3.5 Comparison & Mapping

To compare the different auditory scales the mapping-function from linear to logarithmic scale is plotted for a frequency range from 0 to 22050 Hertz (see figure 2.2). The different scales f_{cent}, f_{mel}, f_{bark} and f_{ERB} are all normalized by dividing them by the maximum value of each scale. One can see that all the auditory scales significantly differ from the linear scale and clearly exhibit a logarithmic characteristic. The scales can be sorted according to the non-linearity of the mapping. Thus, Bark and Cent are the more extreme scales while the mappings of ERB- and Mel-Scale are more moderate.

To wrap a linear frequency spectrum obtained via the DFT according to one of the auditory scales, one first defines frequency bands of equal bandwidth in the logarithmic domain. Via mapping the upper and the lower frequency boundary back onto the linear scale, one can then identify those frequency bins that fall into the corresponding frequency band. Now, to combine the linear frequency bins that fall into a logarithmically spaced band there exist two approaches. First, one can perform a *hard* mapping, where each bin belongs to exactly one logarithmic band and each band only contains distinct bins. Or, one can have a *soft* mapping, where the logarithmic bands are overlapping and each bin belongs to two logarithmic frequency bands but just to some degree. Figure 2.3 visualizes both mappings approaches for the Mel-Scale. In the upper plot the frequency intervals

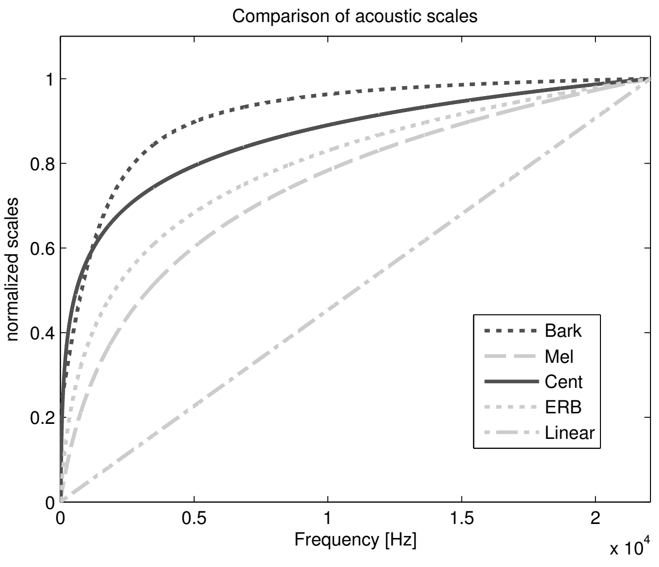


Figure 2.2: Comparison of auditory scales.

on the y-axis (the frequency in Mel) have equal bandwidth, while on the x-axis (linear frequency resolution) the size of intervals is logarithmically increasing. The weighting of the bins for both hard and soft mapping are illustrated in the upper and the lower plot. For the soft mapping triangular filters are used, while for the hard mapping all bins are equally weighted as illustrated in figure 2.3. It is worth mentioning that this mapping of linear to logarithmic frequency scale could also be realized directly via designing a specific filter bank of FIR or IIR filters. However, it is common practice in MIR to first compute the linear FFT and then perform this mapping onto a logarithmic auditory scale for performance reasons.

Up to this point the spectrum of audio signal as a whole has been discussed. In the following *Time-Frequency Representations* (TFR) are introduced, which represent the time-localized, short-time frequency content of an audio signal.

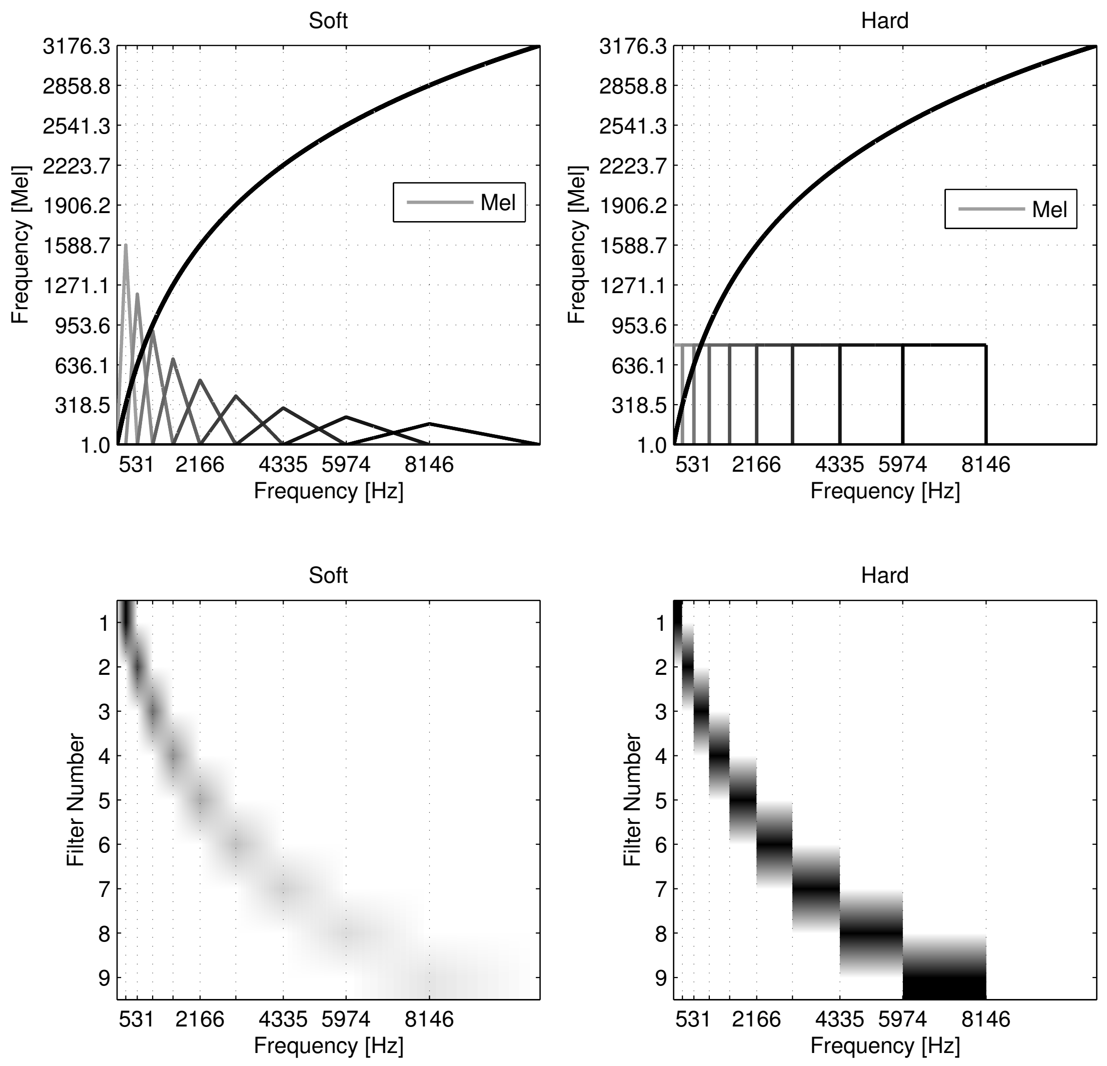


Figure 2.3: Visualization of **hard** versus **soft** mapping according to the logarithmic Mel-Scale.

2.4 Time-Frequency Representation

One assumption of the Fourier transform is that the analyzed signal $x(t)$ is a *stationary signal*, which basically means that the frequency content is constant over time. Unfortunately, the frequency content of audio signals is typically changing over time. Therefore, audio signals fall into the category of so-called *non-stationary* *signals*, but fortunately belong to a specific subclass called *quasi-stationary signals.* Quasi-stationary signals are non-stationary signals but can be modeled as being stationary within local time frames. To be able to capture the changing signal characteristics over time *Time-Frequency Representations* (TFR) are used.

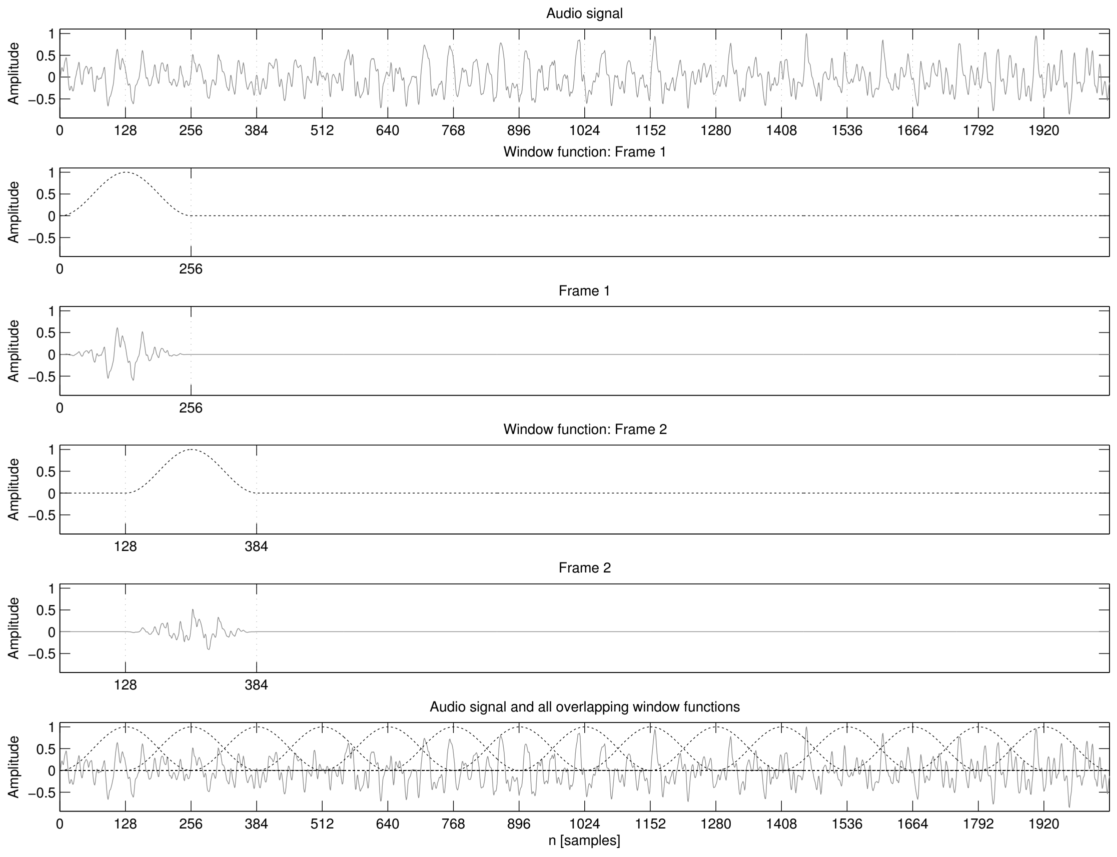


Figure 2.4: Windowing of an audio signal using a Hanning window function .

2.4.1 The Short-Time Fourier Transform

The most popular TFR is the *Short-Time Fourier Transform* (STFT) or *windowed Fourier transform.* To achieve time-localization, short signal excerpts of length N, which are assumed to be stationary, are consecutively cut out of the audio signal. Such short signal excerpts are called *audio frames.* The function that is used to extract an audio frame out of the whole audio signal is called *window function*. Figure 2.4 visualizes the frame wise processing of an audio signal. The size of an audio frame is called *frame size* or *window size*. Audio frames can be overlapping depending on the *hop size*, the number of skipped samples between two consecutive audio frames. To determine the local frequency content each audio frame is transformed to the frequency domain using the DFT. The resulting local-frequency spectrum of an audio frame is often simply referred to as *frame*. Mathematically the STFT can be defined according to equation 2.12, where $w[n]$ is the window function and m defines the position of the window. The definition of the STFT that is used here is discrete in both time and frequency.

$$\mathbf{STFT}\left[x[n]\right] \equiv X[m,k] = \sum_{n=0}^{N-1} x[n]w[n-m]e^{-j2\pi nk/N} \tag{2.12}$$

2.4.2 Window Functions

It is well-known that the window function $w[n]$ that is used to *"cut"* the individual audio frames out of an audio signal has a non-negligible impact on the obtained discrete Fourier spectrum. The influence of the window function on the spectrum can be deduced from the *convolution theorem*. Given that the discrete Fourier transform of the signal $x[n]$ is $X[k]$ and the discrete Fourier transform of the window function $w[n]$ is $W[k]$

$$x[n] \overset{\mathcal{F}}{\leftrightarrow} X[k] \tag{2.13}$$

$$w[n] \overset{\mathcal{F}}{\leftrightarrow} W[k] \tag{2.14}$$

and the audio frame $y[n]$ is the multiplication of the audio signal with the window function

$$y[n] = x[n]w[n] \tag{2.15}$$

then the discrete Fourier spectrum of the windowed signal $y[n]$ is the convolution of the discrete Fourier spectrum of the discrete Fourier transform of the signal and of the window function.

$$Y[k] = X[k] \star W[k] \tag{2.16}$$

Thus, each frequency component of the original spectrum is convoluted with the Fourier transform of the window function. For example windowing of a simple cosine signal of frequency ω causes its Fourier transform to have non-zero values at frequencies other than ω, although the analyzed signal does not contain these frequency components. This effect is known as *leakage effect*. Figure 2.5 visualizes this leakage effect for a cosine signal of 1000 Hz and a rectangular window and for the same cosine signal and a Hanning window. The leakage effect can be interpreted as smearing of the frequency spectrum. The Fourier transform of the window function defines how the frequency spectrum is smeared.

In the worst case frequency components of a signal cannot be resolved because of leakage effects. The width of the main beam of the Fourier transform of the window function defines how close two frequency components can be till one can no longer resolve them. The level of the side lobes define the masking level for other frequencies i.e. spectral components of small amplitude can be hidden by the leakage effect resulting from components with large amplitude. There exist various window functions, as for example *Hanning*, *Kaiser* or *rectangular* window. In MIR the Hanning window (see equation 2.17) is rather popular and if not stated otherwise, is used in this thesis as the standard window function.

$$w[n] = 0.5(1 - cos(\frac{2\pi(n-1)}{N})), \qquad 0 \leq n < N \tag{2.17}$$

2.4.3 Time-Frequency Trade-off

Although it would be desirable, a Time-Frequency Representation cannot have arbitrary time and frequency resolution. For the STFT this trade-off between time and frequency resolution is known as *Uncertainty Principle*. The Uncertainty Principle states that one cannot simultaneously increase both the time and the frequency resolution of a STFT. The time resolution ΔT is defined by the window size. Given that an audio frame contains K samples the time resolution is $\Delta T = KT_s$, where T_s is the sampling period. The frequency resolution Δf of a STFT is the frequency interval between two consecutive Fourier Coefficients, which is $\Delta f = \frac{f_s}{K}$ as the discrete frequency bins are equidistant samples of the Fourier spectrum.

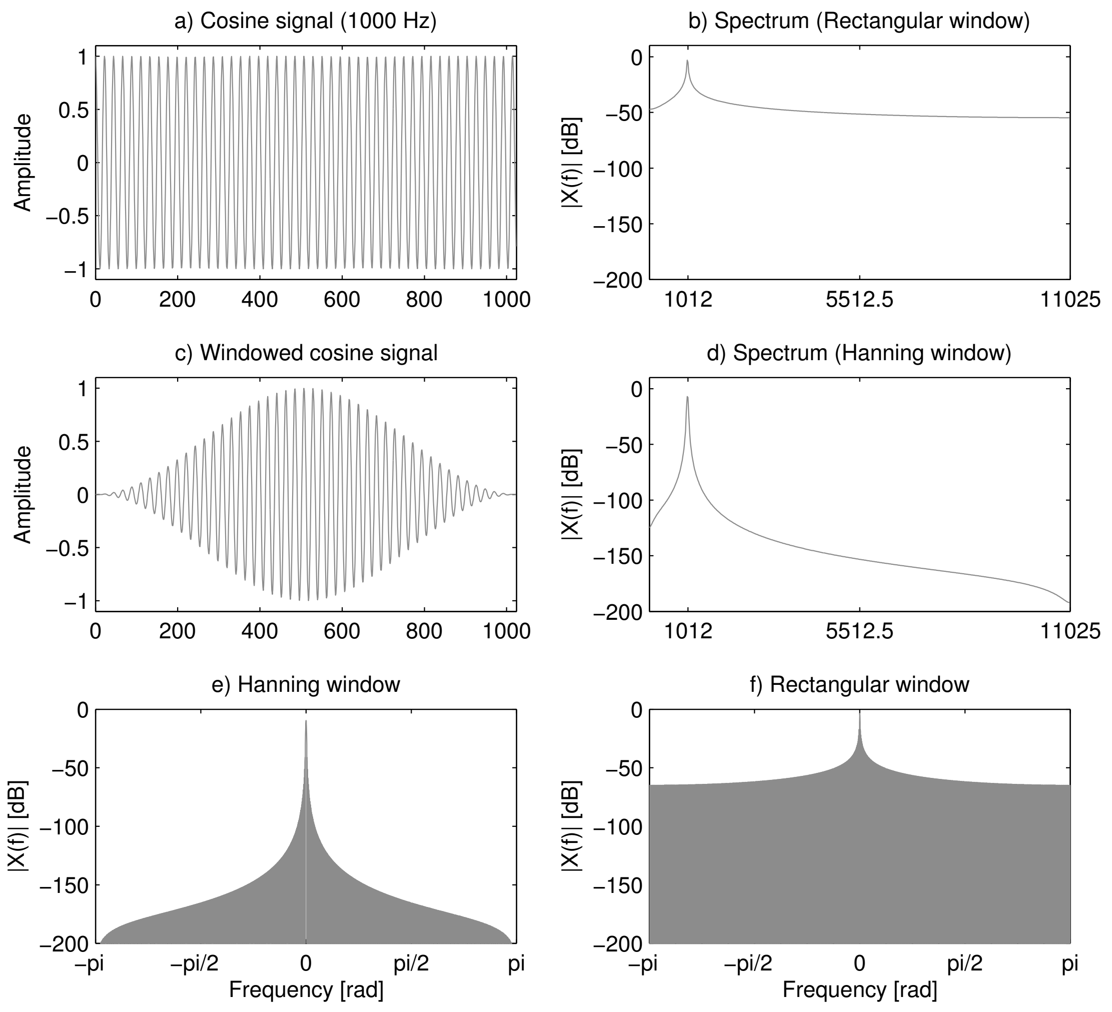


Figure 2.5: Windowing of an audio signal using a Hanning window function .

Based on the relation between sampling rate and sampling period $f_s = \frac{1}{T_s}$ one can infer the inverse relation of the time-resolution ΔT and frequency-resolution Δf of the STFT.

$$\Delta f = \frac{f_s}{K} = \frac{1}{KT_s} = \frac{1}{\Delta T} \tag{2.18}$$

A special visual representation of the STFT is the *spectrogram*. In a spectrogram time increases linearly across the horizontal axis and frequency increases across the vertical axis. For each audio frame along the horizontal time axis the corresponding magnitude spectrum $|X(f)|$ is vertically visualized as gray scale or color values. Figure 2.6 shows the spectrograms of two STFT with different time-frequency resolutions. For spectrograms and in general in MIR a logarithmic amplitude scale is used (see equation 2.5) as the human loudness perception is also roughly logarithmic. In figure 2.6 one can see that using a higher time resolution (lower plot) allows to better capture onset times, while using a better frequency resolution allows to better resolve the frequency components of sustained tones, which are resolved in a more fine-grained way.

The following two subsections now focus on two specific TFR. The first TFR is based on the well-known *Mel Frequency Cepstral Coefficients* (MFCCs) and is a standard representation commonly used in MIR. The second TFR is the Cent-Spectrum and will be used by many of the algorithms presented in this thesis.

2.4.4 Mel Frequency Cepstral Coefficients (MFCCs)

Mel Frequency Cepstral Coefficients (MFCCs) are one way to represent the *spectral envelope* of the spectrum of an audio frame. The spectral envelope is defined as the shape of the power spectrum of a short segment of an audio signal, e.g., an audio frame [Schwarz and Rodet, 1999]. In figure 2.7 the spectral envelope of the power spectrum of an audio excerpt of a violin is visualized. The spectral envelope captures perceptually important information about the corresponding sound excerpt. One important aspect of a spectral envelope is that sounds with similar spectral envelopes are generally perceived as similar.

To represent the spectral envelope of an audio frame via an MFCC vector one has to perform three steps. First, the linear frequency resolution of the magnitude spectrum is mapped onto the logarithmic Mel-frequency scale as described in section 2.3.5. Then the obtained Mel-Spectrum is transformed to logarithmic amplitude scale according to equation 2.5. Finally, the logarithmic Mel-spectrum is decorrelated by a *Discrete Cosine Transform* (DCT). The resulting DCT coefficients are the Mel Frequency Cepstral Coefficients. To get a compact and smooth approximation of the spectral envelope only the first few DCT coefficients are kept.

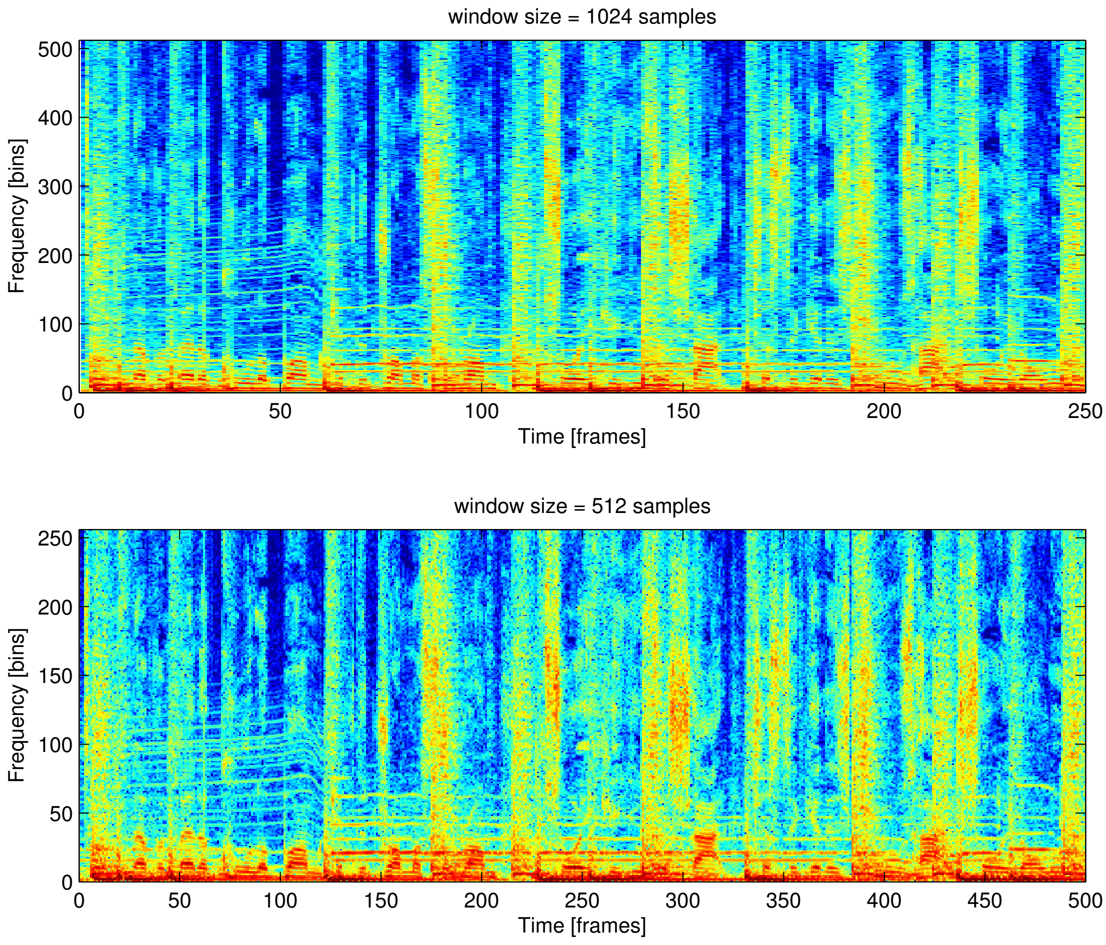


Figure 2.6: Spectrogram of the same audio excerpt for two different window sizes. In the upper plot the frequency resolution is better (twice as high), while in the lower plot the time resolution is higher (twice the number of frames).

Keeping only the first few coefficients can also be interpreted as low-pass filtering the logarithmic Mel-spectrum. As the name already indicates there exists a strong relation between the MFCCs and the *Cepstrum*, the inverse Fourier Transform of the Log-spectrum, as defined in equation 2.19.

$$c_n = \frac{1}{N} \sum_{k=0}^{N-1} e^{j\frac{2\pi}{N}nk} \log |X[k]| \tag{2.19}$$

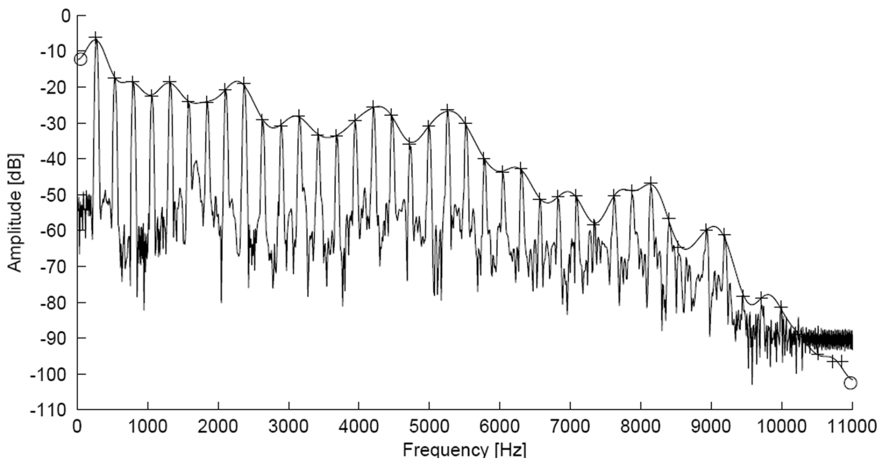


Figure 2.7: Spectral Envelope of a violin (from [Schwarz and Rodet, 1999]).

In contrast to the definition of the Cepstrum in practice for the computation of MFCC vectors the discrete cosine transform is used instead of the inverse Fourier transform. In figure 2.8 three Time Frequency Representations are compared. In the upper plot the magnitude spectrum is visualized, in the middle plot the Mel-spectrum and in the lower plot the MFCC representation is shown. One can see that because of the DCT compression the direct relation to the spectral representation is lost for the MFCC representation.

In the next section the Cent-Spectrum is introduced and it is empirically shown that the auditory Cent-scale is especially suitable for music signals.

2.4.5 Cent Spectrum

While many signal processing techniques in MIR simply use the Bark- or the Mel-Scale to account for the logarithmic frequency perception of the human ear, here a simple experiment is presented that indicates that the Cent-Scale is indeed a good choice to account for the musical nature of digital audio signals in MIR.

The main idea of this experiment is that the frequency resolution of a Time-

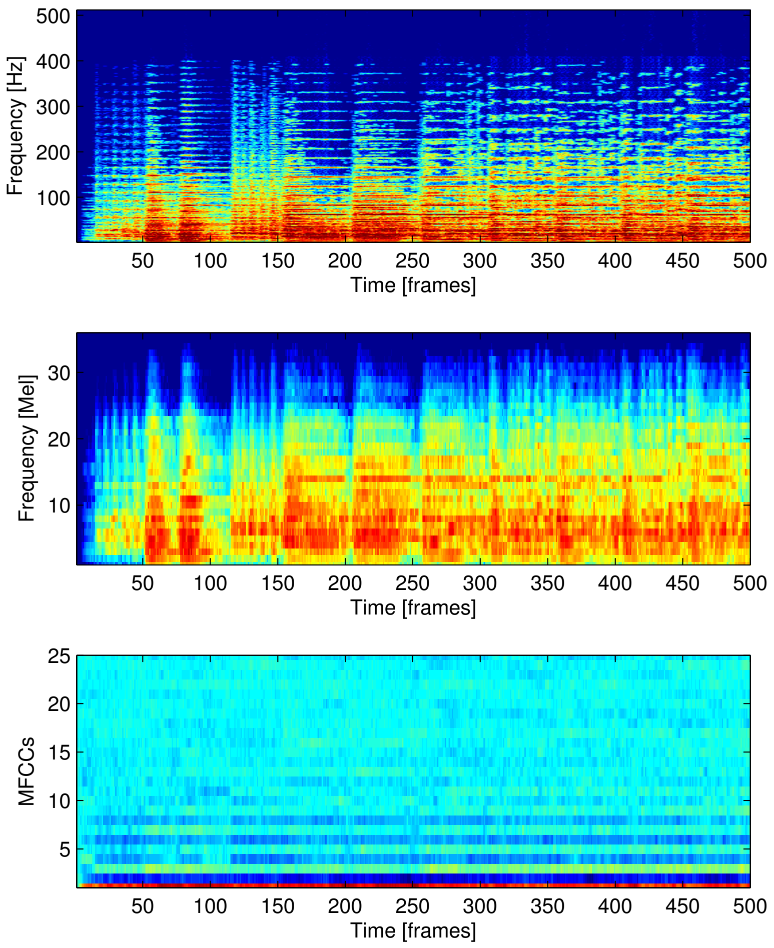


Figure 2.8: Three Time-Frequency Representations (the Spectrogram, the Mel-scaled Spectrogram and MFCCs) .

Frequency representation should be adapted in such a way that the actually observed frequency content of music signals can be adequately represented. To obtain a model of the typical frequency content of audio files all songs of the *'1517-Artists'* evaluation collection (see 3.2.3) were transformed to the frequency domain and the magnitude spectrogram was computed. Then mean and standard deviation for each frequency bin over all magnitude spectrograms of all files were computed separately for each genre to also reveal genre specific effects.

Figure 2.9 shows the results of this experiment. Some frequency bins have higher

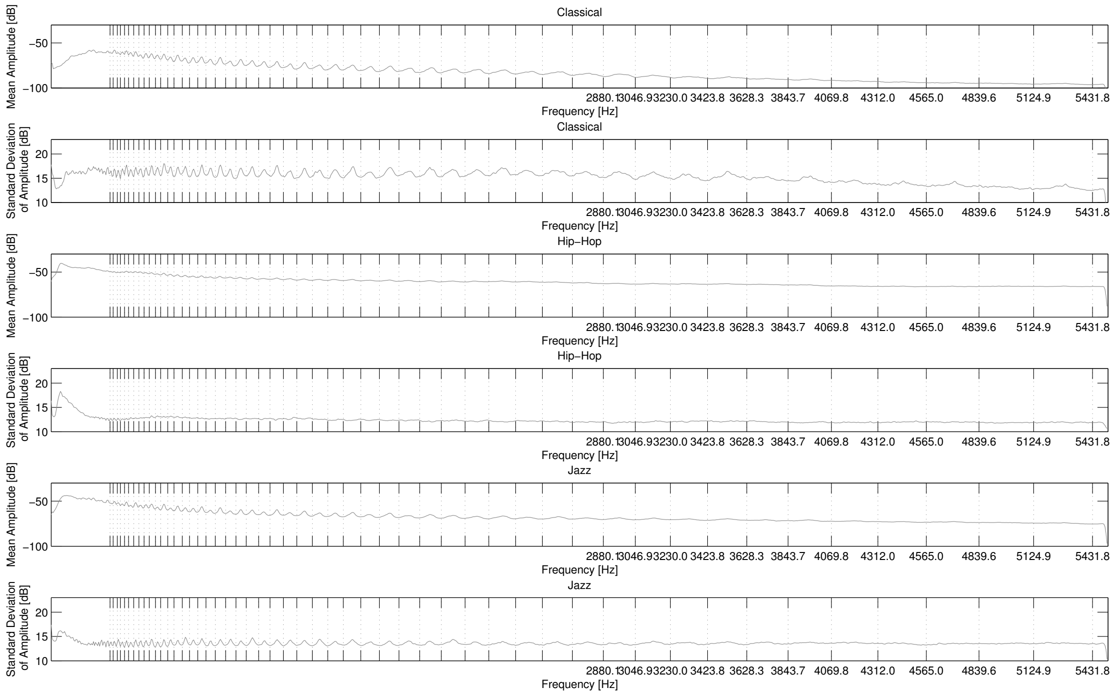


Figure 2.9: Observed mean over the frequency content of the audio files of the '1517-Artist' evaluation collection.

mean and variance than others. This is indicated by the peaks in these plots. Additionally, in figure 2.9 the tics of the x-axis have been adjusted such that the intervals correspond to semitone steps according to the Cent-Scale. A semitone is equal to an interval of precisely 100 cent. The tics start at 5050 cent. The borders of the semitone intervals precisely fall into the valleys for both the empirical mean and the empirical standard deviation of the frequency bins in the plots. Consequently the Cent-Scale perfectly fits the observed frequency distribution and the cent-scaled frequency bins would maximize the captured variance. Furthermore, one can see in figure 2.9 that the influence of the western musical scale differs depending on the genre. Some genres like *Classical* and *Jazz*, which have been picked here as examples, show like many other genres a strong influence of the western musical scale, whereas the influence seems to be marginal for the genre *Hip-Hop*. Thus, not only from a musical point of view, but also based on this observation it seems reasonable that the Cent-Scale as a logarithmic auditory scale is a good choice for music signals.

It is worth mentioning that taking advantage of the western musical scale is not a fundamentally new idea, but is the basic idea of the *constant Q transform* [?], which is known to be especially suitable for music signal processing. Thus, the Cent-Spectrum is related to the constant-Q transform and can interpreted as a fast approximation thereof, but with a constant analysis window length for all frequency bins.

Many of the presented algorithms in this thesis are based on the *Cent-Spectrum*. To obtain the Cent-Spectrum during the STFT a Hanning window with a window length of 2048 samples is used. For performance reasons a hard mapping of the the linear magnitude spectrum onto Cent-Spectrum is performed and logarithmic amplitude scale is used. The last section of this chapter focuses on audio normalization techniques, which are useful to make spectral representations intensity invariant.

2.5 Audio Normalization

Audio normalization is an important pre-processing step, as audio files are often recorded at different volume levels. From a technical point of view this means that the whole audio signal $x[n]$ is amplified by a constant factor a.

$$\hat{x}[n] = ax[n] \tag{2.20}$$

The magnitude spectrum $|\hat{X}[k]|$ of the amplified signal is also scaled by the constant factor a as the Fourier transform is a linear transformation.

$$|\hat{X}[k]| = a|X[k]| \tag{2.21}$$

As most of the algorithms process the magnitude spectrum on a logarithmic amplitude scale (in dB) we focus on the amplified magnitude spectrum. The amplified magnitude spectrum (in dB) is offset by a constant.

$$20\log_{10}(|\hat{X}[k]|) = 20\log_{10}(a|X[k]|) = 20\log_{10}(a) + 20\log_{10}(|X[k]|) \tag{2.22}$$

In many cases it can be advantageous to be intensity invariant, which can be achieved via an audio normalization. A simple approach that is used in some audio applications is to subtract the mean of each frame. Removing the mean of each frame of course makes the spectral representation invariant to the constant offset. Especially for systems that use MFCCs this can be easily realized as the first MFCC coefficient represents the mean over the whole frame. Thus, by removing the first coefficient, MFCC vectors get intensity invariant. However, by using such an approach the local intensity information is lost, as all frames will have zero mean. The only information left is the spectral envelope of the audio frame. In this thesis a variant of this standard audio normalization approach is used. To keep some local intensity information but still make the whole audio signal intensity invariant the constant offset of a frame is estimated not just based on a single local frame, but using a fixed size neighborhood (for the audio normalization performed in this thesis a neighborhood of +-100 frames is used) around each frame. From each frame the mean of its neighborhood is removed to obtain the intensity invariant Cent-Spectrum. This intensity invariant representation is then called *normalized Cent-Spectrum* and is a commonly used time-frequency representations in this thesis.

This chapter has introduced all the necessary signal processing fundamentals that will be used in the subsequent chapters of this thesis to describe the audio similarity algorithms that are then used to generate music recommendations.

3 Evaluation of Music Similarity Algorithms in the Context of Recommendation

This thesis focuses mainly on qualitative improvements of content-based music recommender systems. Consequently, one central aspect is to measure the quality of music recommender systems. This chapter will present general evaluation methods of music recommender systems, quality measures and the genre classification datasets that are used in throughout this thesis. In the context of genre classification two special issues will be discussed. First the negative impact of album and artist effects and second how to also make the genre classification performance of automatic systems comparable to human performance on the same dataset. The next section will start with an introduction to standard evaluation methods of music recommender systems.

3.1 Direct Evaluation of Recommender Systems

Basically there exist two methods to evaluate music recommender systems. Music recommender systems can either be directly evaluated via *listening experiments*, or via *A/B-tests*. Both methods will be briefly introduced in the following.

3.1.1 Listening Experiments

The outline of a typical listening experiment is as follows: First, to set up the experiment an appropriate number of participants is needed depending on the required statistical power. Then to each participant a seed song and a list of recommendations is presented. The recommendation list contains the recommendations generated by two or more systems in randomized order. The participants then have to rate the individual recommended songs, hence they have to decide if these are appropriate recommendations or not. In practice one can only perform a limited number of listening tests as listening experiments are in general expensive, require a lot of human resources and are typically time consuming. Thus, for general prototyping and fast evaluation of new approaches listening tests are not well-suited, but for yearly competitive evaluations, e.g. the **Audio Similarity and Retrieval** (AMS) task of the ***M**usic **I**nformation **R**etrieval **E**valuation e**X**change* (MIREX)[1], listening test can be very enlightening and especially can help to monitor the scientific progress.

3.1.2 A/B-Tests

The second possibility to perform a direct evaluation is an `A/B`-test [Ash, 2008]. For an `A/B`-test a large group of users is required. The set of users is then split into two subgroups of equal size. Then depending on the user's group the recommendations generated by system **A** or by system **B** are presented. For both groups the user's actions like click-rate, songs-skips, retention time or number of song purchases are measured. Any significant difference in recommendation quality is expected to have an impact on the measured user actions aggregated over all users. Unfortunately, to perform `A/B`-tests one essentially needs access to an online music platform that is already heavily used. Thus, this is mainly a practical solution for companies that have an already established business, but not for scientific research.

[1]http://www.music-ir.org/mirexwiki

Clearly for rapid scientific prototyping both evaluation methods are not suitable. Consequently, it is common practice to use a workaround to evaluate music recommender systems: *genre classification.*

3.2 Genre Classification - A Workaround

As both direct evaluation methods are no practical solutions for prototyping, in MIR research one tries to automatize the evaluation process. The basic assumption that automatic evaluations are based on is that songs that *"sound similar"* are also good recommendations. Thus, by estimating pairwise song similarities a ground truth for the automatic evaluation can be defined. However, generating a reliable ground truth with respect to song or artist similarity is a well discussed [Ellis et al., 2002], but still open problem. One main issue is that the effort of estimating pairwise song similarities grows quadratically with the number of songs in a dataset. Consequently, directly estimating pairwise song similarities e.g. based on a listening test is only feasible for very tiny datasets. Furthermore pairwise song similarities are hard to obtain, since musical similarity itself is a not well-defined concept [Ellis et al., 2002].

In fact by taking a look at how people organize their music it turns out that estimating pairwise song similarities is not a human way to assess musical similarity. Typically, humans organize their music collections by defining different subcategories. These categories as a whole reflect music that is for some reason considered similar. In many cases these categories correspond to genres. In some other cases the categories cover some period of time e.g. the *"80ies"*. Often such categorization schemata are also inconsistent. For example results of an informal experiment show that about 10% of all songs in a personal music collection are duplicates, which can be interpreted as one indicator for inconsistencies that exist in human categorization schemata.

Nonetheless, it is common practice to make use of these user defined categories, which are assumed to contain groups of similar songs, and interpret the categorization as ground truth. This way the task of evaluating music recommender systems can be interpreted as a classification problem, where the songs in the recommendation list should belong to the same category as the query song. This allows to automate the evaluation procedure of music recommender systems. Once a collection is organized into categories, the collection can be used as ground truth to evaluate different recommendation algorithms.

As this is the standard approach in MIR the evaluations conducted in this thesis follow this procedure and assess the quality of music recommender systems in an indirect way, via *music classification*. Of course any music categorization schema for example based on mood, instrumentation or singing voice can be used for music classification. However, the most widespread and most popular way to subcategorize music is according to the musical genre. This is also reflected by the typically genre based classification datasets that are used for evaluations. For this reason this evaluation method is often referred to as *music genre classification*. One big advantage of genre as classification criterion is that for many songs genre information can be automatically collected from the web, record labels or from personal music collections.

While it seems straightforward to compare different music recommendation systems via genre classification, using genre information as evaluation criterion requires a careful design of the evaluation procedure. First of all, although genre information is widely used to organize and structure music collections, genre definition and attribution is generally accepted to be subjective [Sordo et al., 2008]. Sordo et al. show based on some large-scale analyses that some genres are clearly defined both by experts and the wisdom of crowds, while for other genres it is difficult to get a common consensus on their meaning. Thus, there is some general consensus on genre labels, but the perception of genre can vary from individual to individual. Furthermore, one has to accept that there do exist genre inconsistencies (or genre ambiguities) and annotation errors in any genre classification dataset. Therefore to correctly cope with these problems, it would be necessary to get rid of a single ground truth annotation and reflect multiple opinions in the evaluation datasets. This problem will be addressed in section 3.4.4 of this chapter, where two quality measures based on the user ratings obtained from a listening experiment are defined.

Another problem with respect to the evaluation via genre classification are the genre classification datasets themselves. On the one hand, some of the datasets that are used in the literature are not publicly available, which makes the comparison of systems impossible as a comparison of classification results is only valid on one and the same dataset. On the other hand, most of the datasets that are publicly available have not been designed to be used for the evaluation of music recommender systems. Because of the presence of strong album-, artist- and portfolio-effects the results obtained on these datasets are unreliable (see section 3.3). To address this problem the datasets used in this thesis are all (except one dataset of minor importance) publicly available. Three of the datasets presented in section 3.2.3 are novel and have been carefully designed to help facilitate the evaluation process via genre classification.

Finally, it is not only important to have appropriate evaluation datasets, but also to use appropriate quality measures to measure and compare the performance of different music recommendation system on a specific dataset. The quality measures that are used in this thesis are presented in the following subsection and the reliability of these quality measures is then investigated in subsection 3.2.2.

3.2.1 Quality Measures

Within this thesis three different quality measures are used with respect to genre classification experiments. Two quality measures, the *k-Neighbor Accuracy* and *R-Precision*, are related to a query or recommendation scenario and the third quality measures is the standard *Classification Accuracy* which is a common evaluation metric for arbitrary classification tasks.

3.2.1.1 Query-Scenario based Quality Measures

Both quality measures presented in this subsection (k-NA and R-Prec) are related to a query scenario, where the algorithm is asked to return a set of recommendations given a query song. For such a result set one counts the number of *correctly* returned songs. A song in the result set is assumed to be correct w.r.t. our evaluation if the genre is the same as the genre of the query song. The following paragraph gives a mathematically more precise definition of the various quality measures we computed.

Consider a music collection of n tracks separated into p genres, then a classification function *classify*, which counts the number of songs belonging to genre g in a result set of size r given S_i as query song, is defined by

$$\text{classify}(S_i, g, r) = \sum_{j=1}^{n} \mathcal{G}(S_j) == g \wedge \mathcal{R}(S_j|S_i) < r \tag{3.1}$$

where $\mathcal{G}(S_j)$ denotes the genre of song S_j and $\mathcal{R}(S_j|S_i)$ denotes the rank according to the similarity measure of the song S_j given the song S_i as query song.

3.2.1.1.1 k-Neighbor Accuracy (k-NA) The k-Neighbor Accuracy[2] for a collection of size n is given by

$$\text{k-NA} = \frac{\sum_{i=1}^{n} \sum_{j=1}^{k} \text{classify}(S_i, \mathcal{G}(S_i), k)}{kn}. \tag{3.2}$$

The k-NA is an important measure because in quite many applications only the top ranked songs (i.e., the adequate recommendations) are of use. Unfortunately, the k-NA is extremely prone to artist and album effects (see section 3.3). Thus, the k-NA should be used only in combination with an artist filter. It is also important to note that the k-NA is not comparable to the classification result of a k-NN classifier. In k-NN classification the genre of the song to be classified is unknown and is estimated based on the genre of the k in the recommendation set, whereas k-NA is the percentage of songs in the recommendation set that have the same genre as the query song.

3.2.1.1.2 R-precision (R-Prec) The R-precision, in contrast to the k-NA, considers all songs of the genre of the query song. Hence, for a given query song the

[2]With respect to the MIREX Audio Music Similarity and Retrieval evaluation the k-Neighbor Accuracy is called *Neighborhood Clustering* instead.

size of the result set is equivalent to the size of the song's genre. The R-precision for a query song S_i is defined according to [Aucouturier, 2006]

$$\text{prec}(S_i) = \frac{\text{classify}(S_i, \mathcal{G}(S_i), n_{\mathcal{G}(S_i)})}{n_{\mathcal{G}(S_i)}}. \tag{3.3}$$

The global R-precision over a whole collection is defined as the mean over all songs.

$$\text{R-Prec} = \frac{\sum_{i=1}^{n} \text{prec}(S_i)}{n} \tag{3.4}$$

3.2.1.2 Classification-Scenario based Quality Measures

In contrast to the query scenario, where the genre of the query song is known, in a classification task the genre labels have to be predicted by the classifier. Furthermore, in a classification scenario both query song and recommendation list are not defined, but the collection of songs is just split into a *training set* and a *test set*. It is then the task of the classifier to correctly predict the class labels of the songs in the test-set given the songs in the training-set as training data. Thus, the classification-scenario is more general and any arbitrary classification algorithm can be used for a genre classification task, but only the results obtained via nearest neighbor classification have a semantic interpretation in the context of the evaluation of music recommender systems. Still the results obtained by other classifiers than the nearest neighbor algorithms are useful to identify the general discriminative power associated with a specific feature set.

3.2.1.2.1 Classification Accuracy (CA) For all classification experiments that are conducted in this thesis the classification accuracy as a standard quality measure is reported. Since genre classification is in general a multi-class classification problem, the overall classification accuracy is used to summarize the obtained classification performance.

$$\text{classification accuracy} = \frac{\text{number of correctly classified instances}}{\text{number of total instances}} \tag{3.5}$$

As a name convention within this thesis classification accuracies are always denoted by the classifier name concatenated with *"CA"* for classification accuracy e.g. *"SVM CA"* or *"5-NN CA"*. The classification accuracies reported in this thesis are either based on a leave-one out cross-validation or on ten times 10-fold cross-validation. In case of cross-validation the average of the accuracy over the individual runs is reported, as the obtained results can vary depending on the random splits into the cross-validation folds.

3.2.2 Reliability of Quality Measures based on Genre Classification

As genre classification is only a workaround to evaluate music recommender systems one main question is how reliable the obtained quality measurements are. Hence, do systems that achieve higher genre classification scores really generate better music recommendations than systems with a lower score?

A deeper insight into the relationship between genre classification scores and recommendation quality can be gained by analyzing the data collected by the MIREX Audio Music Similarity and Retrieval Tasks. In [Pohle, 2010] Pohle showed that there does exist a strong correlation between the user similarity ratings and the genre classification accuracies for the MIREX 2007 competition. Here these experiments are extended and the results obtained in 2009 and 2010 are analyzed in more detail to further emphasize the strong correlation between user similarity ratings and genre classification-based quality measures.

Within the MIREX Audio Music Similarity and Retrieval Tasks each algorithm has to compute a similarity matrix for an evaluation collection containing 7000 songs. Out of this similarity matrix the top-5 most similar songs for 100 different query songs are determined. Then human graders are asked to rate the songs that are proposed by the participating algorithms and presented together in randomized order for a given query song. Two scores have to be assigned by the human graders for each song: the *broad-score* and the *fine-score*. The broad-score measures the degree of similarity to the query song on a three level scale (*"not* *similar"*, *"somewhat similar"*, *"very similar"*). The fine score can be any value in between the continuous interval of 0 to 100. Besides the human evaluation judgments also objective statistics are derived from the distance matrices generated by the algorithms. These objective statistics also include the *k-Neighbor Accuracy* (k-NA) for different neighborhood sizes (5, 10, 20 and 50). Both artist filtered and non-artist filtered results are reported. Figure 3.1 shows the obtained results of the participating algorithms in 2009 and 2010. The left plots show the original data, whereas in the right plot the scales of the quality metrics have been adjusted via mean-variance normalization to bring the quality measure to the same scale.

Even visually one can see that there does exist a strong correlation between the user generated similarity assignments and automatically obtained genre classification accuracies. In table 3.1 the Pearson correlation coefficient between the two human generated similarity scores and the reported objective statistics (k-NA, k-NA artist filter) are summarized. Obviously, the artist filtered k-NA quality measures yield a higher correlation with the human similarity ratings. Regarding the not artist filtered quality measures those considering a bigger neighborhood seem to better correspond to the human judgments. For the artist filtered quality k-NA measures a neighborhood size of $k = 20$ seems to have optimal predictability of the human similarity assessments. Thus, in general one can conclude that using a larger neighborhood size is advisable. Another interesting detail is that there does exist an extremely strong correlation between the broad and the fine score. For the MIREX 2009 competition the correlation coefficient between these two scores is 0.9978 and for the MIREX 2010 competition the correlation coefficient is 0.9983. Consequently, the judgments of the human graders are very consistent over both scores.

Altogether, the analysis of the MIREX data shows that genre classification-based quality criteria are quite reliable quality indicators in the context of music similarity estimation. Although this MIREX task clearly focuses on music similarity and not explicitly on music recommendation, this result still provides some justification for using genre classification as an automatic evaluation method for music recommender systems. Interestingly, one of the author's personal findings from participating in the MIREX evaluation as a grader is that many of the proposed

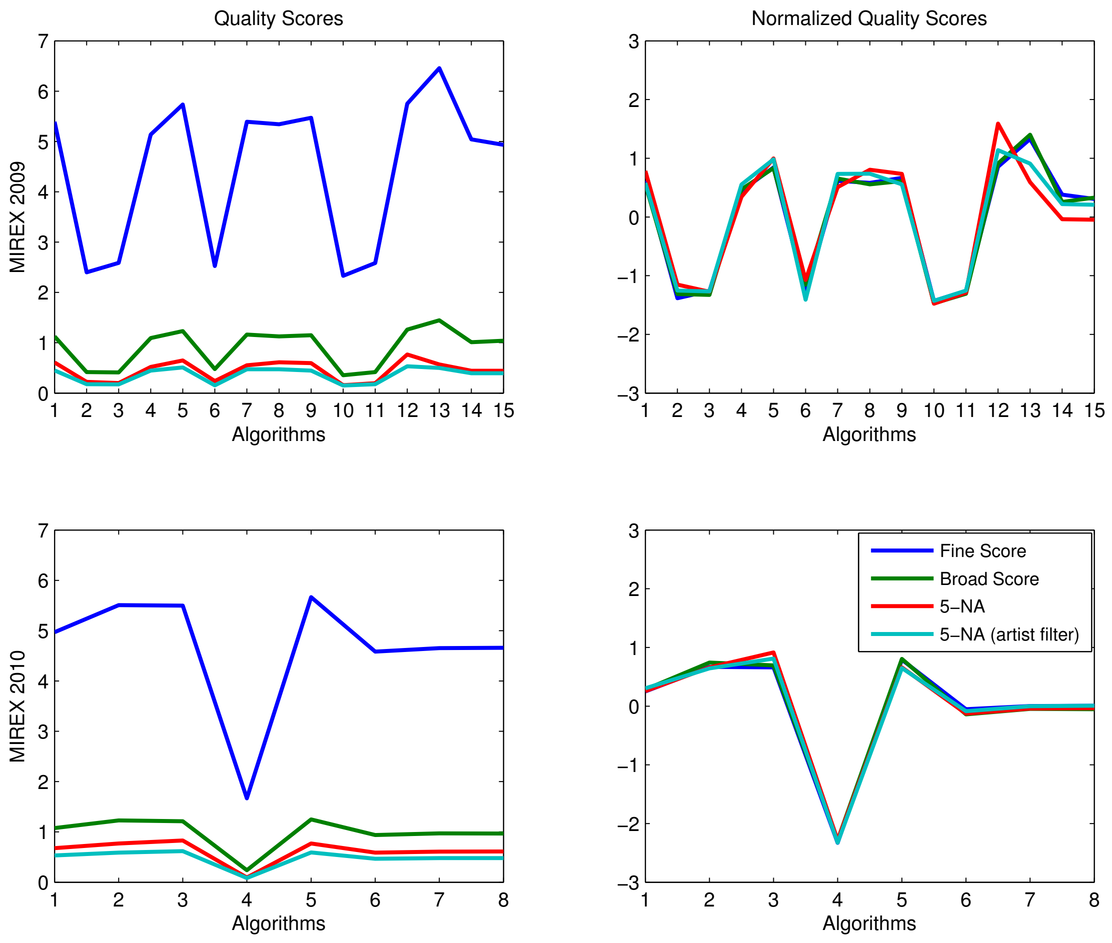


Figure 3.1: Analysis of the user assigned broad and fine scores and the neighborhood accuracies (left: original scores; right: normalized scores).

songs by the participating algorithms seemed to come just from a single artist or are just perfectly similar to each other, which makes listening to the long evaluation lists a tiring task. To increase the diversity of the proposed songs using a portfolio filter (as proposed in section 3.3.2) in the evaluation would be a good idea especially in the context of evaluating music recommendation.

In the next section the genre classification datasets that are used within this thesis for evaluation are presented.

MIREX 2009				
	with Artist Filter		without Artist Filter	
indicator	**fine**	**broad**	**fine**	**broad**
5-NA	0.9856	0.9836	0.9422	0.9439
10-NA	0.9886	0.9869	0.9339	0.9367
20-NA	**0.9906**	**0.9893**	0.9553	0.9586
50-NA	0.9873	0.9857	**0.9860**	**0.9884**
MIREX 2010				
	with Artist Filter		without Artist Filter	
indicator	**fine**	**broad**	**fine**	**broad**
5-NA	0.9966	0.9958	0.9933	0.9945
10-NA	**0.9966**	0.9962	0.9935	0.9949
20-NA	0.9964	**0.9967**	0.9939	0.9957
50-NA	0.9957	0.9966	**0.9946**	**0.9965**

Table 3.1: Pearson correlation between genre neighborhood accuracy, k-NA for (k = 5, 10, 20 and 50), and human assigned broad and fine scores for the MIREX 2009 and MIREX 2010 results.

3.2.3 Genre Classification Datasets

In this thesis eight different genre classification datasets are used for the evaluation. Three of these (*"GTZAN"*, *"ISMIR 2004 Genre"* and *"Ballroom"*) are well-known datasets and should help to make the evaluation results comparable to previous results in the literature. However, we suspect that there is a significant artist effect in these three datasets. For the other five datasets (*"Homburg"*, *"1517-Artists"*, *"Annotated"*,*"103-Artists"* and *"Unique"*), artist information is available and we use an artist filter to prevent any artist or album effects. Table 3.2.3 gives a compact overview on the used datasets. Detailed information on the class distribution of each dataset can be found in **Appendix A**.

dataset	**#tracks**	**#artists**	**#genres**	**smallest / largest class**
103-Artists	2445	103	21	1.8% / 10.43%
1517-Artists	3180	1517	19	3.963% / 5.88%
Annotated	190	190	19	5.26% / 5.26%
Unique	3115	3115	14	24.59% / 0.83%
GTZAN	1000	N/A	10	10.00% / 10.00%
ISMIR 2004	1458	N/A	6	3.57% / 43.9%
Homburg	1886	1463	9	2.49% / 26.72%
Ballroom	698	N/A	8	8.6% / 15.9%

Table 3.2: A comparison of genre classification datasets used in this thesis.

3.2.3.1 GTZAN

The GTZAN dataset contains 1000 song excerpts (30 seconds) evenly distributed over 10 genres. Unfortunately, there is no artist information available and the number of different artists is unknown. However, we expect an artist effect in this dataset, as listening to some of the songs reveals that some artists are represented with several songs.

3.2.3.2 ISMIR 2004 Genre

The ISMIR 2004 Genre dataset was used as ground truth in the first public evaluation of content-based algorithms in 2004[3]. There exist two different versions of the dataset. In our evaluation we will use the larger dataset consisting of 1458 full length tracks distributed over 6 genres. The class distribution is unequal. The dominating class *"classical"* comprises 43.9% of all songs. Artist information is only partially available, but the number of different artists is rather low and consequently a strong artist effect is to be expected.

[3]http://ismir2004.ismir.net/ISMIR_Contest.html

3.2.3.3 Ballroom

The ballroom collection consists of 698 song excerpts (30 seconds) of 8 different dance music styles (Cha-cha, Jive, Quickstep, Rumba, Samba, Tango, Viennese Waltz and Waltz). The genre distribution is quite uniform. The category with the fewest examples is represented by 8.6%, the largest category by 15.9% of all examples. Artist information is missing, but for a part of the songs album information is available. We expect that there exists an album or artist effect on this dataset, since there are often many examples that seem to belong to one album.

3.2.3.4 Homburg

The *"Homburg"* dataset [Homburg et al., 2005] contains 1886 songs by 1463 different artists. Consequently, the artist effect should be relatively small, but still we will use an artist filter on this dataset since artist information is available. The rather short song excerpts of 10 seconds length are unequally distributed over 9 genres. The largest class contains 26.72%, the smallest class 2.49% of all songs.

3.2.3.5 1517-Artists

This genre classification dataset consists of freely available songs from download.com[45]. It has been previously used in [Seyerlehner et al., 2008] and also in [Seyerlehner et al., 2009c]. To ensure reasonable track quality approximately the 190 most popular songs (according to the number of total listenings) were selected for each genre. Altogether there are 3180 tracks by 1517 different artists distributed over 19 genres. It is worth mentioning that this collection has an almost uniform genre distribution, contains tracks from a large number of different artists, and can be freely downloaded from the author's personal homepage[6].

[4]http://music.download.com/

[5]The `http://music.download.com/` began redirecting all artist pages and category doors to corresponding pages on their sister music site Last.fm on March 2009.

[6]www.seyerlehner.info

3.2.3.6 Unique

Dataset *"Unique"* contains 30 seconds song excerpts from 3115 popular and well-known songs distributed over 14 genres and was used in [Seyerlehner et al., 2010b]. The dataset is compiled in such a way that no two songs by one and the same artist are in the dataset. Consequently, there is no need to apply an artist filter. The class distribution is skewed. The smallest class accounts for 0.83%, the largest class for 24.59% of all songs.

3.2.3.7 Annotated

The *"Annotated"* dataset is basically a subset of the songs of the *"1517-Artists"* dataset. For each genre some examples have been randomly drawn of which very untypical examples were manually removed. Furthermore, the imprecisely defined genre *"Religious"* was removed and the genre *"Easy Listening & Vocals"* was split into two separate genres. All artists in this dataset are unique and all genres consist of an equal number of songs. This dataset has been used in a user study on human genre classification [Seyerlehner et al., 2010a]. Besides the ground truth genre information, the genre ratings from the participants of the user study are available for this dataset. For this reason this dataset was named *"Annotated"*.

3.2.3.8 103-Artists

The *"103-Artists"* is the only in-house collection that is used in this thesis. It is one of the first larger genre classification datasets and has been used in [Pampalk, 2010] and [Pohle, 2010] and was named *DB-L* there. On average there are 23.74 songs per artist in this collection and an average of only 4.9 artists per genre. Consequently, on this dataset one can observe an extreme artist effect. Because of the extreme artist effect this dataset is not an adequate dataset for the evaluation of music recommender systems and has only been used in some early experiments.

In the following section the impact of album and artist effects on the evaluation results will be discussed in detail.

3.3 Artist and Album Effects

One major issue related to the evaluation of music recommender systems via music genre classification is that production specific effects can have a great influence on the quality measures introduced in 3.2.1. The influence of production specific effects on the introduced quality measures can be explained as follows:

Naturally, songs of the same artist often sound very similar to each other, because of the very specific characteristics of the artist. From an acoustic point of view the characteristic style of an artist is related to the instrumentation, the singing voice and the type of music played. As a consequence, even the spectral representations of the songs of one artist are often very similar to each other. Consequently, identifying songs of one and the same artist is relatively easy using spectral audio similarity algorithms. This effect is known as *artist effect*. In some cases even album specific production effects have an important influence on the spectral representation of songs, which is then called *album effect*. Artist and album effects are not a problem in the context of music similarity computation, as it is reasonable to assume that all songs of one artist sound similar. In the context of recommendation it is however problematic, since then in many cases the recommended songs will be songs of the same artist as the query song, which can be annoying for users. A music recommender system should clearly present interesting and probably complex relationship among songs, but not trivial ones, which can easily also be generated by simply searching for the artist's name. While one could argue that it is rather restrictive not to allow songs of the same artist as the query song in the recommendation list, the most promising way to overcome this problem is to have two different browsing modes: one to explore the music space and another one to further explore a specific artist.

As only non-trivial musical similarities are interesting for recommendation, it is important for proper evaluation of music recommender systems to ensure that songs from the same artists are not in both training and test sets for classification-scenario based quality measures. For query-scenario based quality measures songs of the same artist as the query song should be removed from the result set prior to the evaluation. In both cases artist information is used for filtering during the evaluation process. For this reason this filtering operation is called *artist filter* and has been initially proposed by Pampalk et al. [Pampalk et al., 2005]. In the following section the influence of artist and album effects on the evaluation results is discussed.

3.3.1 Artist Filter

The artist effect has been first observed by Pampalk [Pampalk, 2010] on rather small datasets. Some researchers speculated that artist and album effects would be less sever on large datasets. Recently, Flexer et al. [Flexer and Schnitzer, 2009] could show on a large scale dataset, that both artist and album effects can be observed even on large datasets. Their results indicate that not using an artist filter yields very over-optimistic results, when nearest neighbor quality measures are used. They could also show that the database size clearly influences genre-classification-based quality measures. Small datasets are too optimistic if no or only an album filter is used, but they are somewhat too pessimistic if an artist filter is applied. Thus, in this thesis a conservative approach is taken and an artist filter is used wherever possible, in such a way that the obtained results are rather pessimistic.

3.3.1.1 Influence of Artist and Album Effects on Evaluation Metrics

Not all quality measures are equally influenced by artist and album effects. To illustrate the different influences of the artist effect on the quality measures an experiment was set up. A classic frame-level similarity algorithm, the Single Gaussian (SG) approach (see section 4.2.2), is used to compute song similarities. The k-NA, the k-NN CA and the R-Prec quality measures (see 3.2.1) are computed for both cases, with and without artist filter. For nearest neighbor based quality measures the neighborhood size is varied from 1 to 50 neighbors.

Figure 3.2 visualizes the result. The k-NA is extremely influenced by the artist filter. The difference is extreme for small neighborhood sizes for both quality mea-

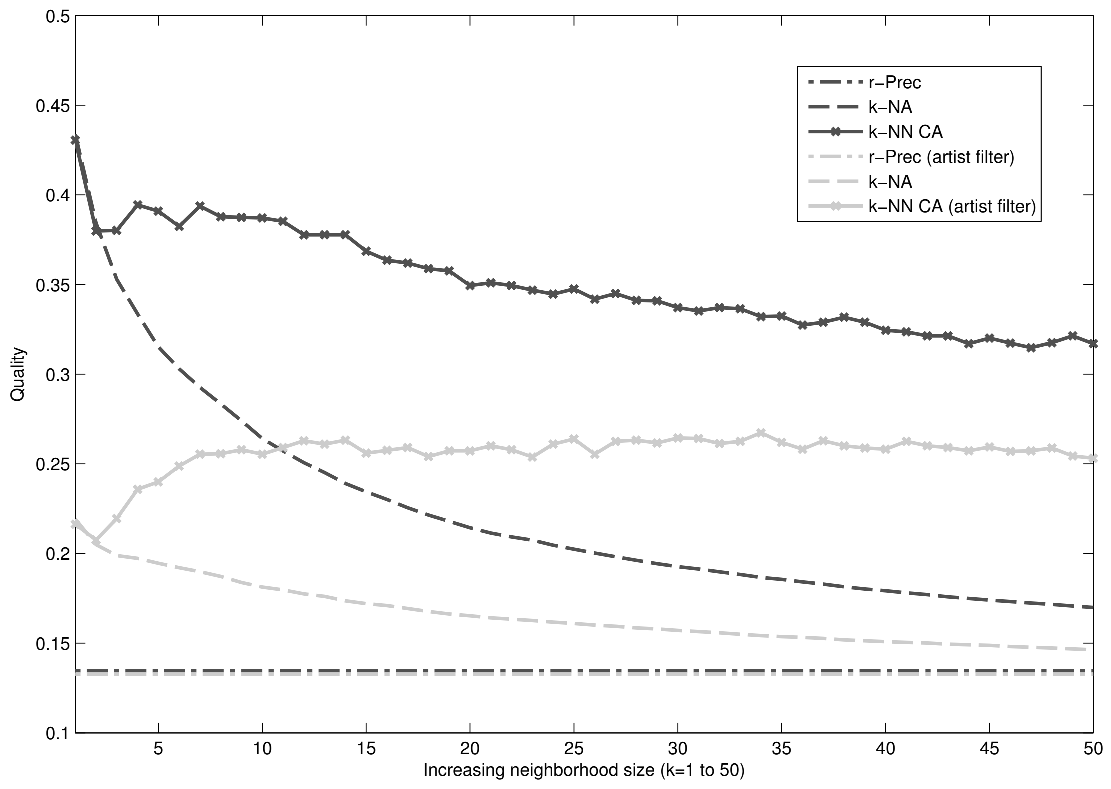


Figure 3.2: Comparison of the influence of an artist filter on different quality measures for varying neighborhood size k.

sures, but decreases with growing neighborhood size. As the R-precision is not influenced by the neighborhood size, it stays the same for a varying k. Interestingly, the R-precision is not much influenced by the artist filter. To obtain the classification results for the k-NN classifier (k-NN CA) a leave-one-out cross-validation was performed. Without using an artist filter, the k-NN CA is decreasing with increasing neighborhood size, while it is increasing if an artist filter is used. This behavior of the k-NN classifier can be explained by the fact that in the case that no artist filter is used, the first nearest neighbors belong to the same artist and are consequently reliable estimates of the genre. However, this is a rather untypical behavior of the k-NN classifier, as in a typical classification task the classification accuracy would increase with increasing neighborhood size till the optimal neighborhood size is reached and would then decrease with a further increasing neighborhood size. This untypical behavior of the k-NN classifier can even be used to detect the presence of a strong artist effect (see 5.5). For example if the 1-NN classifier outperforms all other k-NN classifiers, then it is rather likely that the obtained results are too optimistic due to the presence of an artist or album effect.

Independent of the neighborhood size the obtained quality measures significantly differ depending on the application of an artist filter. Especially nearest neighbor based quality measures are extremely influenced by the artist effect. However, nearest neighbor based evaluation criteria are important as the nearest neighbors of a query song are the songs that would actually be recommended by a system. On the one hand, the influence of an artist filter is decreasing with increasing neighborhood size, therefore it is advisable to use a large neighborhood size for evaluations. On the other hand, the neighborhood size used for evaluation should not exceed a plausible recommendation list size. Neighborhood sizes in between 5 to 20 seem therefore most appropriate. Still the choice of an appropriate neighborhood size also depends on the dataset and should never exceed the minimum number of artists per genre, if an artist filter is used.

3.3.1.2 Genre Prediction based on Artist Information

To further demonstrate the strong relation between artist and genre information another experiment was conducted. In this experiment it is shown that one can obtain high classification accuracies for nearest neighbor quality measures, without using any content-based techniques, but just using meta information like artist names, song names or album names. One can assume that this information is in general available for any song, either because it is stored in the file itself — e.g., in the id3 tags — ,or encoded in the filename, or it can be obtained by using audio fingerprinting techniques to identify any anonymous file. In the conducted experiment solely the filenames are analyzed. Dataset *"103-Artists"* is used because the filenames often, but not always contain artist, song or album names like one can expect in a typical user collection. Then song similarities are estimated by estimating string similarities on the filenames. As a measure of the similarity of two filenames we use a string similarity algorithm that is based on the *longest common substring* of the two strings. Given two string $s1$ and $s2$ the string similarity measure that is used is computed by

$$d = \frac{\text{length}(\text{LCS}(s1, s2))}{\min(\text{length}(s1), \text{length}(s2))}, \tag{3.6}$$

where LCS is a function that returns the *Longest Common Substring* of both strings.

Table 3.3 summarizes the obtained quality measures for this string similarity algorithm (LCS) for both cases, with and without artist filter. For comparison also the classification accuracies obtained by the classic Single Gaussian (SG) approach summarized in table 3.3. The baseline values are obtained by randomly choosing the nearest neighbor songs to a given query song. Interestingly, although the LCS algorithm has no information about the musical content of a song, classification accuracies far above the baseline and outperforming those of the Single Gaussian approach are obtained when no artist filter is used. By applying an artist filter, the classification accuracies of LCS approach radically drop and hardly outperform the baseline. Obviously, a song's genre can be predicted with high accuracy solely based on the artist information. However, the song similarities and the genre predictions of the string based similarity approach are obviously only based on the artist information. Using an artist filter then reveals that the string based approach — as expected — cannot identify any more complex and interesting relations among songs. In contrast for the Single Gaussian approach a significant decrease in classification performance can be observed — also expected —, but the classification results after artist filtering are still far above the baseline. As it is important in the context of music recommendation to identify interesting and no trivial relationships among songs, one can see from this experiment that using an artist filter is a must-do with respect to the evaluation of music recommender systems. In the next section the concept of an artist filter is further extended to an even more restrictive filter, the *portfolio filter*.

Longest Common Substring (LCS)				
indicator	**without af**	**Baseline**	**with af**	**Baseline**
1-NA	0.7607	0.0618	0.0769	0.0470
5-NA	0.7233	0.0595	0.0470	0.0464
10-NA	0.6772	0.0603	0.0401	0.0470
15-NA	0.6246	0.0581	0.0470	0.0456
20-NA	0.5752	0.0588	0.0380	0.0462
1-NN CA	0.7607	0.0618	0.0769	0.0470
5-NN CA	0.8180	0.0691	0.0376	0.0474
10-NN CA	0.8192	0.0659	0.0303	0.0511
15-NN CA	0.8074	0.0704	0.0258	0.0479
20-NN CA	0.7988	0.0695	0.0213	0.0425
R-Prec	0.1507	0.0414	0.0544	0.0367

Single Gaussian (SG)				
indicator	**without af**	**Baseline**	**with af**	**Baseline**
1-NA	0.7677	0.0618	0.2793	0.0470
5-NA	0.6741	0.0595	0.2658	0.0464
10-NA	0.6058	0.0603	0.2554	0.0470
15-NA	0.5515	0.0581	0.2473	0.0456
20-NA	0.5108	0.0588	0.2398	0.0462
1-NN CA	0.7677	0.0618	0.2793	0.0470
5-NN CA	0.7333	0.0691	0.2986	0.0474
10-NN CA	0.7264	0.0659	0.3104	0.0511
15-NN CA	0.7072	0.0704	0.3174	0.0479
20-NN CA	0.6908	0.0695	0.3182	0.0425
R-Prec	0.2321	0.0414	0.1750	0.0367

Table 3.3: Comparison of artist filtered and not artist filtered quality indicators for a simple string similarity based approach (LCS) and the classic Single Gaussian (SG) approach on dataset “103-Artists”.

3.3.2 Portfolio Filter

The portfolio effect is an undesired effect for example known in news recommendation. There the recommendation lists often contain identical or nearly identical news messages only. With respect to music recommendation there exists a similar effect. Often all the recommended songs, although not from the same artist as the query song (which can be ensured using an artist filter), are songs of a single artist only. Such a recommendation list definitely lacks in diversity. To increase the diversity of recommendation lists in this thesis it is proposed to use a *portfolio filter*. While an artist filter ensures that songs in the recommendation list are not from the same artist as the query song, a portfolio filter ensures that there is only one song per artist in each recommendation list. Thus, a portfolio filter is even more restrictive than an artist filter and forces content-based recommender systems to increase the diversity of the recommendations. In chapter 5.5 the impact of the portfolio effect on the evaluation result will be investigated.

3.3.2.1 Portfolio-filtered Datasets

An easy way of realizing an artist filter or even a portfolio filter is to construct a dataset where only one single song per artist is in the dataset. Using this approach no special evaluation procedures are necessary. Additionally, the generalization capabilities of the methods are better evaluated, as only non trivial similarity relationships exist in such a dataset. Furthermore, such an approach also leads to reduced database sizes and speeds up the evaluation time. The only drawback of realizing an artist or portfolio filter this way is that datasets containing many different artists are more difficult to compile. For the evaluations conducted in this thesis one such special dataset was compiled, the *"Unique"* dataset (see 3.2.3), which contains only one song per artist.

The next section presents the author's efforts to also make results of automatic genre classification comparable to human genre classification performance to be able to judge how state-of-the-art automatic classification algorithms compare to humans.

3.4 Comparison of Human, Automatic and Collaborative Classification

Although genre classification datasets help to assess the quality of music recommender systems, genre classification by itself does only allow to assess the relative performance of systems to each other, but does not provide any information on how a system performs compared to human judgment. To make the classification accuracies obtained on a genre classification dataset comparable to the performance of human beings, a listening experiment was designed. The results of this listening experiment and the comparison of the performance of humans and automatic methods are presented in the following subsections. The next subsection starts with a review of related work on this topic.

3.4.1 Related Work

In general genre as an evaluation criterion is a well-discussed topic [Ellis et al., 2002] [Geleijnse et al., 2007] [Sordo et al., 2008] [McKay and Fujinaga, 2006] [Craft et al., 2007] and it is broadly accepted in Music Information Retrieval (MIR) as an evaluation criterion for content-based systems (see section 3.2). As a consequence there exist numerous publications focusing on comparing automatic systems to each other using genre information. There also exists some scientific work on evaluating the human abilities to classifying music into genres. Most notably Gjerdingen et al. in [Gjerdingen and Perrott, 2008] showed that humans are very fast at classify music into genres. About 300ms of audio are enough for humans to come up with the same categorization decision as with 3000ms of audio. Bella et al. in [Bella and Peretz, 2005] investigated the human ability to classify classical music into sub-genres. Furthermore, Guaus et al. [Guaus and Herrera, 2006] study the effect of rhythm and timbre modifications on the human music genre categorization ability. They find that timbre features provide more genre discrimination power than rhythm.

However, there exists little work on **comparing** automatic to human performance on the same genre classification task. In [Soltau et al., 2010] Soltau et al. mentioned that the genre confusions observed in a listening experiment are similar to those of a proposed automatic system, but no evaluation to directly compare human to automatic performance was conducted. The only work that really focuses on a comparison of human to automatic classification performance is the work of Lippens et al. [Lippens et al., 2004] and dates back to 2004. In [Lippens et al., 2004], a listening experiment is conducted were 27 human listeners manually classified a collection of 160 songs (the "MAMI dataset"), into 6 possible genres by listening to 30 seconds excerpts. The average performance of the participants (76%) is then compared to an automatic classification approach with a classification performance of 57%, and the baseline accuracy (26%). Unfortunately, the MAMI dataset and the survey data are not publicly available. To be able to also compare state-of-the-art systems to human classification, a listening experiment quite similar to the one presented in [Lippens et al., 2004] was conducted.

3.4.2 The Listening Experiment

In this listening experiment 24 persons were asked to do exactly the same task the machine was asked to solve, namely to categorize a set of songs into 19 genres. The participants of this survey were aged between 20-40 and most of them had no specific musical background, but can be characterized as typical mainstream music consumers. The songs were drawn randomly from the *"1517-Artists"* dataset (see section 3.2.3) in such a way that each genre is represented by 10 songs. Details on the resulting dataset, called *"Annotated"*, can be found in Appendix A. While it seems that just selecting 10 songs per genre is at the lower bound for a descriptive subset of a genre, the number of songs that can be used in such a listening experiment is of course limited by the available human resources. With respect to the conducted listening experiment many of the participants reported that it took them many hours to complete the survey and far longer than expected.

Comparing the conducted listening experiment presented here to the listening experiment in [Lippens et al., 2004] there are some important differences in the data, the design of the experiment, and the analysis of the results:

- **Unique Artists**
  To prevent artist effects and album effects [Flexer and Schnitzer, 2009], no two songs by one and the same artists are in the dataset used for the listening experiment. This is very important as artist and album effects can have a huge biasing influence on the obtained classification accuracies, especially on small datasets.
- **Number of Genres**
  The number of genres (19) in our listening experiment is significantly larger, and the musical scope is broader than in the MAMI dataset.
- **Equal number of tracks per genres**
  Each genre is represented by 10 representative songs, making this a balanced classification task that is not biased towards a popular, dominating genre like, e.g., *"Pop&Rock"*.
- **Explicit Genre Annotations**
  There exists a ground-truth genre label per song that has been assigned by the artists that produced the songs via the music platform. The genre categories are the same as used by the music platform[7].
- **Publicly Available Data**
  The music files used in the presented experiment and the genre votes obtained through the listening experiment are both publicly available.[8] This will allow others to compare other methods not presented here to human performance in the future.
- **Collaborative Result**
  In section 3.4.3.3 the votes of the subjects are used to collaboratively estimate a song's genre. Thus, we are able to also compare the collaborative result of all subjects to both individual results as well as automatic classification systems.

[7]music.download.com
[8]www.seyerlehner.info

It is important to note that it is not claimed that the genre annotations of this dataset are particularly correct and we are aware that there might even exist inconsistencies in this genre taxonomy. However, this is true for any dataset as genre and genre taxonomies by definition are ambiguous and inconsistent. Still, it is important to see that for comparative evaluations as presented here annotation errors are not crucial since all evaluated approaches have to deal with the same annotation errors. Furthermore, with respect to genre inconsistencies it is proposed in section 3.4.4 to use so-called *user scores* as evaluation criteria, which allow to account for existing genre ambiguities.

The experiment was carried out as follows: Each participant was instructed to move the 190 anonymized full-length audio files into a set of folders representing the 19 genres, plus an extra folder *"other"* in case they had no idea what genre a song might belong to. Then a list of the files in the directory structure representing the genres was generated by a script and returned by each subject via e-mail. Finally, these files were parsed to obtain the votes of each individual.

3.4.3 Comparison of Approaches

3.4.3.1 Human Classification

The collected information from the listening experiment is represented as a set T of tuples $t = (u_t, s_t, \hat{g}_t, g_t)$, where u_t (1 to 24) identifies the participant and s_t (1 to 190) the rated song. The ground truth genre of the song s_t is denoted $g_t \in G$, where G is the set containing the 19 ground truth genres. $\hat{g}_t \in G^+$ represents the genre predicted by participant u_t. G^+ is the set of genres plus the *"other"* category. The classification accuracy of subject u with respect to the given ground truth annotation is then given by

$$acc_u = \frac{\sum_t^{\{t \in T | u_t = u\}} \hat{g}_t == g_t}{|\{t \in T | u_t = u\}|} \tag{3.7}$$

A look at Figure 3.3 shows that there is a huge variation in the performance of individual participants. Obviously the individual results heavily depend on the

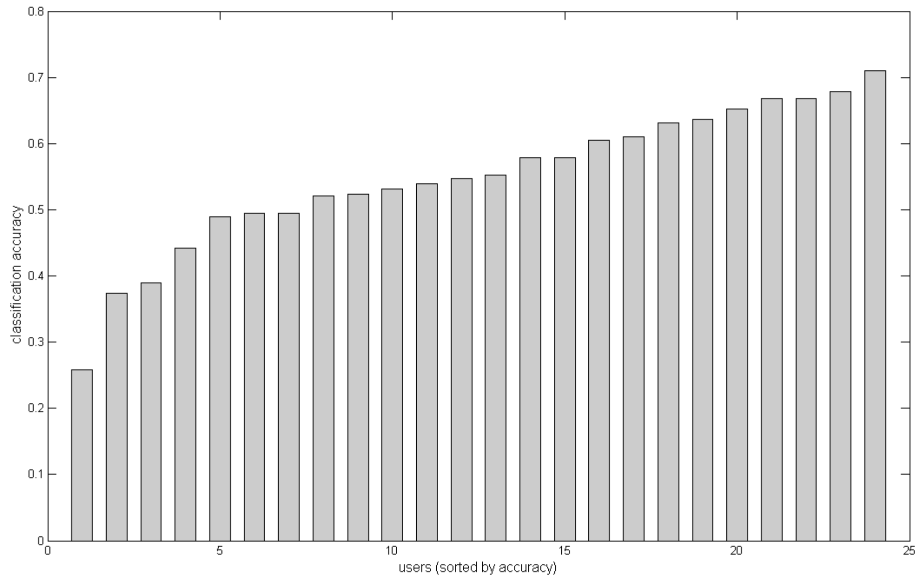
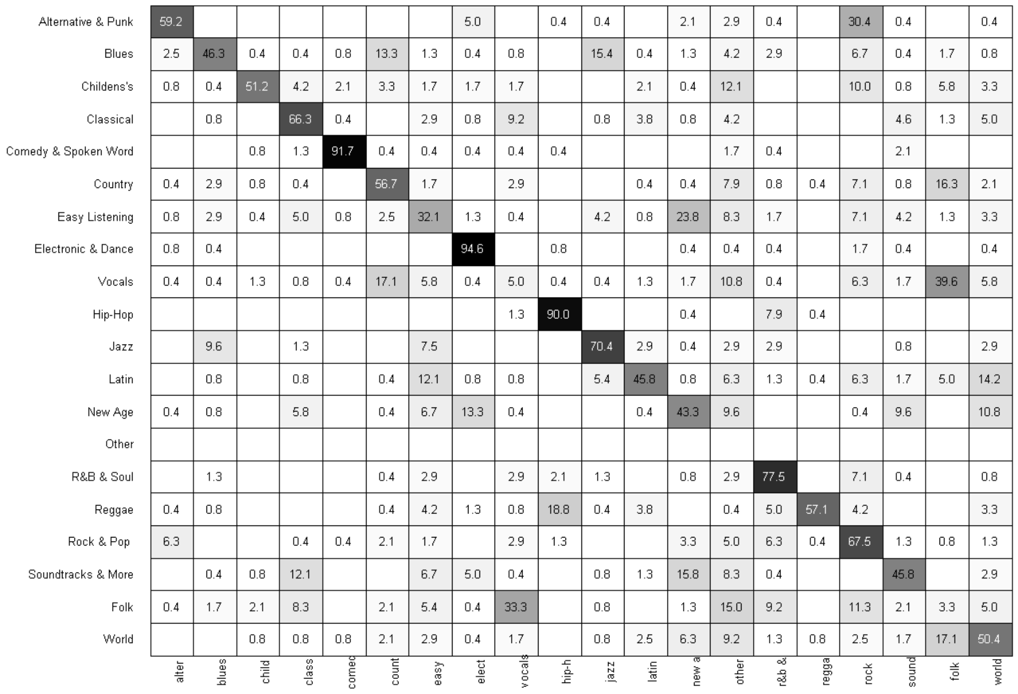


Figure 3.3: Ordered classification accuracies of the participants.

musical knowledge of the individuals. The worst participant exhibits a classification accuracy of 26%, which is still far better than the baseline (guessing), which would be 5%. The classification rate of the best individual is 71%. The average classification accuracy obtained by the participants is 55%, the median is also 55%. Figure 3.3 visualizes the classification accuracies achieved by the individual participants sorted from the worst to the best participant.

Aggregating the individual results of all users yields the overall classification result. Figure 3.4 shows the confusion matrix with respect to the ground truth. Altogether 55% of all song-genre assignments of the participants were correct. However the performance depends on the genre. While some genres seem to be well-defined (e.g. *"Comedy&Spoken Word"*, *"Electronic&Dance"*, *"Hip-Hop"*), there is almost no agreement among the participants for the genres *"Folk"* and *"Vocals"*. For the other genres the participants agree to a certain extent. The most significant genre confusions are *"Folk"* - *"Vocals"*, *"Alternative&Punk"* - *"Rock&Pop"*, *"EasyListening"* - *"NewAge"*, *"Country"* - *"Folk"*, *"Blues"* - *"Jazz"*, *"Reggae"* - *"Hip-Hop"*



	alter	blues	child	class	comec	count	easy	elect	vocals	hip-h	jazz	latin	new a	other	r&b &	regga	rock	sound	folk	world
Alternative & Punk	59.2							5.0		0.4	0.4		2.1	2.9	0.4		30.4	0.4		0.4
Blues	2.5	46.3	0.4	0.4	0.8	13.3	1.3	0.4	0.8		15.4	0.4	1.3	4.2	2.9		6.7	0.4	1.7	0.8
Childens's	0.8	0.4	51.2	4.2	2.1	3.3	1.7	1.7	1.7			2.1	0.4	12.1			10.0	0.8	5.8	3.3
Classical		0.8		66.3	0.4		2.9	0.8	9.2		0.8	3.8	0.8	4.2				4.6	1.3	5.0
Comedy & Spoken Word			0.8	1.3	91.7	0.4	0.4	0.4	0.4	0.4				1.7	0.4			2.1		
Country	0.4	2.9	0.8	0.4		56.7	1.7		2.9			0.4	0.4	7.9	0.8	0.4	7.1	0.8	16.3	2.1
Easy Listening	0.8	2.9	0.4	5.0	0.8	2.5	32.1	1.3	0.4		4.2	0.8	23.8	8.3	1.7		7.1	4.2	1.3	3.3
Electronic & Dance	0.8	0.4						94.6		0.8			0.4	0.4	0.4		1.7	0.4		0.4
Vocals	0.4	0.4	1.3	0.8	0.4	17.1	5.8	0.4	5.0	0.4	0.4	1.3	1.7	10.8	0.4		6.3	1.7	39.6	5.8
Hip-Hop									1.3	90.0			0.4		7.9	0.4				
Jazz		9.6		1.3			7.5				70.4	2.9	0.4	2.9	2.9			0.8		2.9
Latin		0.8		0.8		0.4	12.1	0.8	0.8		5.4	45.8	0.8	6.3	1.3	0.4	6.3	1.7	5.0	14.2
New Age	0.4	0.8		5.8		0.4	6.7	13.3	0.4			0.4	43.3	9.6			0.4	9.6		10.8
Other																				
R&B & Soul		1.3				0.4	2.9		2.9	2.1	1.3		0.8	2.9	77.5		7.1	0.4		0.8
Reggae	0.4	0.8				0.4	4.2	1.3	0.8	18.8	0.4	3.8		0.4	5.0	57.1	4.2			3.3
Rock & Pop	6.3			0.4	0.4	2.1	1.7		2.9	1.3			3.3	5.0	6.3	0.4	67.5	1.3	0.8	1.3
Soundtracks & More		0.4	0.8	12.1			6.7	5.0	0.4		0.8	1.3	15.8	8.3	0.4			45.8		2.9
Folk	0.4	1.7	2.1	8.3		2.1	5.4	0.4	33.3		0.8		1.3	15.0	9.2		11.3	2.1	3.3	5.0
World			0.8	0.8	0.8	2.1	2.9	0.4	1.7		0.8	2.5	6.3	9.2	1.3	0.8	2.5	1.7	17.1	50.4

Figure 3.4: Confusion matrix of the classifications of the experiment with respect to the ground truth annotation. Entry i, j is the percentage of user votes that predicted class j when the true class was i.

and *"Latin"* - *"EasyListening"* and vice versa. These confusions indicate genre ambiguities, but can also be interpreted as some sort of genre similarities. Also, many genre pairs are never or extremely rarely confused, which implies that it is very easy for humans to distinguish these genres. Based on the user votes one can define the genre-song voting matrix $V = (v_{g,s})$, where $v_{g,s}$ denotes the number of times the participants voted for genre g given song s:

$$v_{gs} = \sum_{t}^{\{t \in T | s_t = s\}} \hat{g}_t == g \tag{3.8}$$

The genre-song voting matrix is visualized in figure 3.6. One can even visually see that the majority of the participants agree with the ground truth information for most of the songs. In contrast to the confusion matrix, the genre-song voting matrix visualizes the classification result for each song separately and is a compact representation of the results of the listening experiment. To further analyze the votes one can define the number of different genres $D(s)$ the participants have assigned to a specific song s:

$$D(s) = \sum_{g}^{G^+} v_{gs} > 0 \tag{3.9}$$

Figure 3.5 (left) shows a histogram of the number of different genres $D(s)$ the user voted for. Although there are 20 options to choose from, in general the participants did not vote for more than 8 different genres. This indicates that some genres are not relevant at all for some songs. Furthermore one can identify the most frequently estimated genre, the second most frequently estimated genre and so on, for each song. Then the number for votes for the k (1 to 20) most frequently estimated genre over all songs can be aggregated. The percentage of the accumulated votes relative to the total number of votes is visualized in figure 3.5 (right). Consistent with the histogram in figure 3.5, all votes are within the 12 most frequently estimated genres. In general there exists a strong consensus among the participants on a song's genre. The most frequently predicted genre for each song is responsible for 64% of all votes. The two most frequently predicted genres of each song, together represent 80% of all votes (see figure 3.5). Therefore, one can conclude that the majority of the participants strongly agree on just one or two possible genre assignments for most of the songs.

3.4.3.2 Automatic Classification

To compare human to automatic classification performance five different automatic classification methods are used. The choice of the evaluated approaches contains classical and state-of-the-art systems. Only complete genre classification systems as proposed in the literature are evaluated. Thus, the evaluated systems extract different feature sets and are based on different classification approaches. Two of the evaluated classification systems (SG-NN and RTBOF-NN) are based on nearest neighbor classifiers. The other three algorithms (GT-SVM, BLF1-SVM,

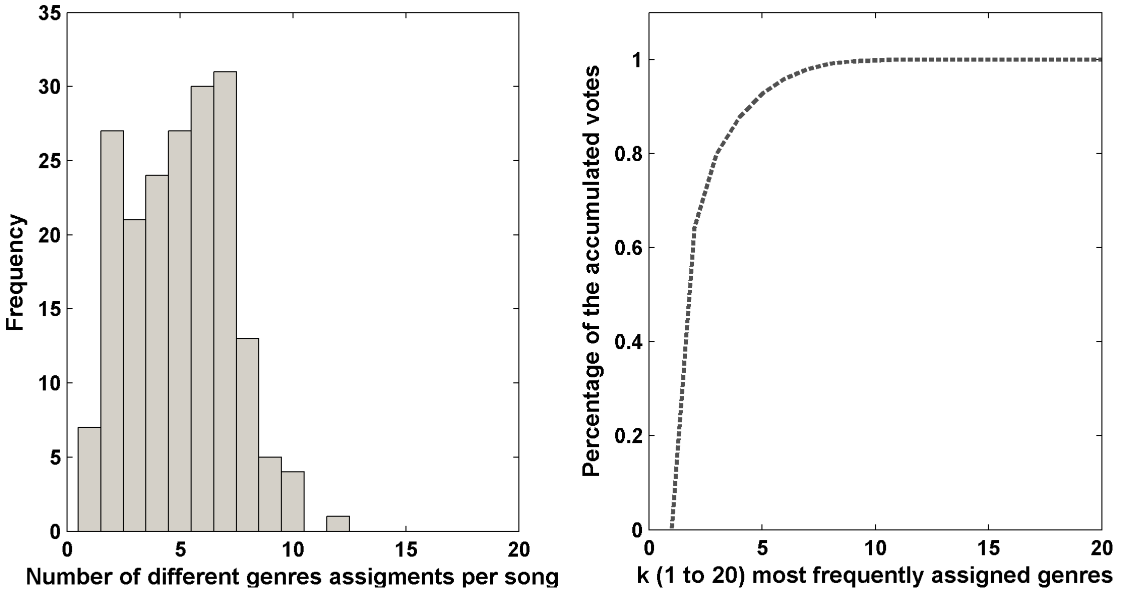


Figure 3.5: Histogram of the number of different genres per song the participants have voted for (left) and percentage of the accumulated number of votes for the k most frequently assigned genres per song (right).

BLF2-SVM) are based on a support vector machine classifier. The reported classification accuracies are obtained via leave-one-out cross-validation. The automatic classification methods are briefly described below.

3.4.3.2.1 Single Gaussian (SG-NN) The *Single Gaussian Nearest Neighbor Classifier* (SG-NN) is based on the classic Bag of Frames (BoF) approach (see 4.2). Each song is modeled as a distribution of Mel Frequency Cepstrum Coefficients (MFCCs). A single multivariate Gaussian distribution is used to model the distribution of MFCCs of a song. To identify the nearest neighbors the Kullback-Leibler (KL) divergence between two models is computed. The SG algorithm is the by now "classic" and de-facto standard method to compute timbral similarity.

3.4.3.2.2 Rhythm Timbre Bag of Features (RTBOF-NN) The *Rhythm Timbre Bag of Features Nearest Neighbor Classifier* (RTBOF-NN) is a state-of-the-art music similarity measure proposed by Pohle et al. in [Pohle et al., 2009]. This measure ranked first in the MIREX 2009 music similarity and retrieval task and has proven to be statistically significantly better than most of the participating algorithms. In contrast to the classic Single Gaussian approach this RTBOF-NN Classifier reflects the current state-of-the-art in nearest neighbor music classification.

3.4.3.2.3 Block-Level Feature (BLF-SVM) The Block-Level Feature Support Vector Machine approach (BLF-SVM) is a genre classification algorithm based on block-level features (see chapter 5). An earlier version of this algorithm only using the Spectral Pattern, the Delta Spectral Pattern and the Fluctuation Pattern participated in the MIREX 2009 Audio Genre Classification task and took rank 14 out of 31[Seyerlehner and Schedl, 2009]. However, no statistically significant difference to the winning algorithm was found. This approach will be denoted BLF1-SVM. Additionally, we also evaluate an improved variant of this algorithm, which we call BLF2-SVM here. This algorithm includes three additional block-level features (Spectral Contrast Pattern, Correlation Pattern and Variance Delta Spectral Pattern). This improved variant outperformed most of the participating algorithms on different train-test-set task in the MRIEX 2010 competition and is one of the state-of-the-art methods in automatic genre classification.

3.4.3.2.4 Marsyas (MARSYAS-SVM) The Marsyas (Music Analysis, Retrieval and Synthesis for Audio Signals) framework[9] is an open source software that can be used to efficiently calculate various audio features. For a detailed description of the extracted features we refer to [Tzanetakis and Cook, 2002]. This algorithm has participated in the MIREX Genre Classification task from 2007 onwards, and the features as well as the classification approach have been the same over the years. The framework is used to extract the features exactly as for the MIREX 2009 contest (MARSYAS version 0.3.2). Then the WEKA Support Vector Machine implementation to perform cross-validation experiments is used. This method

[9]http://marsyas.info

is closest to the automatic approach by Lippens et al. [Lippens et al., 2004] and should help to make our experiment more comparable to this previous experiment.

3.4.3.3 Collaborative Classification

In this subsection two straight-forward collaborative classification approaches (CV and CSS-NN) based on the users' aggregated votes are presented.

3.4.3.3.1 Collaborative Voting (CV) The Collaborative Voting (CV) approach is simple. The genre most participants have voted for is the predicted genre of a song. This method basically combines the individual classification results of the participants following the majority rule like a meta-classifier.

3.4.3.3.2 Collaborative Filtering (CF-NN) The Collaborative Filtering Nearest Neighbor Classifier (CF-NN) is related to an item-based collaborative filtering approach. Each song is represented by its voting profile, which corresponds to the column vector of a song in the genre-song voting matrix (see figure 3.6). One can then derive song similarities by comparing the voting profiles of the songs. To compare song profiles the *city-block* distance (l_1 *norm*) was used in our experiments. The song similarity information can then be used to perform nearest neighbor classification.

3.4.3.4 Comparison

In figure 3.7 the classification results of the automatic methods, the collaborative approaches and the individual results of the participants are visualized together, sorted according to the achieved accuracy. Clearly, the content-based approaches perform worse than most of the participants, whereas the collaborative approaches achieve high classification accuracies and outperform most of the participants. Comparing the best content-based approach (BLF2-SVM) to the best collaborative approach (CF-NN) it turns out that the latter achieves almost double the

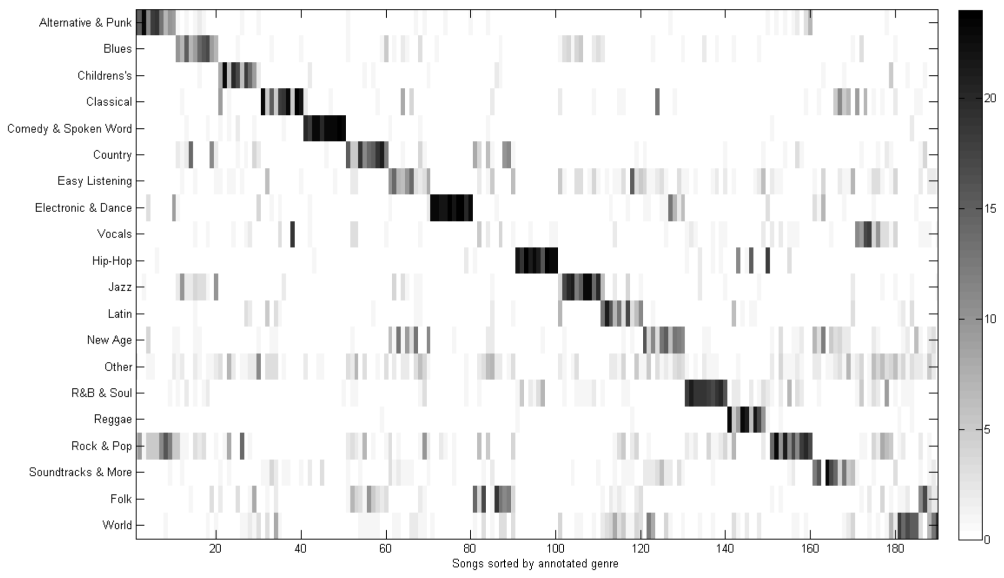


Figure 3.6: Visualization of the genre-song voting matrix. Tracks are sorted according to the ground truth genre.

classification accuracy of the content-based approach. Taking a look at the various content-based methods, we can see that there exist clear differences. The classical timbral similarity measure performs worst, just outperforming the worst participant. The classic MARSYAS-SVM approach does not perform much better. Both recent methods RTBOF and BLF2-SVM show an improvement in classification accuracy over the “classic” approaches. This indicates that the improvements in automatic classification reduced the gap between human and automatic classification, but still there exists a difference of about 10 percentage points between the best automatic method and the average human participant. Furthermore, based on the obtained results we can define an upper bound on the achievable classification accuracy for automatic methods on this dataset. Clearly because of inconsistencies of the classification taxonomy and possible annotation errors none of the evaluated methods will ever reach perfect classification accuracy. However, as all evaluated methods have to deal with these problems the classification re-

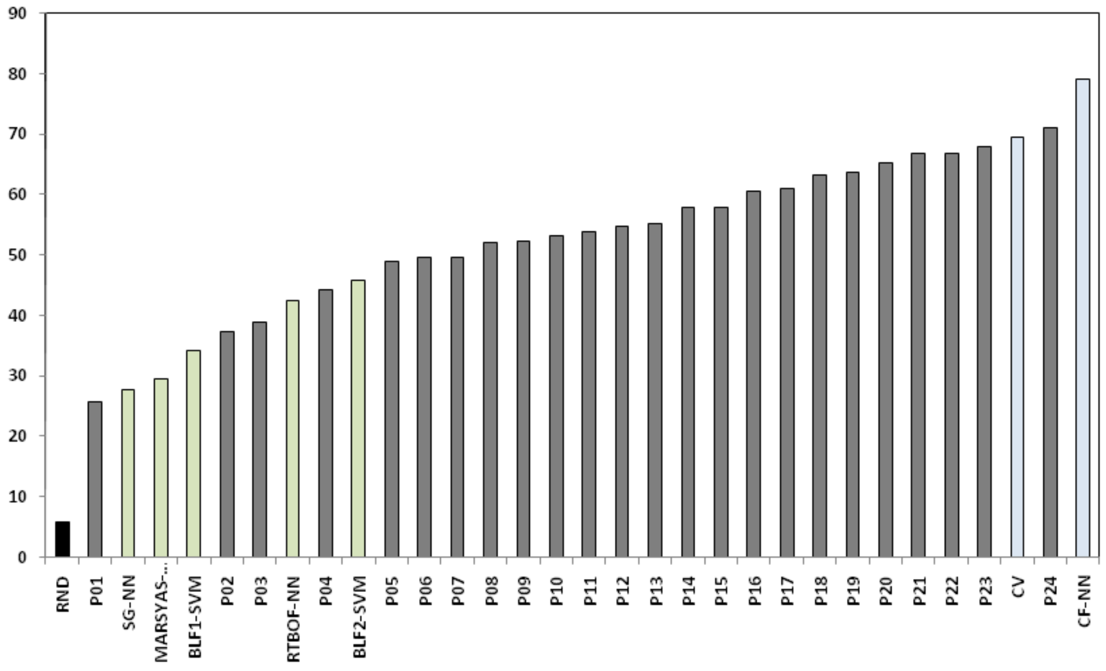


Figure 3.7: Comparison of classification results of the individual participants, automatic methods and the collaborative approaches.

sult of the CF-NN approach can be interpreted as an upper bound for automatic methods on this dataset.

3.4.4 Evaluation based on User Data

One of the main disadvantages of using the classification accuracy as evaluation criterion is that such experiments heavily depend on the quality of the ground truth annotations. To improve the quality of the ground truth one can of course ask an expert to define the genre annotations, but still the evaluation would just depend on a single opinion and, as already pointed out, there will always exist some annotation errors due to the inconsistency of the genre taxonomy itself.

To overcome these limitations it is proposed to perform a user centric evaluation by aggregating the collected genre votes of the participants of the listening experiment. Thus, the ground truth is no longer based on a single opinion, but on the aggregated opinions of all the participants regarding the genre affinity of a given song. This way the obtained data from the listening experiment can not only be used to make automatic classification methods comparable to human classification performance, but this information can also be used to account for genre ambiguities whenever genre classification is used in an evaluation, as already proposed in [Craft et al., 2007] and [Lippens et al., 2004]. The basic idea for such a quality measure is straight-forward: *If even humans are unsure about a genre label then it will be hard for the machine to get the label right.*

To reflect these uncertainties of the genre annotations in a quality measure, a *user score* is defined similarly to [Lippens et al., 2004]. A user score measures the agreement of the predictions of an automatic method with the genre assignments of the humans participating in the listening experiment. Thus, any algorithm can collect points for each song s in the dataset according to the agreement with the user votes. In particular, for each song $s \in S$ the classification of the algorithm into genre $\hat{g}_s \in G$ is rated by the number of times this genre was voted for ($v_{\hat{g}_s,s}$) relative to the number of times the participants voted for the most frequently predicted genre ($\max(\{v_{g,s}|g \in G\})$).

$$\text{US1} = \frac{1}{|S|} \sum_{s}^{s \in S} v_{\hat{g}_s,s} / \max(\{v_{g,s}|g \in G\}) \tag{3.10}$$

Extending the idea in [Craft et al., 2007], another straight-forward definition of a *user score* — this score is denoted US2 — is to take the number of collected points relative to the maximum number of points one can obtain on the dataset.

$$\text{US2} = \sum_{s}^{s \in S} v_{\hat{g}_s,s} / \sum_{s}^{s \in S} \max(\{v_{g,s}|g \in G\}) \tag{3.11}$$

The difference of the two scores is that for US1 each song contributes equally, whereas for US2 it is more important to correctly predict songs where the participants agreed pretty much on a single genre. One important advantage of both user scores is that they no longer rely on the ground truth annotation, but are solely based on the user ratings. By definition both scores are in the range between 0 and 1.

Approach	**US1**	**US2**	**CA**
BLF2-SVM	0.5615	0.5080	0.4579
RTBOF-NN	0.4352	0.3827	0.4253
BLF1-SVM	0.3672	0.3382	0.3421
MARSYAS-SVM	0.3217	0.3031	0.2953
SG-NN	0.3156	0.2791	0.2779
RND	0.0578	0.0673	0.0584

Table 3.4: Comparison of the user scores (US1, US2) and the classification accuracy (CA) obtained for the automatic approaches presented in section 3.2.

Table 3.5 summarizes the user scores and the classification accuracy for the automatic classification methods presented in section 3.4.3.2. To our knowledge this is the *first comparison* of automatic classification methods also accounting for genre ambiguities in the literature. The ranking of the analyzed algorithms is the same for all quality criteria. However, taking genre ambiguities into account clearly changes the evaluation result. For example the difference between the BLF2-SVM and the RTBOF-NN is relatively bigger for the users scores compared to the classification accuracy. An improvement of a user score over the classification accuracy reveals that the misclassified songs are not classified into an arbitrary, completely unrelated genre, but into a genre that users find similar, or tend to confuse also. This method is advocated for future evaluations of genre classifiers, whenever appropriate data are available.

3.5 Conclusions

Genre classification is a practical, cost efficient and reliable workaround to evaluate music recommender systems, which is especially well-suited for rapid prototyping as is typically done in academic research. However, there also exist some issues related to the evaluation based on genre classification. Especially, artist and album specific production effects can lead to over-optimistic quality measures. To get reliable quality estimates an artist filter has to be used and the application of the even more restrictive portfolio filter is recommended. Furthermore, the results of the conducted experiments indicate that using small neighborhood sizes for nearest-neighbor based quality measures can lead to unstable and unreliable quality estimates. It is therefore advisable to use larger neighborhood sizes. The k-NA or k-NN CA with k=10, 15 or 20 seem to be most appropriate. Furthermore, the evaluation of music recommender systems would profit by using user-centric evaluation metrics like the evaluated user-scores, as these evaluation metrics no longer depend on a single ground truth annotation, but on many aggregate user judgments.

Furthermore, some important conclusions result from the presented listening experiment. First of all, one can conclude that there is some progress with respect to automatic genre classification, as the gap between automatic methods and human classification has decreased. However, the best performing automatic method in the conducted experiment still performs about 10 percentage points worse than the average human participant. Thus, there is still a gap between automatic and human classification performance. Furthermore, it could be shown that collaborative approaches outperform both automatic methods and individual human performances. This indicates that collaboratively collecting meta-information about music, e.g., via a music platform is a very powerful method and is also the clear trend in the music business. For content-based methods this implies that they are most beneficial in situations where no other data is available. For instance, in cold start situations, or in special application scenarios where no access to collaboratively collected meta-data is possible, or as one component of a more complex hybrid music recommender system.

4 Improving Frame-Level Music Similarity Algorithms

4.1 Music Similarity Algorithms: An Introduction

Ideally a content-based music audio similarity metric should approximate the ill-defined *"sounds like"* relation for songs (e.g $Song_A$ *"sounds like"* $Song_B$). In this thesis this similarity information will then be used to generate recommendations to support browsing and exploring a given music corpus (see Chapters 1 and 3). However, approximating this *"sounds like"* relation is not a trivial task, especially since this relation depends on the individual perception of various musical aspects (e.g. instrumentation, rhythmic structure, singing voice, timbre, melody, tempo or lyrics). This chapter will focus on a specific type of music similarity algorithms often referred to as *Bag-of-Frames* (BoF) approaches[1] that extract information related to a song's timbre. Although the first variants of frame-level similarity algorithms have already been proposed in the late 90ies, this rather classic type of algorithms is still widely used for example as one component in many state-of-the-art algorithms participating in the last runs of the *MIREX Audio Music Similarity and Retrieval Task*[2]. Unfortunately, it is well-known [Aucouturier and Pachet, 2004] that for these types of algorithms there does exist a *"glass ceiling"*, beyond which no further improvement seems to be possible. However, the reason for this class ceiling is still not yet completely understood. In section 4.4 three limiting factors

[1]BoF approaches are also named *frame-level similarity algorithms* to emphasize the difference to the so-called *block-level algorithms* that will be presented in chapter 5.

[2]http://www.music-ir.org/mirex/

have been identified. Thus, in general there are still some insights to be gained into the BoF approach itself. Furthermore, the ability of BoF algorithms to generate non-trivial recommendations (see 3.3) has not yet been studied in detail and will be in the focus of this chapter.

This chapter will report on two variants of the Bag-of-Frames approach that have been proposed to analyze and better understand the BoF approaches with the ultimate goal to identify possible ways for improvements. This chapter is organized as follows: First in section 4.2 an introduction to the Bag-of-Frames approach is given. Then in the second section two questions are investigated:

- First, a vector quantization based variant of the BoF approach is presented. Based on this variant it will be analyzed if using non-parametric distribution models instead of parametric or semi-parametric models can help to improve the recommendation quality of a BoF approach.
- Secondly, to better understand which frames really make two songs sound the same from a machine's point of view a nearest neighbor density estimation based similarity algorithm is proposed. Based on the obtained results a frame-selection strategy has been developed that leads to a qualitative improvement of frame-level algorithms.

Finally, section 4.4 then concludes on the obtained results and discusses the general limitations of frame-level algorithms. The next section gives an introduction to the Bag-of-Frames approach and presents different algorithmic variants known in the literature.

4.2 The Bag-of-Frames Approach (BoF)

The general idea of the BoF approach is to model a song as the long-term distribution of local features. As local features, which are typically extracted on a frame-by-frame basis, commonly the Mel Frequency Cepstral Coefficients (see 2.4.4) are used. MFCCs are a compact representation of the spectral envelope of a short audio frame and were and are still one of the most widespread features in the MIR community. Since the spectral envelope characterizes the timbre of a short audio frame, the distribution of all local MFCC vectors is related to the overall timbral characteristic of a song.

One essential part of the BoF approach is the strategy that is used to model the distribution of the local features, because both memory requirements and runtime depend on the used distribution model. Furthermore, it seems that the type of distribution model (e.g. parametric or non-parametric) also has some influence on the recommendation quality, which will be investigated in section **??**. For this reason many different variants of the BoF approach have been proposed in the literature. The following four sections, where BoF algorithms are categorized according to the type of the distribution model, give an overview on these different variants of the BoF approach and point out their advantages and disadvantages.

4.2.1 Gaussian Mixture Models

Some of the first approaches [Logan and Salomon, 2001] to model timbral similarity were based on Gaussian Mixture Models (GMM), a semi-parametric way of modeling a distribution. Although GMM based frame-level music similarity algorithms have been intensively studied in the literature [Aucouturier and Pachet, 2004, Levy and Sandler, 2006], today semi-parametric models like GMMs are rarely used. One of the major drawbacks is the time consuming training process, which relies on the *Expectation Maximization* (EM) algorithm. The second crucial shortcoming is that comparing two distributions modeled by a GMM is not trivial at all. Often the similarity of two distribution models is measured by computing the Kullback-Leibler (KL) divergence [Kullback and Leibler, 1951]. The KL divergence is a measure of the relative entropy of two probability distributions, P and Q.

$$D_{KL}(P||Q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)}\, dx \tag{4.1}$$

However for GMMs there exists no closed form formula to compute the KL divergence. The only way to compute the KL divergence is to approximate it via

Monte-Carlo sampling of MFCC vectors from one distribution and estimating the likelihood of the sampled vectors given the other distribution, which of course is quite computationally expensive. Another popular variant to compute the distance of two distributions is the Earth Movers Distance (EMD)[Rubner et al., 1998]. Unfortunately, the EMD is quite expensive to compute as well. Altogether, it can be summarized that GMM based BoF approaches are computationally not suitable for real world applications, which leads to the development of more efficient frame-level algorithms.

4.2.2 Single Gaussian Model

To speed up the overall process various simplified distribution models were under consideration (e.g. GMMs with diagonal covariance matrices only, or with a reduced number of components). Surprisingly, it turned out that a single multivariate Gaussian distribution of MFCCs can perform as well as mixture models [Mandel and Ellis, 2005, Levy and Sandler, 2006]. This did not only reduce the size of the models themselves, but also simplified the computation of the KL divergence a lot, since a closed form exists for the KL divergence for single Gaussians. Given the parameters of two Gaussians $p(x) = N(x; \mu_p, \Sigma_p)$ and $q(x) = N(x; \mu_q, \Sigma_q)$ the KL divergence can be computed by

$$2KL(p||q) = \log \frac{\det \Sigma_q}{\det \Sigma_p} tr(\Sigma_q^{-1}\Sigma_p) + (\mu_q - \mu_p)^T \Sigma_q^{-1}(\mu_p - \mu_q) - d \tag{4.2}$$

where tr is the matrix trace and d the dimensionality. Using this closed form expression the computational costs of comparing two song models are radically reduced. Furthermore, the training process simplifies to the computation of mean and covariances over MFCC vectors, altogether resulting in a dramatically improved performance of the similarity measure. Because of the compact song model and the rather efficient overall performance the SG model is today the *de-facto* standard algorithm for computing timbral similarity.

Nevertheless, it is important to note that the *Single Gaussian* (SG) model has also some disadvantages. Altogether, the similarity computation for an entire collection is still expensive — to be more precise, it will require $(N^2-N)/2$ distance computations, because the KL distance does not fulfill the triangle inequality and therefore one cannot easily apply more powerful search strategies. It would be desirable to have a vector space model instead, because this would enable non-exhaustive search strategies like *KD-trees* or *Local Sensitive Hashing* (LSH) to identify the most similar items in a music catalog.

4.2.3 Vector Quantization Model

Vector Quantization (VQ), as a non-parametric distribution model, has not received much research attention, since parametric methods (SG or GMMs) seem to outperform models generated via VQ [Aucouturier, 2006]. Interestingly, the first frame-level algorithms were actually based on vector quantization [Pye, 2006, Foote, 1997]. Pye and Foote proposed a supervised tree-based quantization scheme to learn a decision tree from the test set which splits the MFCC feature space into maximally discriminative regions with respect to the associated genre. For each song of the training set they compute a histogram over the leaf nodes of the tree, after subdividing the MFCC vectors according to the trained tree structure. These histograms are then compared to genre histogram templates by using euclidean or cosine distance to predict the genre. Although this tree-based quantization schema allows to efficiently generate a global quantization structure, this quantization method is limited because of its supervised nature and the simple tree learning algorithm. A more appropriate way to come up with a global partitioning of the feature space are unsupervised clustering algorithms, e.g. *k-means* or *self-organizing maps*. Self-Organizing Maps (SOMs) have been proposed by [Vignoli and Pauws, 2005] and have been evaluated in [Levy and Sandler, 2006]. The SOM-VQ approach, according to the results in [Levy and Sandler, 2006], seems to perform worse than the SG or GMM variant. The *k-means* clustering algorithm has been investigated in [Aucouturier, 2006] by Aucouturier. Additionally, he also investigates a supervised variant known as *Learning Vector Quantization* (LVQ). For both variants he reports classification results about 15% less precise than GMMs. However, all these results are based on not-artist filtered classification experiments, which are useful to evaluate similarity algorithms in general, but not in the context of recommendation (see 3.3). In section **??** a comparison of parametric versus non-parametric distribution models with respect to the ability to generate non-trivial music recommendations is carried out.

4.2.4 Hidden Markov Models

Besides parametric, semi-parametric and non-parametric distribution models, also stochastic generative models have attracted some research attention in the literature. In contrast to distribution models, where the temporal sequence of the local audio features is completely ignored, generative models do not assume that the local audio features are independent, but consider their temporal relation. One of the most popular generative models are the *Hidden Markov Models* (HMM) which explicitly account for the temporal evolution of local audio features. Surprisingly, Aucouturier [Aucouturier, 2006] could show that using generative models like HMM does not improve the quality of frame-level algorithms and that generative models are at best equivalent to simpler static distribution models. This is surprising, as distribution models consider frame-permutations of the same audio signal as identical, while for example listening to an inverted (from the end to the beginning) audio track is perceptually significantly different. Moreover, comparing the HMMs of two songs is computationally expensive. To compare two models, first, a sequence of observed MFCC vectors has to be generated by each of them. Then the likelihood of observing the generated sequence by the other HMM has to be computed mutually. While more recent generative models, e.g., the *Hierarchical Dirichlet Process* [Hoffman et al., 2008] definitely reduce the computational burden, generative models seem to be more useful in the context of genre or tag classification than for music similarity estimation and are not in the focus of this chapter. In the next section the influence of the distribution model on the recommendation quality of BoF approaches is studied.

4.3 Analysis and Improvements

4.3.1 Analysis 1: *Parametric versus Non-parametric distribution models*

Previous research work [Aucouturier, 2006] on the comparison of parametric and non-parametric distribution models showed that BoF approaches using parametric distribution models achieve in general a higher genre classification accuracy than algorithms using non-parametric distribution models, if no artist filter is used. In this section it will be shown that both types of distribution models achieve the same results with respect to their ability to identify non-trivial song relations as needed for music recommendation, i.e. in case that an artist filter is applied. The main assumption is that the recommendation quality of vector quantization approaches is not limited by the type of the non-parametric modeling strategy itself, but that the achievable recommendation quality is more related to the quality of the underlying codebook. Generating codebooks of high quality via clustering all local feature vectors of all songs is unfortunately computationally extremely burdensome and is typically circumvented via random sub-sampling the local features, resulting in poor codebooks. In the following a novel Multi-Level Vector Quantization (MLVQ) schema is proposed that radically reduces the computational costs of generating high quality codebooks.

4.3.1.1 A Multi-Level Vector Quantization Approach (MLVQ)

The proposed MLVQ approach is based on Lloyd's variant [Lloyd, 1982] of the *k-means* clustering algorithm that is used to partition the overall feature space of local audio features into k quantization regions. In this variant of the BoF approach the frames of the normalized cent spectrum (see 2.4.5) serve as local audio features. The whole universe of audio features that would have to be partitioned contains a huge number of audio frames. To be more precise it contains all audio frames of all songs in an evaluation collection. From this point of view Lloyd's iterative refinement heuristic is indeed a good choice as it is known to converge very quickly.

However, for the standard k-means clustering algorithm there is no guarantee on the quality of the resulting partitioning. It depends heavily on the chosen initial vectors. To address this problem a special seeding algorithm, proposed by Arthur and Vassilvitskii [Arthur and Vassilvitskii, 2007], which is known as the *k-means++* algorithm, is used. If initialized with this seeding technique, the total error of the resulting clustering can be expected to be $O(\log k)$ worse than the optimal clustering. Thus, seeding ensures some quality guarantees on the generated global codebooks. Still, to deal with large audio collections the number of local feature vectors has to be reduced to come up with a global codebook in reasonable time.

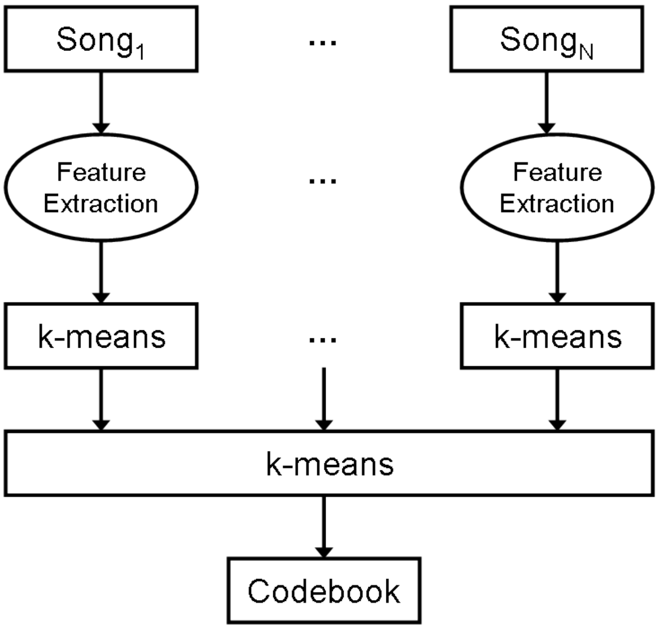


Figure 4.1: Overview of the Multi-Level Vector Quantization (ML-VQ) approach.

One way of reducing the number of feature vectors is to simply randomly subsample the overall distribution of feature vectors. Another way is to make use of Aucouturier's observation that the quality of the codebook increases with the number of songs used to generate the codebook, rather than the number of frames per song [Aucouturier, 2006]. Redundant feature vectors from a single song just increase computational costs, but do not improve the quality of the global partitioning of the feature space. Consequently, it seems to be advantageous to already remove redundant feature vectors at the song level. To do so, the *k-means++* algorithm is used to cluster the feature vectors within individual songs first and then pass these song-level cluster centers to the global codebook generation stage, where once again a *k-means++* algorithm is used to generate the final codebook. Figure 4.1 gives an overview on this process. This multi-level clustering architecture greatly reduces the computational costs, of course depending on both the number of cluster centers at the song-level and the number of cluster centers in the final codebook generation stage. Figure 4.2 shows such a codebook learned from dataset *"1517-Artists"*.

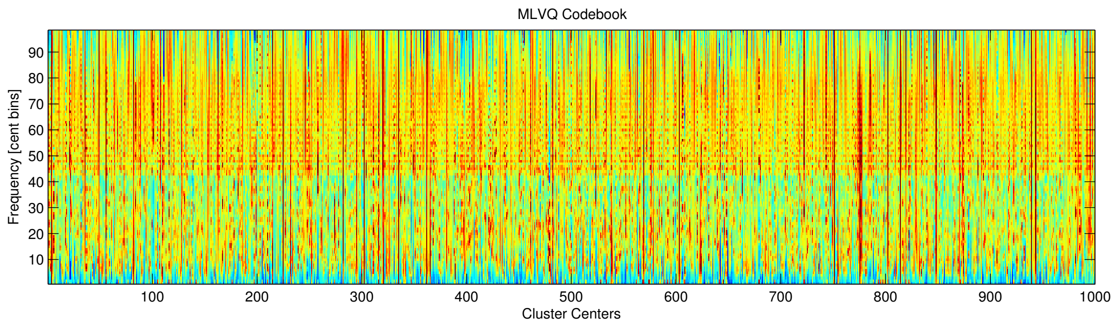


Figure 4.2: An example of a codebook generated by the MLVQ approach.

4.3.1.2 Song Model Generation

Once a general codebook has been constructed, song models based on this codebook have to be generated. A histogram H over the k quantization units (cluster centers) of the codebook is built. Each local feature vector X_j of all n local feature vectors is mapped to its closest codebook vector C_{u_j}, where u_j is the index of the closest codebook vector and is computed according to equation (4.2).

$$u_j = \arg \min_{l \in \{1,\ldots,k\}} |X_j - C_l| \tag{4.3}$$

$$H_i = \frac{1}{n} \sum_{j}^{n} u_j == i \tag{4.4}$$

For each codebook vector C_i we end up with a corresponding histogram bin H_i indicating the relative frequency of local feature vectors mapped to the i-th codebook vector, see equation (4.3). The resulting normalized histogram forms a probability distribution. Figure 4.3 visualizes the histogram models of four songs. To measure the distance between codebook vectors and the local audio features the *Manhattan* or L_1 distance is used.

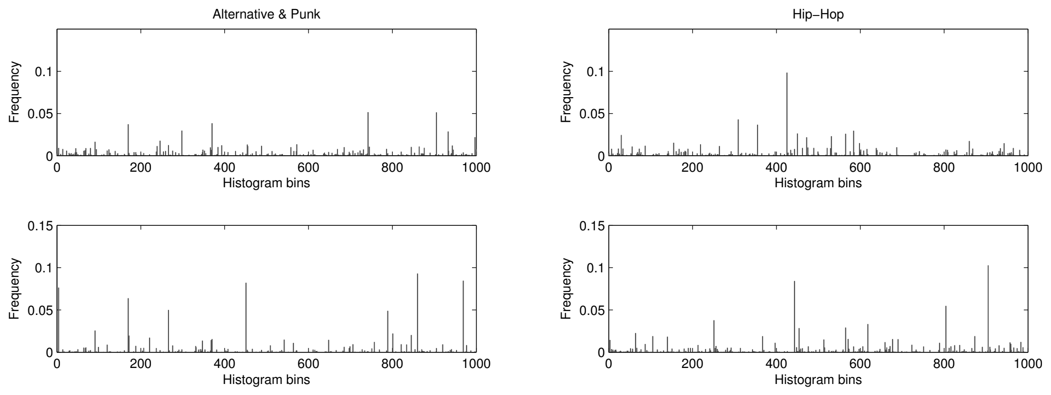


Figure 4.3: Four examples of histogram song models generated by the MLVQ approach.

4.3.1.3 Comparison

Various types of distance measures can be used to compare two histograms. In [Foote, 1997] euclidean and cosine distance are proposed. In our experiments the *histogram intersection* as introduced by Swain and Ballard for image indexing [Swain and Ballard, 1991] is used. Given a pair of histograms, I and M, each containing m bins, the histogram intersection distance is defined by equation (4.4).

$$D(I, M) = 1 - \frac{\sum_{i=1}^{m} \min(I_i, M_i)}{\sum_{i=1}^{m} M_i} \tag{4.5}$$

In the special case where the sum over the histogram bins is constant — for our probability distributions the sum over all bins is always one ($\sum_i^m I_i = \sum_i^m M_i = 1$) — the histogram intersection reduces to the *Manhattan* or L_1 distance.

$$D(I, M) = 1 - \frac{\sum_{i=1}^{m} \min(I_i, M_i)}{\sum_{i=1}^{m} M_i} = \frac{1}{2n} \sum_{j=1}^{m} |I_i - M_i| \tag{4.6}$$

Histogram intersection is a very simple and fast distance measure, which can even be computed incrementally [Swain and Ballard, 1991]. It is important to note that the histogram intersection distance is a full featured metric implying non-negativity, identity of indiscernibles, subadditivity and symmetry. Consequently any histogram can be interpreted as a point in a normed vector space, which probably allows to apply more powerful search algorithms, e.g., *Locality Sensitive Hashing* (LSH). This could be especially useful in the context of very large audio archives and is an essential advantage over the KL divergence, for example, which does not fulfill the triangle inequality and is not symmetric by itself.

4.3.1.4 Results and Conclusions

In our experiment the proposed non-parametric MLVQ approach is compared to the SG algorithm as a well-known parametric variant of the BoF approach. The comparison is based on the *"1517-Artists"* genre classification dataset. The full length audio tracks down-sampled to 11kHz serve as input signals for this experiment. The Neighbour Accuracy (NA) and the R-precision (R-Prec) serve as quality indicators (see 3.2.1). Table **??** summarizes the obtained results for the two approaches and a random baseline algorithm. Both artist-filtered and not-artist-filtered results are reported. In line with the results reported in [Aucouturier, 2006] the SG model outperforms the MLVQ approach, when no artist filter is used. With artist filter, however, the MLVQ approach achieves a better result. Thus, we can conclude that the VQ model does not fit the observed distributions as closely as the SG model. By closely fitting the observed distribution, the SG model can even model artist specific aspects of songs. Still, the VQ approach models the general shape of the distribution in an adequate way so that genre specific aspects are captured. Based on this result we can expect equal performance of both with respect to their ability to identify non-trivial recommendations. Thus, the MLVQ approach might also be a good choice to generate recommendations, especially

AF	indicator	MLVQ	SG	RND
without	1-NA	36.65%	**37.33%**	5.03%
	5-NA	28.16%	**28.22%**	5.63%
	R-Prec	0.138	**0.141**	0.061
with	1-NA	**22.83%**	21.57%	4.97%
	5-NA	**19.96%**	18.70%	5.58%
	R-Prec	0.136	**0.140**	0.050

Table 4.1: Evaluation results for three similarity algorithms: Multi-Level Vector Quantization (**MLVQ**), Single Gaussian (**SG**), Random Guess (**RND**).

because it has some additional advantages compared to the SG algorithm. For example, in contrast to semi-parametric and parametric modeling approaches, the MLVQ approach does not suffer from the *hub* problem (see 6.2.3) that much. This will be shown in section 6.2.3. Additionally, the intersection distance that is used to compare two histograms is a full-featured metric. This allows to use more efficient search algorithms to identify similar songs within a music catalog. Another advantage of the MLVQ algorithm is that the song models can easily be resynthesized [Seyerlehner et al., 2008], which then allows to analyze which properties of a song are actually captured by a song model. Nonetheless, there also exist some disadvantages. First of all the training phase to generate a codebook is still computationally expensive and furthermore, there does not exist a straightforward solution to update the learned codebook and the generated models whenever new songs are added to an existing database.

Altogether, the conducted analysis shows that non-parametric distribution models do not perform worse than parametric distribution models, at least not with respect to the ability to identify non-trivial recommendations. The second analysis that is presented in this chapter will focus on identifying those audio frames that make two audio track similar according to BoF approaches.

4.3.2 Analysis 2: *Why do songs sound the same from the machine's point of view?*

Although BoF approaches are widely used to compute content-based audio similarity, it has not yet been analyzed in the literature which are actually those MFCC vectors that make two songs similar from a machine's point of view. In this section these feature vectors are identified via analyzing the contribution of the individual MFCC vectors to the KL divergence of two songs. To be able to compute the individual contribution of a feature vector to the KL divergence an even more direct approach to modeling the distribution of MFCC vectors is used by applying *nearest neighbour density estimation.* Using a nearest neighbour density estimation method has several advantages compared to GMMs and SG. First of all, common parametric forms rarely fit densities actually encountered in data, in particular because all the *"classical"* parametric densities (e.g. the Gaussian distribution) are unimodal, whereas many practical problems involve multi-modal densities. Another general advantage of non-parametric in contrast to parametric procedures is that they can be used with any distributions and without the assumption that the form of the underlying densities are known. Unfortunately, estimating the KL divergence of two songs via nearest neighbour density estimation is not a practical solution due to the computational complexity. For this reason it has not been used much in the literature. In the only related work [Godfrey and Chordia, 2008], where kernel density estimation is used, each dimension of a MFCC vector is modeled independently of the others for performance reasons. In this section, where BoF approaches are studied and analyzed, performance is not an issue, but it is of course important for the conducted analysis that the existing dependencies of the dimensions of the MFCC vectors are correctly modeled. Thus, in the next subsections it will be first discussed how to derive correct density estimates using nearest neighbour estimation and then how to identify those frames which contribute most to the KL divergence of two distributions of MFCC vectors.

4.3.2.1 Nearest Neighbour (NN) density estimation

Given a set of n points sampled from an arbitrary distribution, one can estimate the density $p(\mathbf{x})$ at or around a point $\mathbf{x}$ by counting the number of sample points that fall into a small region around $\mathbf{x}$. If k is the number of points in that region and V is the volume of the region, one can estimate the density $\hat{p}(\mathbf{x})$ at point $\mathbf{x}$ according to equation (4.6).

$$\hat{p}(\mathbf{x}) = \frac{k/n}{V} \tag{4.7}$$

This method is called *nearest neighbour density estimation* [Duda et al., 2000], [Bishop, 2006]. Given infinitely many sample points, the radius r could be chosen infinitely small and the estimate $\hat{p}(\mathbf{x})$ would converge toward the true probability $p(\mathbf{x})$. In practice, given a finite sample set, r is a crucial application-specific parameter. In the following, NN density estimation will be used to model feature distributions. Together with an estimation of the KL divergence, this will permit to analyze the role of individual sample points in the computation of similarities.

4.3.2.2 Estimating the KL divergence

Given n MFCC vectors of a song, one can directly estimate the density at each vector $\mathbf{x}$ by counting the number k of MFCC vectors with a distance $\leq r$ from $\mathbf{x}$, and taking k/n as an estimate of (proportional to) the density of $p(\mathbf{x})$. For two sets X_P and X_Q representing two songs, we first reduce the larger set so that both sets have equal sizes ($|X_P| = |X_Q|$). This can be easily achieved by randomly removing points from the larger set and should have little influence on the overall distribution.

The usual approach to compare two distributions P and Q of two songs is to compute the discrete KL divergence:

$$\hat{D}_{KL}(P||Q) = \sum_{\mathbf{x}} \hat{p}(\mathbf{x}) \log \frac{\hat{p}(\mathbf{x})}{\hat{q}(\mathbf{x})}, \tag{4.8}$$

where $\mathbf{x} \in X_P \cup X_Q$. Inserting the density estimates according to equation 4.6 one arrives at the following expression for the KL divergence, where $k_{\mathbf{x},p}$ denotes the number of points in the small region around $\mathbf{x}$ for distribution P and $k_{\mathbf{x},q}$ for distribution Q respectively. n_p is the number of sample points of distribution P and n_q the number of sample points of distribution Q.

$$\hat{D}_{KL}(P||Q) = V \sum_{\mathbf{x}} k_{\mathbf{x},p}/n_p \log \frac{k_{\mathbf{x},p}/n_p}{k_{\mathbf{x},q}/n_q} \tag{4.9}$$

Note that the larger set of sample points is reduced so that the number of sample points n is the same for both distributions P and Q. Therefore the estimate of the KL divergence can be further reduced.

$$\hat{D}_{KL}(P||Q) = \frac{V}{n} \sum_{\mathbf{x}} k_{\mathbf{x},p} \log \frac{k_{\mathbf{x},p}}{k_{\mathbf{x},q}} \tag{4.10}$$

Unfortunately, this approximation is numerically unstable, because whenever the NN estimate of the probability density $\hat{q}(\mathbf{x})$ is zero (which can happen depending on the radius r), $\log \frac{\hat{p}(x)}{\hat{q}(x)}$ will be undefined. We decided to circumvent this problem by increasing $k_{\mathbf{x},p}$ and $k_{\mathbf{x},q}$ by one so that the estimates $\hat{p}(\mathbf{x})$ and $\hat{q}(\mathbf{x})$ cannot become zero. We can also derive the symmetric KL divergence $D_{KL_{sym}}$.

$$\begin{aligned}
\hat{D}_{KL_{sym}}(P||Q) &= \hat{D}_{KL}(P||Q) + \hat{D}_{KL}(Q||P) \\
&= \sum_{\mathbf{x}} \hat{p}(\mathbf{x}) \log \frac{\hat{p}(\mathbf{x})}{\hat{q}(\mathbf{x})} + \sum_{\mathbf{x}} \hat{q}(\mathbf{x}) \log \frac{\hat{q}(\mathbf{x})}{\hat{p}(\mathbf{x})} \\
&= \sum_{\mathbf{x}} (\hat{p}(\mathbf{x}) \log \frac{\hat{p}(\mathbf{x})}{\hat{q}(\mathbf{x})} + \hat{q}(\mathbf{x}) \log \frac{\hat{q}(\mathbf{x})}{\hat{p}(\mathbf{x})}) \\
&= \sum_{\mathbf{x}} (\hat{q}(\mathbf{x}) - \hat{p}(\mathbf{x})) \log \frac{\hat{q}(\mathbf{x})}{\hat{p}(\mathbf{x})} \\
&= \sum_{\mathbf{x}} (\hat{p}(\mathbf{x}) - \hat{q}(\mathbf{x})) \log \frac{\hat{p}(\mathbf{x})}{\hat{q}(\mathbf{x})} \\
&= \frac{V}{n} (\sum_{\mathbf{x}} (k_{\mathbf{x},p} - k_{\mathbf{x},q}) \log \frac{k_{\mathbf{x},p}}{k_{\mathbf{x},q}})
\end{aligned} \tag{4.11}$$

Since the symmetric KL divergence is used as a distance measure and V/n is a constant factor, V/n can safely be neglected for our purpose. From equation 4.10

it can bee seen that the estimate of the KL divergence is a sum over all sample points. Thus, it is easy to compute the contribution of an individual vector $\mathbf{x}$ to the overall KL divergence.

4.3.2.3 Validation of the Approach

To validate the proposed nearest neighbour density approach and the implementation thereof, it is compared to the standard Single Gaussian (SG) method with KL-divergence (see 4.2.2). As validation dataset the *"Annotated"* dataset is used. The main reason for this small classification dataset is the high computational cost of comparing two distributions using the nearest neighbour density method. Figure 4.4 shows the results of the genre classification. For the NN density estimation the radius r has been varied in between 20 to 60. The dotted line represents the neighbour accuracy obtained by the SG algorithm, which does not depend on the radius r. The radius that is used for NN density estimation has a significant influence on the neighbour accuracy. Around the optimum the NN density estimation approach outperforms the SG algorithm. But that is not the point of this experiment. The important aspect of this experiment is that the NN approach itself and the implementation thereof is correct, because the classification accuracy is far above the baseline and close to the result obtained using the SG approach for an appropriate radius r. Thus we can make use of this approach to analyze and understand the BoF in more detail.

4.3.3 A simple Frame Selection Strategy

To analyze which frames make two songs similar from a machine's point of view, a little tool has been developed that loads two audio files, computes their spectral representation, and calculates the similarity value based on nearest neighbour density estimation. Furthermore, it sorts and visualizes the audio frames according to their contribution to the overall distance (see figure **??**). Using the developed tool many pairs of songs have been manually analyzed to find out which frames make them appear similar to the machine. What jumps to the eye is that frames

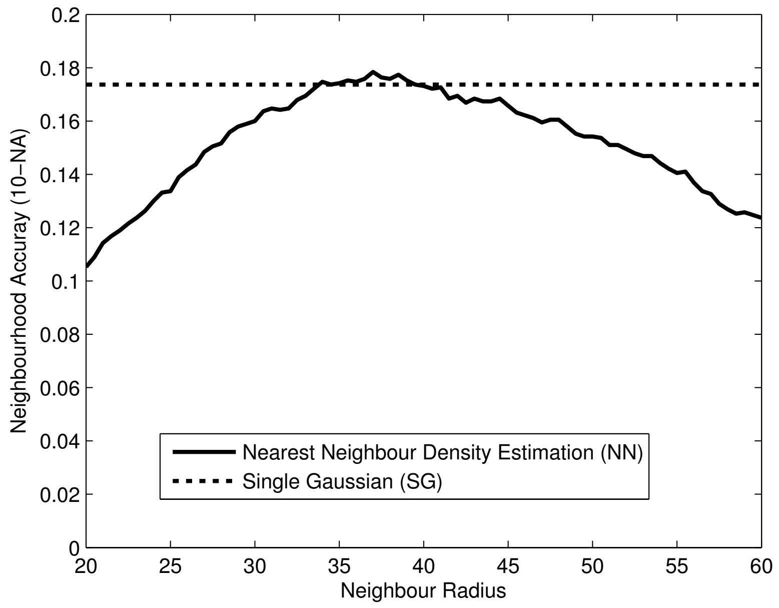
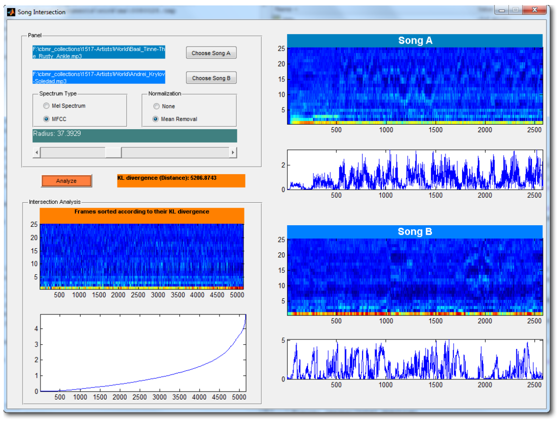


Figure 4.4: Neighbour Accuracy (NA) for the NN density approach of different smoothness for dataset *"Annotated"*.

with rather low energy tend to contribute quite a lot to the similarity judgment. This is especially interesting as those frames are of course not the most salient ones with respect to human perception. This finding is supported by a related observation that was reported in [Godfrey and Chordia, 2008] were Godfrey et al. present different methods to analyze and reduce the *hubness* (see 6.2.3) of frame-level similarity algorithms. From a technical point of view frames with low energy will of course have a low euclidean distance to each other, which implies that the density of low energy frames will be high. Thus the KL divergence for those frames will only be low if the other song has an equal amount of, e.g., silent or almost silent frames. Consequently, songs having low energy frames will more likely match songs that have low energy frames as well, although from a human point of view the similarity of two songs is surely related to the dynamic parts of a song and is not too much influenced by the silent parts of a song. We therefore decided to investigate whether removing those low energy frames before building a model would improve audio similarity measures. To select frames of high energy, we determine



F:\cbmr_collections\1517-Artists\World\Baal_Tinne-The_Rusty_Ankle.mp3

F:\cbmr_collections\1517-Artists\World\Andrei_Krylov-Soledad.mp3

Spectrum Type: ( ) Mel Spectrum [✓ MFCC]

Normalization: ( ) None [✓ Mean Removal]

Radius: 37.3929

KL divergence (Distance): 5206.8743

Figure 4.5: The tool that has been developed to analyze the contribution of individual frames to the KL divergence.

the energy of each frame after mapping onto the mel-scale by summing across the mel bands. Then all the frames are sorted according to their energy and a given percentage of frames is dropped (e.g 50%). We then compute the model as usual, but for the remaining frames only. This frame selection strategy can be applied to any BoF approach. In the next section the frame selection strategy is evaluated via classification experiments. For the SG model the improvement in quality that can be achieved with this simple frame selection strategy is illustrated.

4.3.3.1 Evaluation of the proposed Frame Selection Strategy

To evaluate the proposed frame selection strategy the SG algorithm is modified. First, the classification accuracy using all frames (the original algorithm) is computed. Then before building the distribution model systematically 10%, 20%, ..., 80% and 90% of the frames are removed according to their energy as described in section 4.3.3. This strategy is called *Frame Selection* (**FS**). Additionally, we compare this strategy against a random frame selection strategy (**RND**), where we remove the same percentage of frames by choosing the frames to be removed randomly. To prevent collection specific effects two different genre classification datasets were used to evaluate the proposed strategy. Both artist-filtered and not

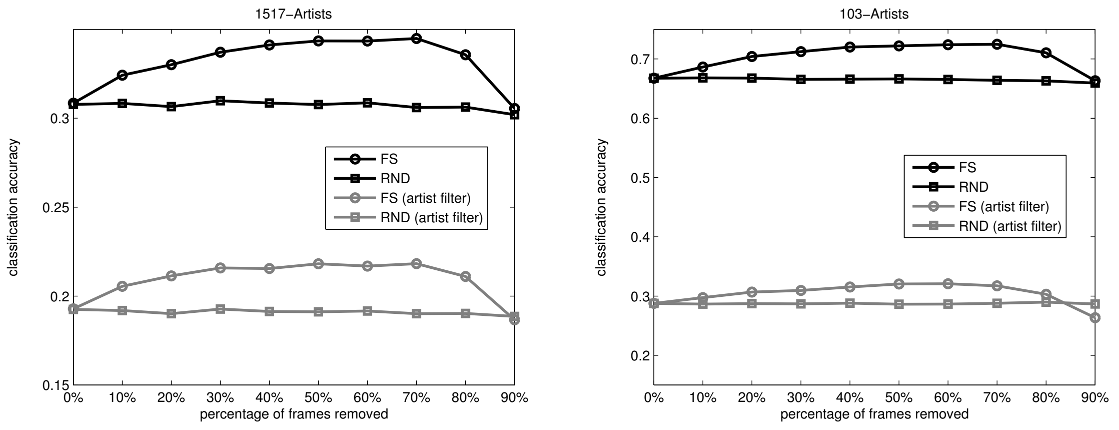


Figure 4.6: Neighbour Accuracies (NA) of the proposed frame selection strategy and of a random selection baseline-strategy for dataset *"1517-Artists"* (left) and dataset *"103-Artists"* (right).

artist-filtered results are reported. With respect to recommendation the artist-filtered results are of course more relevant. In all our evaluations we present results for the 4-NA, because some of the genres of the *"103-Artists"* dataset do not contain more than 4 different artists, and measuring the accuracy for more than the top 4 recommendations would return invalid classification results in combination with an artist filter. Figure 4.5 shows the genre classification results for the *"1517-Artists"* and *"103-Artists"* datasets, respectively. A look at the results for the random frame selection strategy shows that the SG model is remarkably stable. Even with up to 90% of all audio frames removed, the classification accuracy decreases only marginally. Furthermore, for both datasets we can observe the same effect, an improvement in classification accuracy when we remove low energy frames using the proposed frame selection strategy. For both collections we can reach a classification optimum if we remove between 50% and 70% of all frames, giving an improvement of about 3 percentage points in classification accuracy on our datasets.

To conclude, using the NN density estimation approach a simple *"Frame-Selection"* has been developed that can help to improve the recommendation quality of BoF approaches. Another interesting finding is that while BoF approaches are based on the euclidean distances between spectral features, the euclidean distance between local spectral features obviously does not correspond to the human similarity perception in some cases. One example is that human beings would not claim that two songs are similar because they share some silent parts, although these silent frames will have a low euclidean distance to each other. In the next section additional experiments are presented to demonstrate the general limitations of BoF approaches.

4.4 Limitations and Conclusions

4.4.1 Limitations of Frame-Level Algorithms

In the literature it is well-known that the performance of frame-level similarity algorithms with respect to genre classification is fundamentally bounded. This behavior is known as *"glass ceiling"* [Aucouturier and Pachet, 2004]. The important question is of course *why* are distribution models over low level features not representative of musical genres [**?**]. One can identify at least three aspects of the BoF approach that explain this inability: *Pitch Dependency*, the *Inability of Matching Components in Polyphonic Music* and *Loss of Temporal Information.*

4.4.1.1 Pitch Dependency

One limiting factor of the BoF approach is the pitch dependency of the local spectral features (e.g. MFCCs). In contrast to humans, who would identify two violin-tones played at different pitch as similar, frame-level algorithms based on MFCCs are not able to infer this relation. This can be demonstrated using the NN density estimation approach presented in the previous chapter. Figure **??** visualizes the MFCC vectors of the 12 semi-tones of an octave played by a soprano saxophone[3]. Additionally, also the pairwise euclidean distance of the MFCC vec-

[3]The instrument sample was taken from the free musical instrument samples library of the University of Iowa, http://theremin.music.uiowa.edu/MIS.sopranosax.html

tors are visualized in figure **??** (a). These distances can be used to analyze which frames can be matched by the NN density based BoF approach. If the distance between two MFCC frames is smaller than r, then these frames are *"matched"* by that BoF variant. In the last section it was found that an optimal value for r is 37. In figure **??** (b) only those distances of matching frames, where the distance is smaller than r, are set to 1 and otherwise to 0. Obviously, the majority of frames of different semi-tones do not match, while the short silent part in between the played notes do match. This indicates that MFCC vectors are pitch-dependent and that tones that are played at different pitches but by the same instrument are not found to be similar by frame-level similarity algorithms, which is definitely a weakness limiting the achievable recommendation quality. In the following a related experiment is conducted that shows the inability of the BoF approach to capture similarities among components in polyphonic music.

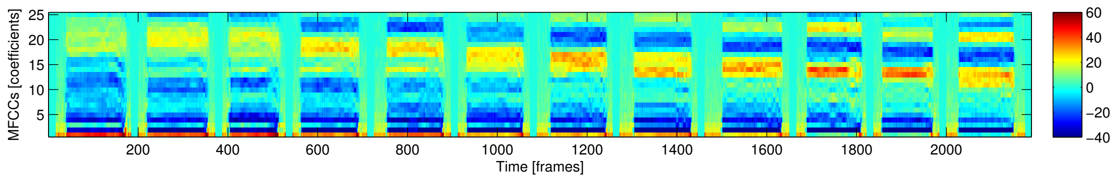


Figure 4.7: MFCC representation of the 12 semi-tones played by a soprano saxophone.

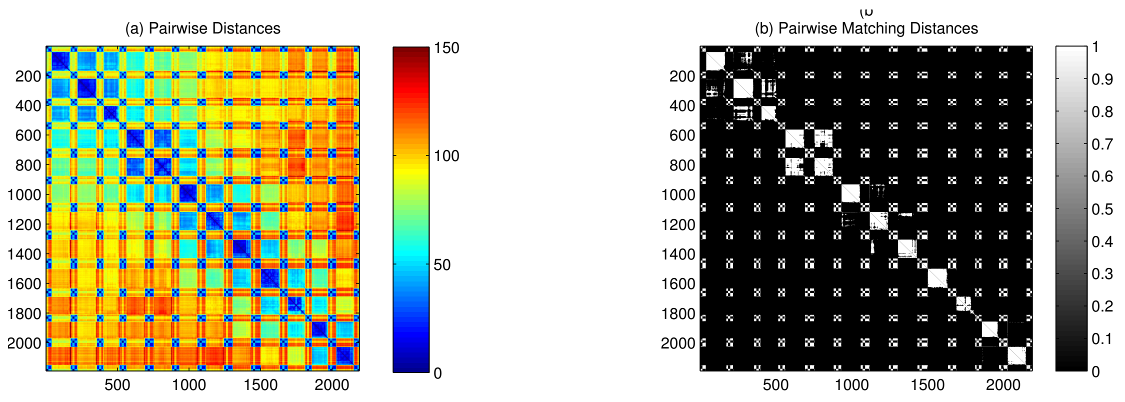


Figure 4.8: Pairwise distances between the MFCC vectors of the 12 semi-tones of a soprano saxophone (a). In (b) one can see that in the best case some frames of adjacent semi-tones can be matched by the NN density estimation approach.

4.4.1.2 Inability of Matching Components in Polyphonic Music

In [Aucouturier et al., 2007] Aucouturier found that the BoF approach is a sufficient model for soundscapes, but not for polyphonic music. He argues that soundscape signals are very homogeneous and statistically redundant, while polyphonic music signals are far more heterogeneous. This heterogeneity can be explained by the fact that music signals are typically complex mixtures of different sound sources. Unfortunately, frame-level algorithms can only identify similarities between mixtures of signals if these mixtures consist of approximately the same components. For example a BoF approach is not able to identify any similarity between one component of a mixture and the mixture itself. Figure **??** visualizes the MFCC representation of the 12 semi-tones of the soprano saxophone in **??** combined with an a-capella accompaniment[4]. The 12 semi-tones can still be identified in the MFCC representation, but one can also observe that the MFCC vectors have changed significantly. The modified signal was appended to the original signal. Figure **??** shows the pairwise distances of the MFCC representation of

[4]An excerpt of an a-capella song from the *Golden Gate Quartet* out of the *"103-Artists"* dataset served as a-capella accompaniment.

the original with the appended modified signal. As previously done, those frames that can be matched by the NN density estimation approach for a radius $r = 37$ are visualized in the lower plot. The upper left quarter of the illustration is the same as in **??**. The lower left and the upper right quarter are those that indicate matches between the soprano saxophone and the mixture of the soprano saxophone and the a-capella accompaniment. Interestingly, most of the frames from the mixture signal cannot be matched to the soprano saxophone signal that is inside the mixture. Of course this effect will be even worse for more complex music signals containing many instruments and a singing voice. Thus, the often high number of sources in a polyphonic music signal could be an explanation for their heterogeneity. Furthermore, this example clearly shows the inability of the BoF approach to match individual components within polyphonic music signals. Thus, one future research direction could be to decompose music signals into their sources prior to modeling them, e.g. using the harmonic and percussive decomposition proposed in [**?**]. In the following subsection the loss of temporal information resulting from the distribution models of frame-level algorithms is discussed.

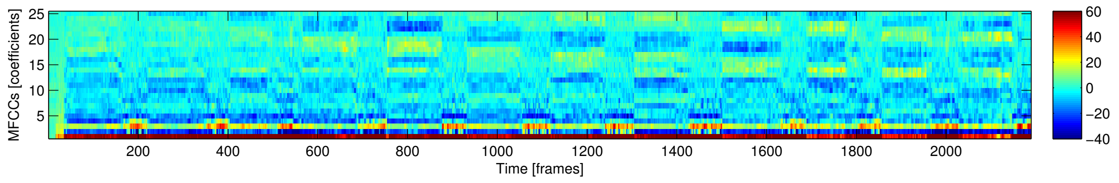
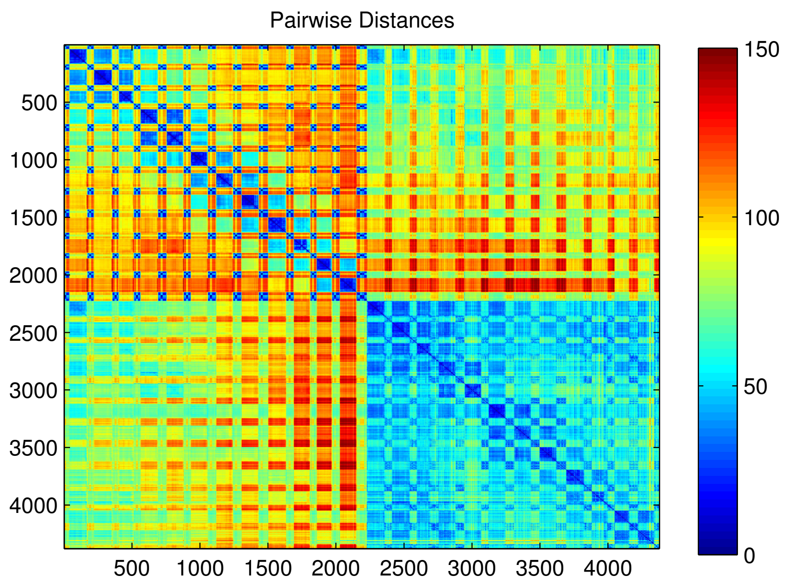


Figure 4.9: MFCC representation of the 12 semi-tones played by a soprano saxophone with an a-capella accompaniment.



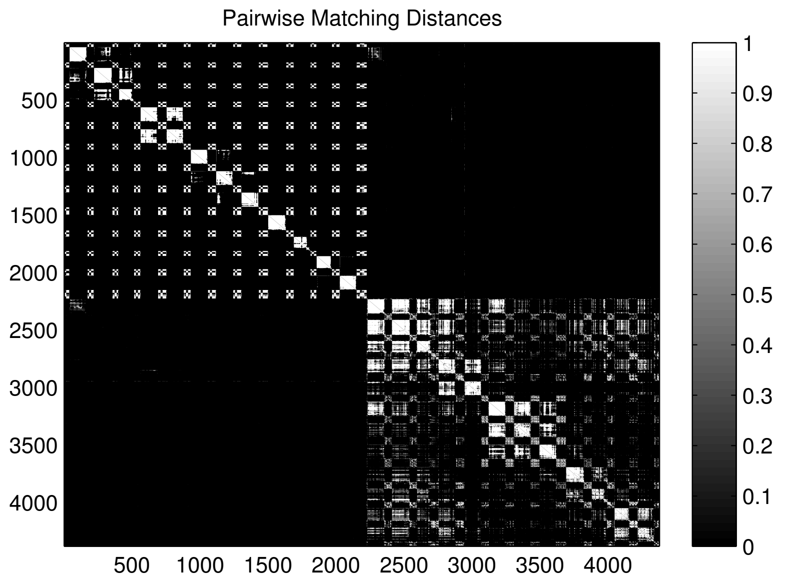


Figure 4.10: Pairwise distances between the MFCC vectors of just the 12 semitones of a soprano saxophone and the mixture of a soprano saxophone with an a-capella accompaniment. There are almost no matches between the mixture and solo saxophone.

4.4.1.3 Loss of Temporal Information

The third and most obvious problem of frame-level algorithms is that the temporal relation of the MFCC vectors is completely lost because of the distribution model. However, randomly shuffling audio frames has a strong influence on the human perception of music [?]. Aucouturier found via a listening experiment that *"for music however, the similarity relations that are consensually established for normal signals are completely lost by splicing"*. Thus, local temporal information seems to be crucial for the human perception of music. While this finding motivates models that incorporate more temporal information, straightforward approaches of modeling the temporal relation of local spectral features (for example HMMs) do not perform better than static models (see **??**). In this thesis another approach to capture some temporal information is proposed by defining so-called *block-level* features that are intended to capture some local temporal information in the local audio features themselves. This is the topic of the following Chapter.

4.4.2 Conclusions

The presented analysis of frame-level algorithms reveals that the type of the distribution model is not crucial with respect to the ability of these algorithms to generate non-trivial recommendations. Furthermore, based on a NN density estimation method a simple frame selection strategy has been identified that can improve the recommendation quality of frame-level similarity algorithms. However, the main finding is that BoF approaches are rather limited in identifying musical similarities among songs in general and only match songs that have very similar spectra. The main reasons of these limitations seem to be that BoF approaches are not pitch invariant, cannot identify existing similarity relations in case of multiple sound sources and do not capture any temporal information of the analyzed signal. In the next chapter the latter deficiency will be addressed by moving from frame-level algorithms to so-called *block-level* algorithms that can also capture some temporal information of an audio signal.

5 Block-Level Music Similarity Algorithms

In the last chapter *frame-level music* similarity measures have been analyzed and the main finding of the previous chapter was that frame-level algorithms are rather limited in their ability to capture non-trivial music similarity relations as it would be necessary for music recommendation. One limiting factor that has been identified is the loss of temporal information that is a consequence of the distribution model in combination with the local frame-level audio features. While the distribution model does not consider any temporal relation among audio frames, the local audio features themselves are not able to capture any temporal information, because of the very tiny amount of audio data that is analyzed. To address this problem in this chapter so-called *block-level* features are defined. In contrast to *frame-level* features, the proposed features are defined on a consecutive sequence of audio frames called *block*. This way the features themselves are able to capture some local temporal information. Another important advantage of the proposed block-level features that will be useful in this chapter is their simple vector space representation, which allows to take advantage of the whole mathematical toolbox that is available for vector spaces.

The outline of this chapter is as follows: First the basic idea of the conceptual block processing framework is sketched. Then in section 5.2 seven novel features that can be extracted within this conceptual framework are introduced. Section **??** and **??** then introduce two different approaches to compute music similarity based on these block-level features. One approach is to directly define a music similarity measure by combining the similarity estimates of the individual patterns, and the second approach is to define a similarity measure by performing a mapping over a semantic tag space. Then in section 5.5 an extensive evaluation of the proposed similarity algorithms is presented. Via genre classification experiments the proposed algorithms are evaluated and compared to a set of music similarity algorithms consisting of *"classical"* and *"state-of-the-art"* algorithms. Furthermore, several combinations of the best performing algorithms are compared to identify any potential for further improvement. Besides this in-house evaluation also the results of the *MIREX 2010* evaluation are discussed, where the proposed block-level features have been used for three different tasks. Section **??** concludes on the obtained results in this chapter. The next section starts with an introduction to the block processing framework, the fundamental processing concept behind the proposed block-level audio features.

5.1 The Block Processing Framework

The idea of processing audio block by block is inspired by the feature extraction process of the *Fluctuation Patterns* described in [Lidy and Rauber, 2005, Pampalk et al., 2002, Rauber et al., 2003]. Here, the idea of processing time-frequency representations block by block is generalized and serves as a general processing paradigm, called the *block processing framework*. Following this block processing paradigm, in addition to the Fluctuation Patterns several novel audio features (see 5.2) have been defined that are useful to describe the content of an audio signal. The basic block processing framework can be subdivided into two stages: first, the *block processing stage* and second, the *generalization stage*.

5.1.1 Block Processing

For block-based audio features the whole spectrum is processed in terms of blocks. Each block consists of a fixed number of spectral frames defined by the block size. Two successive blocks are related by advancing in time by a given number of frames specified by the hop size. Depending on the hop size blocks may overlap,

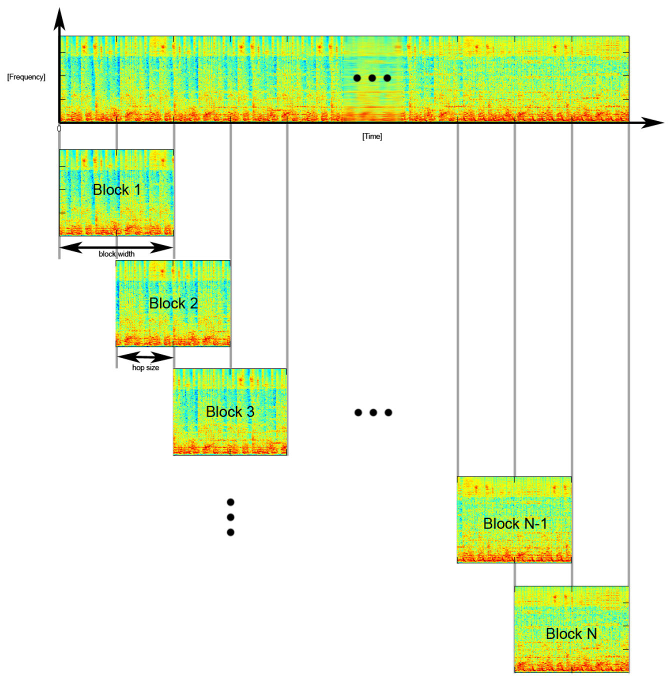


Figure 5.1: Block by block processing of the cent spectrum.

or there can even be unprocessed frames in between the blocks. Although the hop size could also vary within a single file to reduce aliasing effects, the features that are introduced in this section are extracted using a constant hop size. Figure 5.1 illustrates the basic process.

Intuitively, a block can be interpreted as a matrix that has W columns defined by the block width and H rows defined by the frequency resolution (the number of frequency bins):

$$\text{block} = \begin{bmatrix} b_{H,1} & \cdots & b_{H,W} \\ \vdots & \ddots & \vdots \\ b_{1,1} & \cdots & b_{1,W} \end{bmatrix} \tag{5.1}$$

The main advantage of defining features on blocks of frames instead of defining

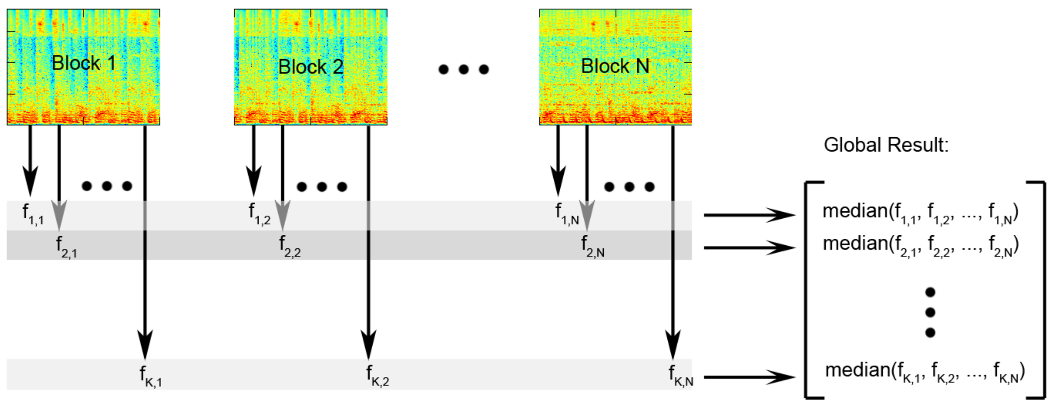

Block 1
Block 2
Block N
Global Result:
$f_{1,1}$
$f_{2,1}$
$f_{1,2}$
$f_{2,2}$
$f_{1,N}$
$f_{2,N}$
$f_{K,1}$
$f_{K,2}$
$f_{K,N}$
median($f_{1,1}$, $f_{1,2}$, ..., $f_{1,N}$)
median($f_{2,1}$, $f_{2,2}$, ..., $f_{2,N}$)
median($f_{K,1}$, $f_{K,2}$, ..., $f_{K,N}$)

Figure 5.2: Generalization from block level features to song feature vectors, with the median as summarization function.

features on single frames, e.g. MFCCs, is that each block comprises a sequence of several frames, which allows to perform some temporal processing within the local feature extraction process. It is important to note that there are fundamental differences between the block-processing framework and temporal modeling via *texture windows* [Tzanetakis and Cook, 2002, ?, ?]. Texture-windows summarize and aggregate frame-level features by building low-order statistics over a longer sliding time window. However, no explicit temporal processing is performed. In contrast the block processing framework offers the possibility to explicitly perform temporal processing, as all frames in a block are processed by the local feature extraction process at once. This way the resulting features can capture temporal information, e.g. rhythm or dynamics related properties, over a longer local time span, which is defined by the block width.

5.1.2 Generalization

To come up with a global feature vector per song, the local feature vectors of all blocks must be combined into a single representation for the whole song. To combine local block-level features into a song model, a *summarization function* is applied to each dimension of the local feature vectors separately. Typical summarization functions are, for example, the mean, median, certain percentiles, or the variance over a feature dimension. Interestingly, also the classic Bag-of-Frames approach (see 4.2) can be interpreted as a special case within this framework. The block size would in this case correspond to a single frame only, and a Gaussian Mixture Model would be used instead of the simple summarization function considering each dimension separately. However, we explicitly do not consider distribution models as summarization functions here, as our goal is to define a song model whose components can be interpreted as vectors in a vector space. The *generalization process* is illustrated in Fig. 5.2 for the median as summarization function.

In the following, for each of the proposed block-level features the local feature extraction process for a block is specified and also the summarization function that is used to generate the global song-level feature vector. While Fig. 5.2 depicts the block level features as vectors, the features described below will be matrices. This makes no difference to the generalization step, however, as the summarization function is applied component by component; the generalized song-level features will thus also be matrices.

5.2 Block-Level Features

In principle the proposed block-level features are not tied to any specific time-frequency representation. However, in our implementation all proposed patterns — except the CFA feature — are based on the normalized Cent-Spectrum (see 2.4.5), since the normalized Cent-Spectrum is an adequate time-frequency representation for music signals and is additionally cheap to compute. The following subsections describe the block-level features that are up to now defined within the block processing framework.

5.2.1 Spectral Pattern (SP)

The basic intention behind the *Spectral Pattern* is to characterize a song's timbre via modeling those frequency components that are simultaneously active. To characterize the local frequency components and their dynamics the normalized Cent-Spectrum is processed using short blocks containing 10 spectral frames only. The blocks are overlapping because of the 5 frames hop size between two consecutive blocks. The temporal processing step of the SP is to sort each frequency band of the block along the time axis:

$$\text{SP} = \begin{bmatrix} \text{sort}(b_{H,1} & \cdots & b_{H,W}) \\ \vdots & \ddots & \vdots \\ \text{sort}(b_{1,1} & \cdots & b_{1,W}) \end{bmatrix} \tag{5.2}$$

This way the general frequency content and the general dynamics of a block is kept, while specific timing aspects like the precise onset positions within a block are ignored. To come up with a representative pattern for the whole song the 0.9 percentile is used as summarization function. Altogether, the Spectral Pattern is slightly related to the *Spectrum Histogram* (SH) proposed by Pampalk et al. [?]. In contrast to the SH the SP is not a histogram and the global pattern is derived from the local loudness patterns. Thus, it is a generalization over the local dynamics of co-occurring frequency components.

5.2.2 Delta Spectral Pattern (DSP)

The *Delta Spectral Pattern* is inspired by the Delta MFCCs, the first derivate of the static MFCCs (see 2.4.4). The Delta MFCCs are known to capture some dynamic aspects of music signals. Regarding the DSP the goal is to quantize the strength of onsets. For this reason a longer delay is used. The difference between the original Cent-Spectrum and a copy of the spectrum delayed by 3 frames is computed to emphasize onsets. The resulting delta Cent-Spectrum is rectified so that only positive values are kept. Then we proceed exactly as for the Spectral

Pattern and sort each frequency band of a block along the time axis. A block size of 25 frames and a hop size of 5 frames are used, and the 0.9 percentile serves as summarization function. It is important to note that the DSP's block size differs from the block size of the SP; both were obtained via optimization. This way the two patterns are able to capture information over different time spans.

5.2.3 Variance Delta Spectral Pattern (VDSP)

The feature extraction process of the *Variance Delta Spectral Pattern* is the same as for the Delta Spectral Pattern (DSP). The only difference is that the variance is used as summarization function over the individual feature dimensions. While the Delta Spectral Pattern (DSP) tries to capture the strength of onsets, the VDSP should indicate if the strength of the onsets varies over time or, to be more precise, over the individual blocks. A hop size of 5 and a block size of 25 frames are used.

5.2.4 Logarithmic Fluctuation Pattern (LFP)

To represent the rhythmic structure of a song we extract the *Logarithmic Fluctuation Patterns*, a modified version of the *Fluctuation Pattern* proposed by Pampalk et al. [Pampalk et al., 2002]. For this pattern a block size of 512 and a hop size of 128 are used. For each frequency band of the block the FFT is computed to extract the amplitude modulations out of the temporal envelope in each band. We only keep the amplitude modulations up to 600 bpm. The amplitude modulation coefficients are weighted based on the psychoacoustic model of the *fluctuation strength* [?] according to the original approach in [Pampalk et al., 2002]. To represent the extracted rhythm pattern in a more tempo invariant way, we then follow the ideas in [Pohle et al., 2009, Jensen et al., 2009, Holzapfel and Stylianou, 2009] and represent the periodicity dimension of the rhythm pattern in log scale instead of linear scale. This is realized by summing across the linear periodicity bins that fall into the 37 logarithmically scaled periodicity bins. Finally, the resulting pattern is blurred with a Gaussian filter, but for the frequency dimension only. The summarization function is the 0.6 percentile.

5.2.5 Correlation Pattern (CP)

The *Correlation Pattern* is motivated by the fact that typically rhythm patterns only reflect the periodicity of reoccurring musical events e.g. drum events, but not their relative phase. For rhythm representations the temporal relation of reoccurring events is of course crucial, as striking both high-hat and bass drum simultaneously and striking a high-hat in between the bass events results in a completely different rhythm. To capture the temporal relation of loudness changes over different frequency bands, first the frequency resolution of the normalized cent spectrum is reduced to 52 frequency bands only. This was found to be useful by optimization and also reduces the dimensionality of the resulting pattern. Then the pairwise linear correlation coefficient, Pearson's correlation (see equation **??**), between each pair of frequency bands is computed, which gives a symmetric correlation matrix.

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{(n-1)s_x s_y} \tag{5.3}$$

A related idea has already been proposed by Aylon [Aylon, 2006], who proposed a descriptor called *Band Loudness Intercorrelation* (BLI). The BLI descriptor has been used for music classification by Guaus [**?**]. Besides some differences the main advantage of the CP over the BLI is that it is not computed over the whole signal, but locally for each block separately, which is computationally efficient, and a song-level descriptor is derived via using the 0.5 percentile as summarization function. The CP can capture, for example, harmonic relations of frequency bands when sustained musical tones are present. Furthermore, as already mentioned, rhythmic relations can be reflected by the CP. Using the above example of bass and high-hat positive correlations between low and high frequency bands would indicate that bass drum and high-hat are always hit simultaneously. Conversely, if the high-hat and the bass drum are never played together this would result in strongly negative correlations in the CP, which then results in specific perceivable patterns. Such typical patterns of the CP can be found by visualizing the CP. For example the presence of a singing voice leads to very specific correlation patterns, which is even more obvious for the CP computed based on a time-frequency representation with a higher frequency resolution. In the current implementation a block size of 256 frames and a hop size of 128 frames are used.

5.2.6 Spectral Contrast Pattern (SCP)

The *Spectral Contrast* [Jiang et al., 2002] is a feature that roughly estimates the *"tone-ness"* of a spectral frame. To characterize the tone-ness the difference between spectral peaks and valleys in several sub-bands is computed for each frame within a block. As strong spectral peaks roughly correspond to tonal components and flat spectral excerpts are often related to noise-like or percussive elements, the difference between peaks and valleys characterizes the toneness in each sub-band. In our implementation the Spectral Contrast is computed from the cent scaled spectrum subdivided into 20 frequency bands. For each audio frame, we compute in each band the difference between the maximum value and the minimum value of the frequency bins within that band. This results in 20 Spectral Contrast values per frame. The values pertaining to an entire block are then sorted along the time axis within each frequency band, as already described for the SP above. A block size of 40 frames and a hop size of 20 frames are used. The summarization function is the 0.1 percentile.

5.2.7 Continuous Frequency Activation (CFA)

The *Continuous Frequency Activation* is the last block-level feature that is described in this chapter. In contrast to the previous features it is not a high dimensional feature, but just a single scalar value. However, the CFA feature is a perfect example on how one can exploit the temporal dimension within the feature extraction at the block-level. Initially the CFA feature was designed as a robust and powerful feature for automatic music detection [**?**].

The CFA feature is based on the observation that music tends to have more stationary parts than speech, resulting in perceivable horizontal bars within the spectrogram representation of an audio signal (see figure 5.5). This property was already investigated by Hawley, who was interested in the structure of music [Hawley, 1993] and who was the first to propose a simple *music detector* based on this. The horizontal bars in the spectrogram are continuous activations of specific frequencies and are usually the consequence of sustained musical tones. The CFA is designed to reliably detect these continuous frequency activations, even if other audio signals are present simultaneously. To detect music signals within mixtures five steps are performed per block:

1. **Emphasize local peaks**
   First, all local energy peaks within each frame of the block are emphasized. This is realized by subtracting the running average from the power spectrum[1] of each frame, using a window size of $N = 21$ frequency bins:

$$X_i^{emph} = X_i - \frac{1}{N} \sum_{k=-\frac{N}{2}}^{\frac{N}{2}} X_{\min(\max(k,1),N)} \tag{5.4}$$

   where X_i denotes the energy of the i-th frequency component of the current frame. This step is useful to emphasize very soft tones, belonging to background music: if a soft tone is not masked by another signal over its entire duration (which is unlikely, as non-music signals tend to be less stationary), the perceivable horizontal bars in the spectrogram are compositions of consecutive local maxima. The idea is to emphasize these soft bars by emphasizing all local maxima in the spectrum of a frame.

2. **Binarization**
   Then, to ignore the absolute strength of activation (energy) in a given frame j within the block, each frequency component X_{ij}^{emph} is binarized by comparing to a fixed binarization threshold. The binarization threshold $t = 0.1$ was chosen in such a way that even soft activations could be kept in the binarized block. Only frequency bins which are obviously not active at all, will be set to 0 using this low threshold.

$$B_{ij} = \begin{cases} 1 & X_{ij}^{emph} > t \\ 0 & X_{ij}^{emph} \leq t \end{cases} \tag{5.5}$$

   Neglecting the actual strength of the activation allows to focus on structural aspects of the emphasized spectrogram only.

[1]In contrast to the other block-level features for the CFA the power spectrum is used instead of the normalized Cent-Spectrum.

3. **Computation of the frequency activation**
   For each block the frequency activation function $Activation(i)$ is computed. For each frequency bin i, the frequency activation function measures how often a frequency component is active in a block. To obtain the frequency activation function of a block all binarized frequency bins B_{ij} are simply summed along the time axes. Thus, the frequency activation function contains temporal information over many frames that cannot be captured by simple frame-level features.

   $$Activation(i) = \frac{1}{F}\sum_{j=1}^{F} B_{ij} \tag{5.6}$$

   Figure 5.5 shows the binarized emphasized power spectra of two blocks and the resulting frequency activation functions. Subplot *(b)* is typical of blocks containing music, whereas subplot *(a)* is representative of blocks without any musical elements.

4. **Detect strong peaks**
   Strong peaks in the frequency activation function of a given bock indicate steady activations of narrow frequency bands. The *"spikier"* the activation function, the more likely horizontal bars, which are characteristic of sustained musical tones, are present. Even one large peak is quite a good indicator for the presence of a tone. The peakiness of the frequency activation function is consequently a good indicator for the presence of music. To extract the peaks a simple peak picking algorithm extracts the height of each peak [**?**], called *peak value*.

5. **Quantify the Continuous Frequency Activation**
   To quantify the *Continuous Frequency Activation* of the activation function of a block, the peak values of all detected peaks are sorted in descending order, and the sum of the five largest peak values is taken to characterize the overall *"peakiness"* of the activation function.

As a result of this block-level extraction process we obtain exactly one numeric value for each block of frames, which quantifies the presence of steady frequency

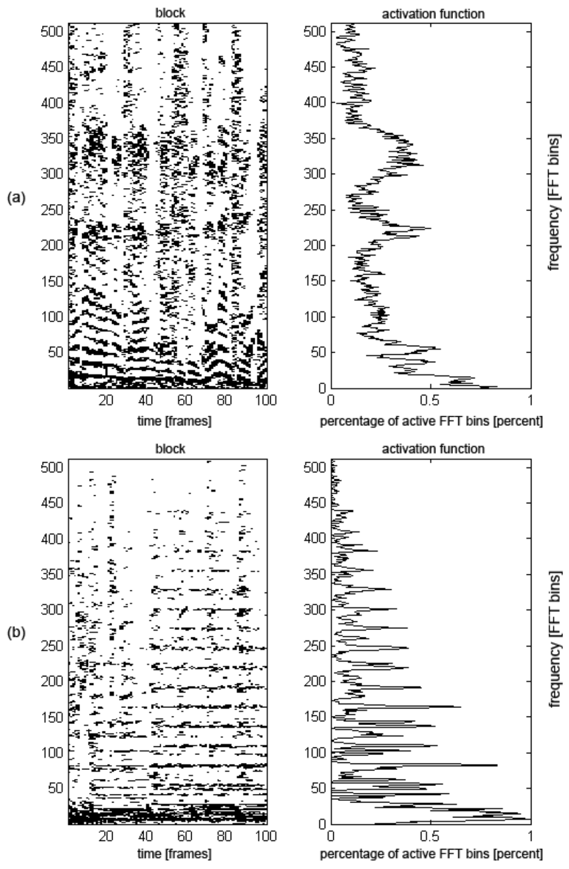


Figure 5.3: Binarized spectrogram of a block and the corresponding activation function. Block **(a)** contains no music, whereas in block **(b)** music is present.

components within the current audio segment. For blocks containing music the resulting value should be higher than for blocks where no music is present, which makes the CFA a perfect feature for music detection. However, this is not the only application area of this feature. By using the median as summarization function over the per block CFA values one obtains a single scalar value that quantifies if a song contains many tone-like components. This scalar value can be used for sorting and retrieval in the same way as the more advanced H2A ratio [?], which extends this idea.

5.2.8 Summary

Together the proposed feature set forms a high-dimensional and powerful description of an audio track. One advantage of this feature set is that it allows to visualize song models. Figure 5.3 visualizes the proposed block-level features — except the CFA, which is a single scalar and not intended for visualization — for two different songs, a Hip-Hop and a Jazz song. Another important aspect of the proposed feature set is that the local features are *not* summarized via a distribution model. Instead, all the presented block-level features can be interpreted as vectors in a vector space. This is a clear advantage of the proposed feature set as the vector space representation allows to take advantage of the whole mathematical toolbox that is available for vector spaces. Consequently, the proposed block-level features can be used in a straightforward way with any standard machine learning algorithm or with any standard dimensionality reduction method. This is for example especially useful with respect to *automatic genre classification* or *automatic tag prediction*, where the block-level features can be used in a straightforward way. However, it is not as straightforward to directly define a high-quality *similarity measure* based on these block-level features. For this reason it will now be discussed how to combine six of the proposed block-level features into a single similarity function.

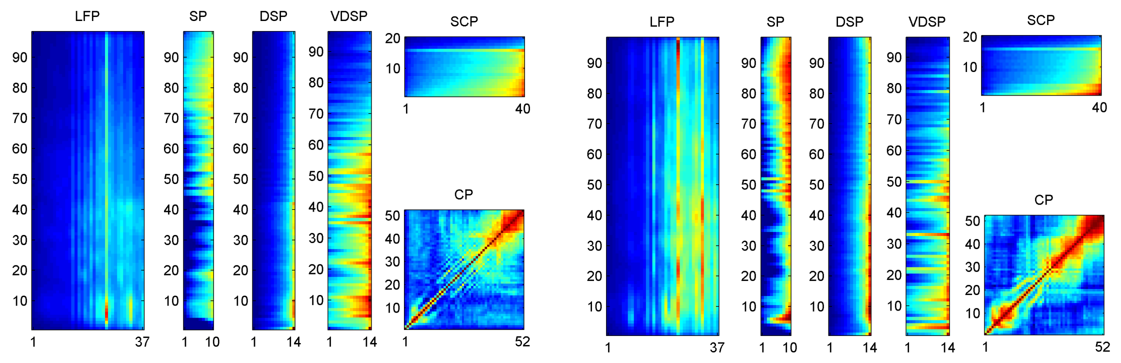


Figure 5.4: Visualization of the proposed block-level features for a Hip-Hop song (left) and a Jazz song (right).

5.3 Music Similarity Estimation based on Block-Level Features

In the previous section seven block-level features have been introduced. In this section a similarity function is defined based on six of them (SP, DSP, VDSP, LFP, CP, SCP). To define a music similarity measure a linear combination approach is used: First, a similarity function is chosen for each pattern individually. In our case for the six block-level features the *Manhattan distance* is used. Then, the distance estimates of the individual patterns are summed to yield a combined distance measure. However, combining the distances of the individual patterns is not as straightforward as it seems. One main problem is that the patterns themselves have different scales and consequently produce distances in completely different numerical ranges. To reduce the influence of the individual numeric scales of the distance estimates a special normalization strategy, called *distance space normalization* is used, which is an important component of the proposed similarity algorithm and will be presented in detail in the following subsection.

5.3.1 Distance Space Normalization

The proposed *distance space normalization* (DSN) is a modified variant of the normalization approach that is used in [Pohle and Schnitzer, 2007], but allows for a more intuitive interpretation. Each distance of a distance matrix $D_{n,m}$ is normalized by subtracting the mean and dividing by the standard deviation (Gaussian normalization) over all distances in row n and column m (see figure 5.6). Thus, each distance between two songs n and m has its own normalization parameters, as all distances to song m and all distances to song n are used for normalization. This way the normalization operation can also change the ordering within a column / row. Interestingly, the ordering after the DSN can be drastically different from the original ordering. Figure **??** shows a histogram of the change of the column ranks over the whole distance matrix. Surprisingly, this change does typically have a positive influence on the nearest neighbor classification accuracy as reported in

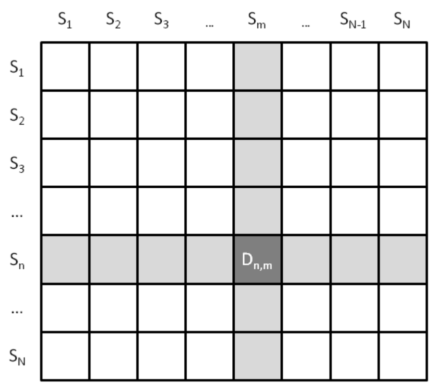


Figure 5.5: Distance Space Normalization of a distance matrix.

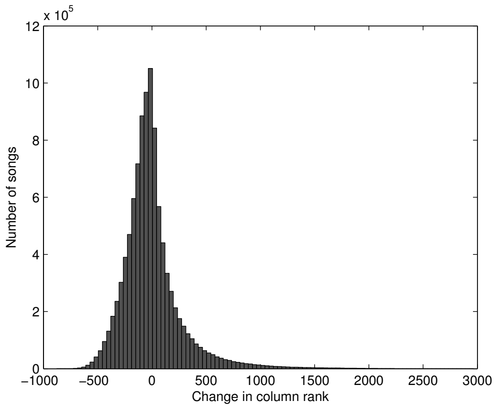


Figure 5.6: Change in column ranks as a consequence of the Distance Space Normalization (DSN). Evaluated on the *"1517-Artists"* dataset for the Spectral Pattern.

[Pohle et al., 2009]. This observation is confirmed by the classification results of the individual components of the BLS algorithm before and after distance space normalization, which are summarized in table 5.1.

Furthermore, the results in table 5.1 also show that the combination of the individual components clearly outperforms the individual components. Additionally, it turns out that removing any of the combined block-level features would reduce the classification accuracy of the combined similarity measure. Thus, we can conclude that all the components contribute to the overall similarity. It is also worth mentioning that several other normalization approaches to combine the individual components have been evaluated, but none outperformed the DSN approach described here with respect to the achieved recommendation quality. However, this normalization method has also a major disadvantage, which is the high runtime complexity of $O(N^2)$.

5.3.2 Extended Distance Space Normalization

Based on the above mentioned observation that the DSN approach all alone can help to improve nearest neighbor classification accuracy, the DSN based combination approach is extend by another final normalization step. Thus, the resulting distance matrix is once more normalized after the individually weighted components have been combined. This results in an improvement of the classification accuracy from 36.49% to 37.69% (see Table 5.1). The overall combination method of the BLS algorithm is visualized in figure 5.7 and is called *extended distance space normalization* (EDSN). The next subsection will discuss how the component weights and the parameter settings of the block-level features have been obtained.

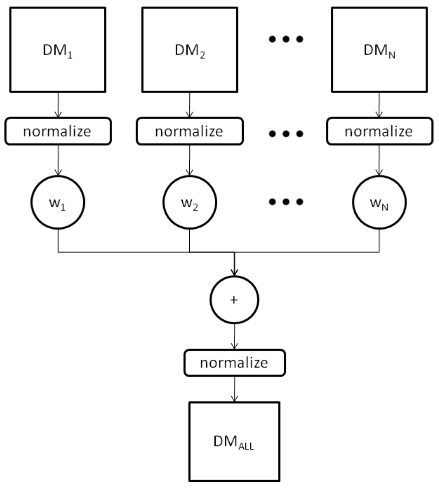


Figure 5.7: Schematic structure of the combination approach.

5.3.3 Optimization of the Similarity Measure

The features presented in section 5.2 come with a large set of parameters. For example, one can compute these features for different block sizes, hop sizes, different percentiles and other parameters. To identify suitable parameter settings for the individual components and find an optimal weighting of the components these settings were optimized on dataset *"1517-Artists"* (see 3.2.3.5). Each block-level feature was optimized separately. Unfortunately, a complete evaluation of all reasonable parameter combinations was impossible, as each parameter setting requires to recompute the feature on the whole collection, which was not tractable computationally. Therefore, the optimization procedure was to vary each parameter separately one after the other and pick the optimum. The *"optimal"* parameter settings found by this still expensive optimization are those reported in section 5.2.

Component	**Weight**	**10-NN CA**	**10-NN CA (DSN)**
LFP	1.0	26.70%	28.75%
SP	1.0	26.76%	26.82%
SCP	1.0	19.31%	20.47%
CP	1.0	23.36%	26.83%
VDSP	0.8	21.44%	24.18%
DSP	0.9	23.18%	25.28%
Combined		36.49%	37.69%

Table 5.1: Component weights of the optimized similarity measure and the respective nearest neighbor classification accuracies on dataset *"1517-Artists"* of the components before and after distance space normalization (DSN).

Furthermore, also the weights of the individual components are found by a greedy optimization: Initially, an equal weight is assigned to each component ($w_i = 1$). Then the weight of a randomly chosen component is reduced, if this results in an improvement in classification accuracy. This step is repeated until no more improvement is possible. The final weights that result from this optimization are summarized in table 5.1 and represent the final configuration of the proposed block-level similarity measure (BLS). The recommendation quality of the BLS algorithm is investigated in detail in section 5.5 of this chapter. In the following section a second music similarity measure that is based on the proposed block-level features is introduced. In contrast to the BLS algorithm, where a distance function is defined directly based on the underlying features, the algorithm that is presented in the following section performs a mapping onto a semantic tag space before estimating song similarities.

5.4 Music Similarity Estimation based on Auto-Tags

In this section a second similarity measure that is based on the proposed block-level features is defined. In contrast to the BLS algorithm, where musical similarities are estimated via directly comparing the individual block-level features, prior to computing similarities among songs a mapping onto a semantic tag space is performed. Such an approach is called *semantic space approach*, as songs are no longer represented by the underlying features, but via semantic concepts instead.

5.4.1 The Semantic Space Approach

Tags are semantic textual annotations like for example *"beat"*, *"fast"* or *"rock"* that are used to describe songs. Together all tags that are associated with a song form a semantic description and can then for example be used to search and browse the emerging semantic music space, which is called *tag-based browsing*. Of course tag information can also be used to generate music recommendations e.g. by recommending songs with similar tag annotations. While tags are typically collected by large online music platforms like for example Last.fm[2] that allow users to annotate the songs they are listening to, there exist several other methods

[2]www.last.fm

to collect tag information [**?**]. Tags can also be obtained through surveys, music annotation games or web-mining. Another variant, which is used here as this thesis is focusing on content-based methods only, is to obtain tag information via *auto-tagging*. An auto-tagger is a purely content-based method that predicts tags which might be associated with a song given a set of audio features extracted out of the audio signal. Consequently, an auto-tagger transforms an audio feature space into *a semantic space*, where music is described by words, via predicting tags. While automatic tag prediction recently gained a lot of research attention and can be considered an emerging research area in Music Information Retrieval, the idea of predicting tags is relatively old. To the best of our knowledge Berenzweig [**?**] was the first to introduce a concept related to auto-tags called *"anchor space"* and he also used this information to compute music similarities among songs. However, it seems that in 2003 the lack in computational resources and the unavailability of adequate tag sources limited further development in this research direction. Then driven by the general growing interest in tags in the MIR community in 2006 this idea was picked up again by West [West et al., 2006] and then by Turnbull [Turnbull et al., 2008], who introduced a first formal definition of the tag prediction task.

The task of predicting tags can be interpreted as a special case of multi-label classification and can be defined as follows: Given a set of tags $T = \{t_1, ..., t_A\}$ and a set of songs $S = \{s_1, ..., s_R\}$ predict for each song $s_j \in S$ the tag annotation vector $y = (y_1, ..., y_A)$, where $y_i > 0$ if tag t_i has been associated with the audio track by a number of users, and $y_i = 0$ otherwise. Thus, the y_i's describe the strength of the semantic association between a tag t_i and a song s_j and are called *tag affinities* or *semantic weights*. If the semantic weights are mapped to $\{0, 1\}$, then they can be interpreted as class labels, which can be used for training and evaluating tag classifiers.

In the following it will be first investigated if the proposed block-level features are useful with respect to tag classification by comparing the results of a state-of-the-art auto-tagger using a standard feature set and the proposed block-level feature set. Then in subsection **??** a music similarity measure is proposed that is solely based on automatically predicted tags.

5.4.2 Block-Level Feature for Automatic Tag Prediction

In this section it will be demonstrated that the proposed block-level feature set is also suitable for the task of automatic tag prediction. To show this, the proposed feature set is compared to a standard audio feature set (SAF), which is introduced in the next subsection. To compare these two feature sets exactly the same tag classification approach is used for both sets. A two stage classification approach called *"stacked generalization"* [Ness et al., 2009] is used in our experiments. This procedure is a state-of-the-art tag classification approach and ranked second (with respect to the per tag f-score) in the 2009 run of the MIREX tag classification contest using the standard audio feature set (SAF) that will also be used in our feature set comparison. Furthermore, this method was found to be only insignificantly worse than the leading algorithm and it is already implemented and publicly available via the MARSYAS (Music Analysis, Retrieval and Synthesis for Audio Signals)[3] open source framework. For this reason the auto-tagging algorithm implemented by the MARSYAS framework is a perfect candidate for a comparison.

Unfortunately, the proposed block-level feature set cannot be used as it is for tag classification, because of the high dimensionality of this feature set. Together the block-level feature form a feature vector of 9448 dimensions. As the runtime of classification algorithms typically increases drastically with the dimensionality of the feature vectors, directly using the block-level feature set is computationally not tractable. For this reason in subsection 5.4.3.2 it is proposed to compress the block-level feature set. In the following the standard audio features that are used in the comparison and the tag classification datasets are briefly introduced.

5.4.2.1 *"Standard"* Audio Features

In the conducted experiments the described block-level feature set is compared to a standard feature set (SAF) that can easily and efficiently be extracted by the

[3]www.marsyas.info

MARSYAS framework. This standard feature set consists of the following feature: the *Spectral Centroid*, the *Rolloff*, the *Flux* and the M*el*-F*requency* C*epstral* C*oefficients* (MFCCs). Altogether, 16 numbers are extracted per audio frame. To capture some temporal information a running mean and standard deviation over a *texture window* of M frames is computed. The resulting intermediate features of the running mean computation still have the same rate as the original feature vectors. To come up with a single feature vector per song the intermediate running mean and standard deviation features are once more summarized by computing mean and standard deviation thereof. The overall result is a single 64-dimensional feature vector per audio clip. A more detailed description can be found in [Tzanetakis and Cook, 2002].

5.4.2.2 Datasets and Performance Measures

To assess the quality of the predicted auto-tags two datasets are used in the evaluation: the *CAL500* and the *Magnatagatune* dataset.

The CAL500 dataset [Turnbull et al., 2008] consists of 500 Western popular songs by 500 different artists. These songs have been annotated by 66 students with predefined semantic concepts that relate to six basic categories: instruments, vocal characteristics, genres, emotions, preferred listening scenarios and acoustic qualities of a song (e.g. tempo, energy or sound quality). These concepts were then mapped to a set of 174 tags including positive and negative tags. Based on the user data binary annotation vectors were derived by ensuring a certain user agreement on the assigned tags.

The second dataset in our evaluation is the Magnatagatune [?] dataset. This huge dataset contains 21642 songs annotated with 188 tags. The tags were collected by a music and sound annotation game, the TagATune[4] game. The dataset also contains 30 seconds audio excerpts of all songs that have been annotated by the players of the game. All the tags in the dataset have been verified (i.e. a tag is

[4]http://www.tagatune.org

associated with an audio clip only if it is generated independently by more than 2 players, and only tags that are associated with more than 50 songs are included).

As a common performance measures for tag classification the well-known *f-Score* is used. Additionally, as a second performance measure the *G-mean* is used. The G-mean (**??**) [**?**] is defined as the combination of *Sensitivity* (Acc^{+}), also known as true positive rate, and *Specificity* (Acc^{-}), also known as true negative rate. As such, it is a nice and compact measure that has the advantage of taking the class imbalance into consideration. This is important as the class distributions of tag classification datasets are often extremely unbalanced. Both measures can be computed globally over the entire (global) binary tag classification matrix, or separately for each tag and then averaged across tags. To differentiate between global and averaged performance measures the averaged per tag measures are named *avg. F-Score* and *avg. Gmean*, respectively.

$$\text{f-Score} = \frac{2 \times \text{precision} \times \text{recall}}{(\text{precision} + \text{recall})} \tag{5.7}$$

$$\text{Acc}^{-} = \text{TN}/(\text{TN} + \text{FP}) \tag{5.8}$$

$$\text{Acc}^{+} = \text{TP}/(\text{TP} + \text{FN}) \tag{5.9}$$

$$\text{G-mean} = (\text{Acc}^{-} \times \text{Acc}^{+})^{\frac{1}{2}} \tag{5.10}$$

5.4.2.3 Reducing the Dimensionality of Block-Level Features

To make the block-level features applicable to the task of automatic tag classification, their dimensionality has to be reduced in order to make the classification computationally tractable. The dimensionality reduction is realized by a standard *Principal Component Analysis* (PCA) [Bishop, 2006]. Obviously, both the compression rate and the achievable classification quality depend on the number of principal components used to represent each block level-feature. An evaluation is conducted to determine the number of principal components for each pattern (LFP, SP, DSP, SCP, CP, VDSP) individually. The total variance captured by the k most important principal components serves as criterion for the number of principal components to represent each pattern. For example, given that 80% of the total variance should be kept, we compute the PCA for each pattern and then keep the number of principal components so that at least 80% of the total variance is accounted for. Thus, the prespecified variance determines the resulting feature size for each pattern individually.

% of total variance	**dimensions (fold 1 / fold 2)**	**f-Score**	**G-Mean**
0.65	37 / 36	0.4863	0.6651
0.70	49 / 47	0.4911	0.6687
0.75	66 / 64	0.4965	0.6727
0.80	89 / 88	0.4983	0.6740
0.85	126 / 124	0.4849	0.6715
0.90	190 / 189	0.4895	0.6675
0.95	321 / 316	0.4886	0.6668
0.99	655 / 647	0.4678	0.6511
uncompressed	9448	0.5015	0.6764

Table 5.2: G-Mean and f-Score performance results of the stacked generalization tag classification approach for various compression levels (CAL500)

To evaluate the PCA compression for different percentages of the total variance the CAL 500 dataset and two-fold cross-validation were used. To prevent possible overfitting effects the principal components were estimated on the features of the training set only. As a consequence, the dimensionality of the compressed feature set differs for the two cross-validation folds. All experiments were carried out for the same split into two cross-validation folds. The evaluation results are summarized in table 5.2. One can see that a feature set capturing about 70% to 80% of the total variance seems optimal in terms of tag classification quality. Interestingly, even an extreme reduction of the feature space to only about 37 dimensions performs comparably well. The best classification performance, however, is achieved by the uncompressed feature set. It is also important to note that the decay in classification quality with a high number of principal components is related to the low number of data points that are available for the projection: the CAL500 consists of only 500 songs, in a feature space with 9448 dimensions. Altogether, one can conclude from these experiments that the proposed PCA compression approach does not diminish the tag prediction quality too much and is therefore a reasonable approach to reduce the size of the feature space of the block-level features.

5.4.2.4 Comparison of Feature Sets

To compare the two feature sets we report on the presented performance measures obtained via 2-fold cross-validation on two different tag classification datasets (CAL500 and Magnatagatune). The same cross-validation split was used for the evaluated feature sets. These results are summarized in table 5.3. Both, the global performance measures computed over the global binary classification matrix, and the averaged per-tag performance measures are reported. SAF denotes the standard audio feature set and BLF-PCA denotes the PCA compressed block-level feature set. BLF-FULL denotes the result of the uncompressed block-level feature set, which we only report for the smaller CAL500 dataset, because it was computationally not tractable on the larger Magnatagatune dataset. On the CAL500 dataset the BLF-PCA feature set consists of the 75 most important principal components capturing 75% of the total variance. On the larger Magnatagtune dataset the same variance threshold of 75% was used. For each performance measure the highest score on each dataset is highlighted in bold face. [5] From table 5.3 one can see that independent of the overall performance measure, either global or averaged per tag, the compressed block-level feature set compares favorably to the standard feature set. Summarizing these results we can conclude that block-level features are suitable for automatic tag prediction. Based on this insight, in the following, two music similarity algorithms are proposed that make use of the predicted tag-information.

[5]It is worth mentioning that only the tag classification results of the second classification stage of the stacked generalization tag classification approach are reported here. More detailed results can be found in [**?**].

feature set	dataset	f-Score	G-Mean
SAF	CAL500	0.4582	0.64387
BLF-FULL	CAL500	**0.5015**	**0.67641**
BLF-PCA	CAL500	0.4965	0.6727
SAF	Magnatagatune	0.3962	0.6252
BLF-PCA	Magnatagatune	**0.4201**	**0.6439**

feature set	dataset	avg. f-Score	avg. G-Mean
SAF	CAL500	0.2577	0.3851
BLF-FULL	CAL500	**0.3061**	**0.4454**
BLF-PCA	CAL500	0.2908	0.4217
SAF	Magnatagatune	0.1932	0.3784
BLF-PCA	Magnatagatune	**0.2185**	**0.4081**

Table 5.3: Comparison of standard audio features (SAF) and block-level features (BLF) for the stacked generalization tag classification approach.

5.4.3 Similarity Estimation using Auto-Tags

The idea of estimating similarities between songs by comparing their auto-tag profiles traces also back to Berenzweig [?] and to web-retrieval techniques, where term frequency vectors are often used to derive document similarities. Recently, several music similarity algorithms based on auto-tags have been proposed [Bertin-Mahieux et al., 2008] and have been evaluated during the MIREX competition [?, ?, ?]. In contrast to these algorithms, which are mostly based on standard audio features, the two algorithms proposed in this section are based on tag and genre affinities that are estimated using the block-level features as described in the previous section. The only difference to the previously presented auto-tagging algorithm is that the stacked generalization classification approach of the MARSYAS framework is replaced by the support vector machine implementation of the machine learning toolbox WEKA. This will allows us to evaluated different types of classifiers in future.

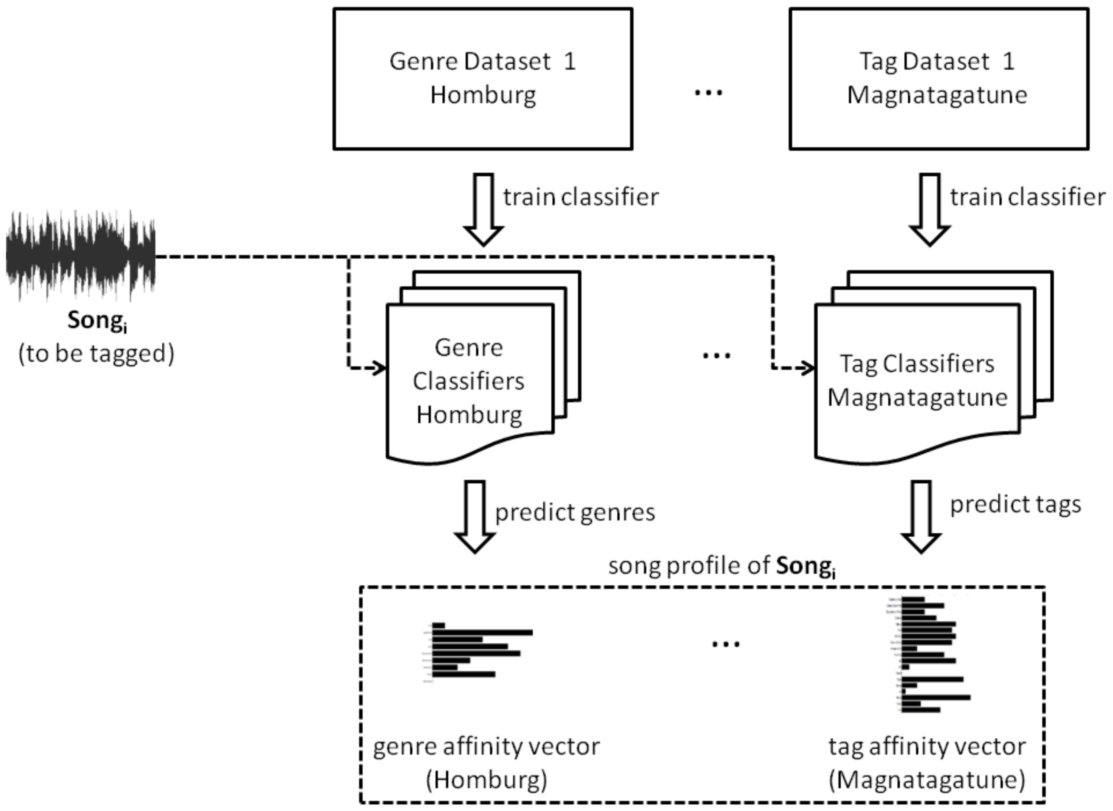


Figure 5.8: Automatically predicting tags learned from different datasets.

For the auto-tag based similarity algorithms the auto-taggers that are used to create the song profiles are constructed as follows: For each of the two tag classification dataset Magnatagatune and CAL500 a set of tag classifiers is learned. Additionally, one genre classifier is trained for each genre classification dataset. Depending on the dataset these classifiers of course predict different genre labels. Besides the datasets presented in section 3.2.3 four additional genre classification datasets are used to train genre classifiers. The tag and genre classifiers are then used to automatically tag songs. The probabilistic predictions of a set of tag classifier are called *tag affinity vector*. The probability estimates of a genre classifier are called *genre affinity vectors* respectively. Together all predicted tag affinity vectors and genre affinity vectors for a song form the *song profile*. To generate song profiles for a whole collection of files, for example the *"1517-Artists"* dataset, all files in the collection are tagged using all tag and genre classifiers, except the genre classifiers that have been trained on the respective dataset itself. The whole process is illustrated in figure ??.

The two variants to compute tag based similarity (TAG and TAGVS) that are proposed here then only differ in the way the genre affinity vectors and tag affinity vectors resulting from the different classifiers are combined into a single music similarity measure.

5.4.3.1 Block-Level Feature based Auto-Tags (TAG)

The first variant makes use of the *extended distance space normalization* (EDSN) approach (see 5.3.1) to combine the affinity vectors generated by different classifiers. Similarities among songs are estimated separately for the affinity vectors produced by a single classifier. This is realized by computing the Manhattan distance among each pair of affinity vectors and results in one distance matrix for each genre or tag classifier. Considering the example in figure ?? the genre affinity vectors of dataset *"Homburg"* would be used to compute a distance matrix. Another distance matrix would be constructed for the *"Magnatagatune"* tag affinity vectors. The obtained distance matrices are then fused into a single result distance matrix using the EDSN approach. Thus, as for the BLS algorithm the main disadvantage of this approach is the quadratic runtime complexity of the EDSN combination.

5.4.3.2 Block-Level Feature based Auto-Tags Vector Space Model (TAGVS)

The second variant of the tag based music similarity algorithm does not combine the affinity vectors via the EDSN method. Instead the affinity vectors of all classifiers are simply concatenated into a single long feature vector before any distances among songs are computed. As the number of probability estimates per dataset depends on the number of genre or tag classifiers per dataset, the resulting affinity vectors are normalized by dividing by the number of classifiers before they are combined. This way the affinity estimates from each dataset equally contribute to the overall similarity estimate. Thus, each song is represented by a single feature vector. For the example visible in figure ?? the genre affinity vector of dataset *"Homburg"* and the *"Magnatagatune"* tag affinity vector would be concatenated into a single feature vector. The overall similarity is then estimated as the Manhattan distance among these concatenated feature vectors. The main advantage of this approach is that no distance space normalization is used. Consequently, this variant can be used in combination with any advanced indexing and search strategies and is so well-suited even for large datasets.

In this section a music similarity measure that is directly based on the proposed block-level features (BLS) and two variants based on auto-tags (TAG and TAGVS) have been defined. In the following section these algorithms are evaluated and compared to a selection of prominent music similarity algorithms.

5.5 Comparison of Music Similarity Algorithms

In this section the three proposed block-level music similarity algorithms (BLS, TAG and TAGVS) are evaluated and compared to a selection of prominent music similarity algorithms, which are presented in subsection ??. Then results of the evaluation are presented and discussed in subsection ??. Thereafter, to identify any further potential for improvement, combinations of the three best performing algorithms in the evaluation are investigated. Besides the results from this in-house evaluation, in subsection ?? the results of the 2010 run of the public MIREX evaluation are discussed, as block-level feature based algorithms were submitted by the author in three categories: *Audio Music Similarity and Retrieval*, *Audio Genre Classification* and *Automatic Tag Classification*.

5.5.1 Algorithms

In this evaluation eight different music similarity algorithms are compared. First of all, there are two by now *"classic"* algorithms — the Single Gaussian (SG, see 4.2.2) algorithm and the G1C algorithm. Furthermore, two *"state-of-the-art"* (RTBOF, MARSYAS) algorithms are included in the evaluation. Then there are the three block-level feature based algorithms (BLS, TAG and TAGVS). Additionally, a random similarity measure (RND) is used to indicate the baseline accuracy for the various datasets. Three out of these algorithms (G1C, RTBOF and MARSYAS) have not yet been presented and are briefly introduced in the following.

5.5.1.1 G1C: A combination of Single Gaussian and Fluctuation Patterns

The G1C algorithm [Pampalk, 2010] is a linear combination of a rhythm component, the Fluctuation Patterns, three descriptors derived from the rhythm component (*Bass*, *Gravity* and *Focus*) and a timbre component, the Single Gaussian model. The G1C algorithm was one of the first algorithms that combined two aspects of musical similarity in a single algorithm. In 2006 this algorithm ranked first in the MIREX Audio Music Similarity and Retrieval task. The major drawback of this algorithm is the simple linear combination approach. In our evaluation the G1C and the SG algorithm represent the by now classic similarity algorithms that are used to evaluate the qualitative improvement over the years.

5.5.1.2 Rhythm-Timbre Bag of Features (RTBOF)

The *Rhythm-Timbre Bag of Features* approach (RTBOF) is a recent music similarity measure proposed by Pohle et al. in [Pohle et al., 2009]. This measure ranked first in the MIREX 2009 music similarity and retrieval task and has proven to be statistically significantly better than most of the participating algorithms. Thus, it reflects the current state-of-the-art in music similarity estimation. Basically, it has two components – a rhythm and a timbre component similar to the G1C. In contrast to the G1C algorithm these two components are combined using a distance space normalization approach (see ??). Each component consists of a distribution model over local spectral features. The local audio features, described in [Pohle et al., 2009], are complex and also incorporate some local temporal information over several frames. Because of its components this approach is called *Rhythm-Timbre Bag Of Features* (RTBOF).

5.5.1.3 Marsyas (MARSYAS)

The MARSYAS (Music Analysis, Retrieval and Synthesis for Audio Signals) framework[6] is an open source software that can be used to efficiently calculate various audio features. For a detailed description of the extracted features we refer to [Tzanetakis and Cook, 2002]. This algorithm has participated in the MIREX Audio Similarity and Retrieval task in 2007 and 2009 (there was no such task in 2008), taking the 2nd and 7th ranks, respectively. The features as well as the similarity computation were the same in 2007 and 2009. Consequently, this algorithm is also useful to monitor the progress in music similarity estimation over the years.

5.5.2 Comparison of Algorithms

In this large-scale evaluation eight music similarity algorithms are evaluated on seven different datasets (see 3.2.3). For each audio file the same song excerpt is analyzed by all algorithms. Up to a maximum of 180 seconds of audio data are extracted from the middle of each track for analysis. All input signals are resampled to 22 kHz. Only one audio channel is used. As quality indicators the *k-Neighbour Accuracy* (k-NA) and the *Classification Accuracy* that is obtained using a k-NN classifier (k-NN CA) are reported (see 3.2.1). The reported genre classification accuracies of the k-NN classifier are averaged over 10 runs of a stratified 10-fold cross-validation. For both the k-NA and the k-NN CA k is varied from 1 to 20. Figure ?? visualizes the results of this evaluation for the k-NN CA and figure ?? visualizes the results obtained using the k-NA quality measure. As we are in this

[6]http://marsyas.info

evaluation basically interested in non-trivial recommendations that would also be useful in a real world scenario, artist filtered results are reported for those datasets where artist information is available (*"1517-Artist"*, *"Homburg"*, *"Unique"*). The other datasets are still useful to make the reported results more comparable to previous work and point out the difference to artist filtered results.

The first outcome of this evaluation is that all similarity algorithms clearly outperform the random baseline for both quality measures. Regarding the baselines one can observe that for the k-NN CA the random baseline stays constant for datasets *"1517-Artists"* and *"GTZAN"*, while it increases on the other datasets. This increase is related to the imbalanced class distributions of the respective datasets. With increasing k, the random baseline of the k-NN classifier grows towards the accuracy obtainable by always predicting the most frequent genre.

Furthermore, comparing the results obtained on the datasets to the left hand side of figure **??** to the classification results on the datasets to the right hand side, one can see that the classification accuracies are basically decreasing in the plots to the left and increasing in the plots to the right. One can see that this behavior is clearly related to the artist effect. For datasets *"1517-Artist"*, *"Homburg"* and *"Unique"* we do not expect any artist or album effect because of the applied artist filter, whereas for the other datasets we expect an artist effect (due to the nature of the music collections). This phenomenon can be explained by the fact that for datasets with an artist effect the first nearest neighbor of a given query song will likely be a song by the same artist. As typically songs by the same artist belong to the same genre the first nearest neighbor is an almost perfect genre predictor. For the datasets without artist effect, we can clearly observe the expected learning curve, where the k-NN classifier profits from considering an increasing neighborhood. Thus, we can conclude that from an analysis of the nearest neighbor classification results one can even detect the presence of a strong artist or album effect.

Considering the results of the individual algorithms, the SG, the G1C and the MARSYAS algorithms perform significantly worse than the other algorithms on all datasets. This is in line with our expectations and illustrates the progress in music similarity estimation over the last years. What jumps to the eye is the different behavior of the G1C algorithm compared to the other algorithms. This untypical behavior seems to be related to the simple combination strategy of the G1C model, which is not able to adapt to the specific properties of a dataset. For the other four algorithms BLS, RTBOF, TAG and TAGVS a qualitative difference is not so obvious. In general the RTBOF algorithm outperforms the others on those datasets where no artist filter is used. However, on the artist filtered datasets in turn the BLS, TAG and TAGVS algorithms outperform the RTBOF algorithm. So one can conclude that the RTBOF algorithm is more adequate for music similarity estimation, but with respect to music recommendation and identification of non trivial song relationships the block-level feature based variants outperform the RTBOF algorithm. Of course it is also important to note in this context that the BLS algorithm has been optimized on the *"1517-Artists"* dataset, while the RTBOF algorithm has been optimized on the *"ballroom"* dataset. For both algorithms the obtained results reported on their optimization datasets are superior, which might be due to over-fitting effects. Another finding is that the auto-tag based similarity algorithm (TAG) using the DSN approach, yields higher results on most of the analyzed datasets compared to the vector space variant (TAGVS). However, the difference is not huge. Consequently, the TAGVS approach is an adequate choice for extremely large music collections, where search and retrieval times are a critical issue.

Regarding the different quality indicators it is important to note that they are not always consistent. The first inconsistency that can be observed is among quality indicators of different neighbourhood sizes. For example the RTBOF algorithm yields a classification accuracy of 88.26% on the *"ismir2004all"* dataset, while the BLS algorithm reaches only 87.67%, when considering the 1-NN. However, when the 20-NN is considered one obtains the inverse result and the BLS algorithm reaches 83.6%, while the RTBOF algorithm reaches 82.06%. Furthermore, one can observe another inconsistency that exists between the k-NN CA quality indicator and the k-NA quality indicator. For example the 20-NN CA quality measure indicates that the BLS algorithm (38.62%) outperforms the TAGVS algorithm (37.02%). The 20-NA measure, however, indicates the opposite, reporting a neighbour accuracy of 25.26 for the TAGVS algorithm and a neighbour accuracy of 24.43 for the BLS algorithm. As the results of the evaluation depend on the chosen quality indicator, one can conclude that a sound comparison of algorithms should always report various quality indicators so that existing inconsistencies in the evaluation results become at least obvious.

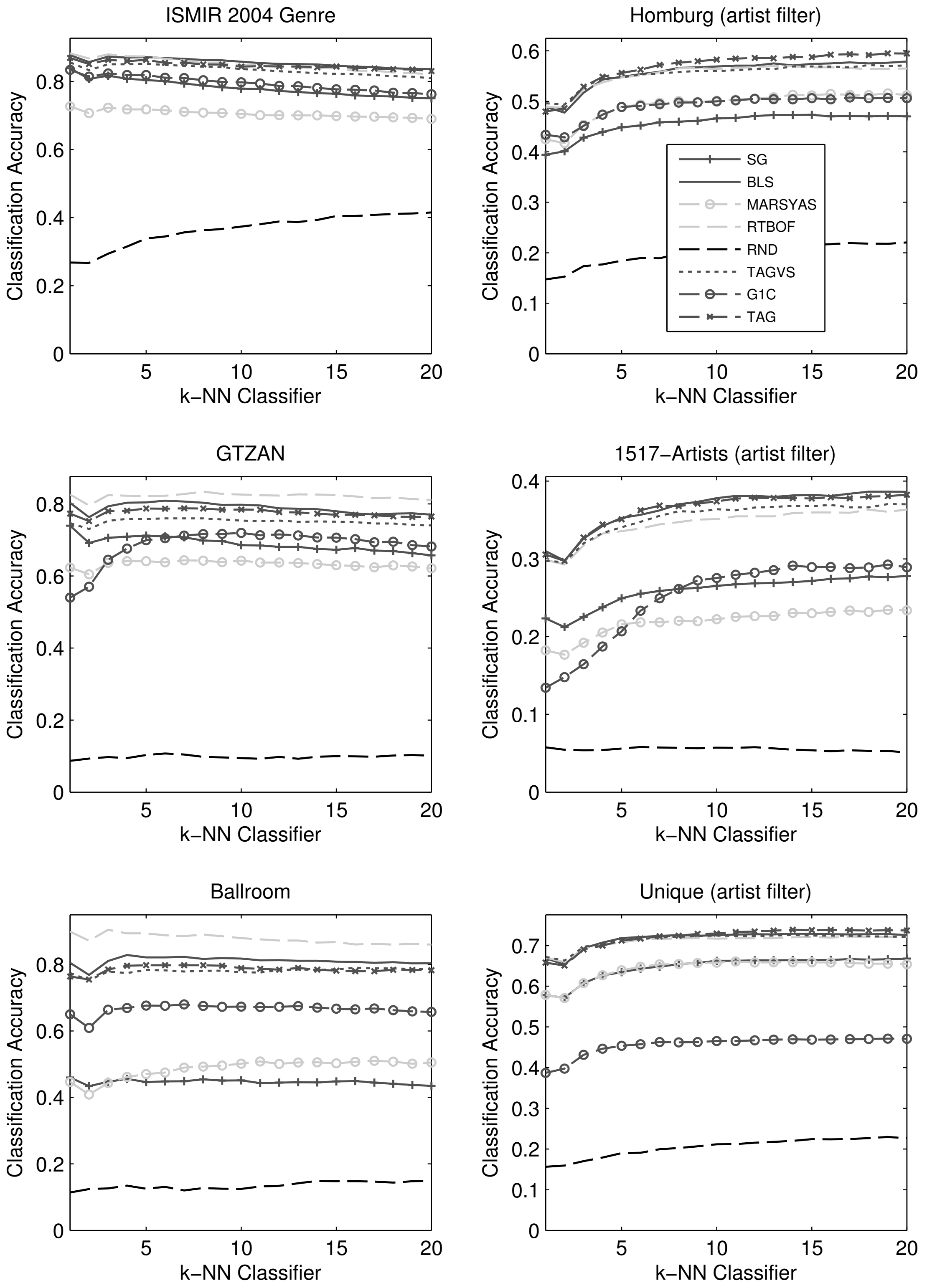


Figure 5.9: Comparison of music similarity algorithms (k-NN CA).

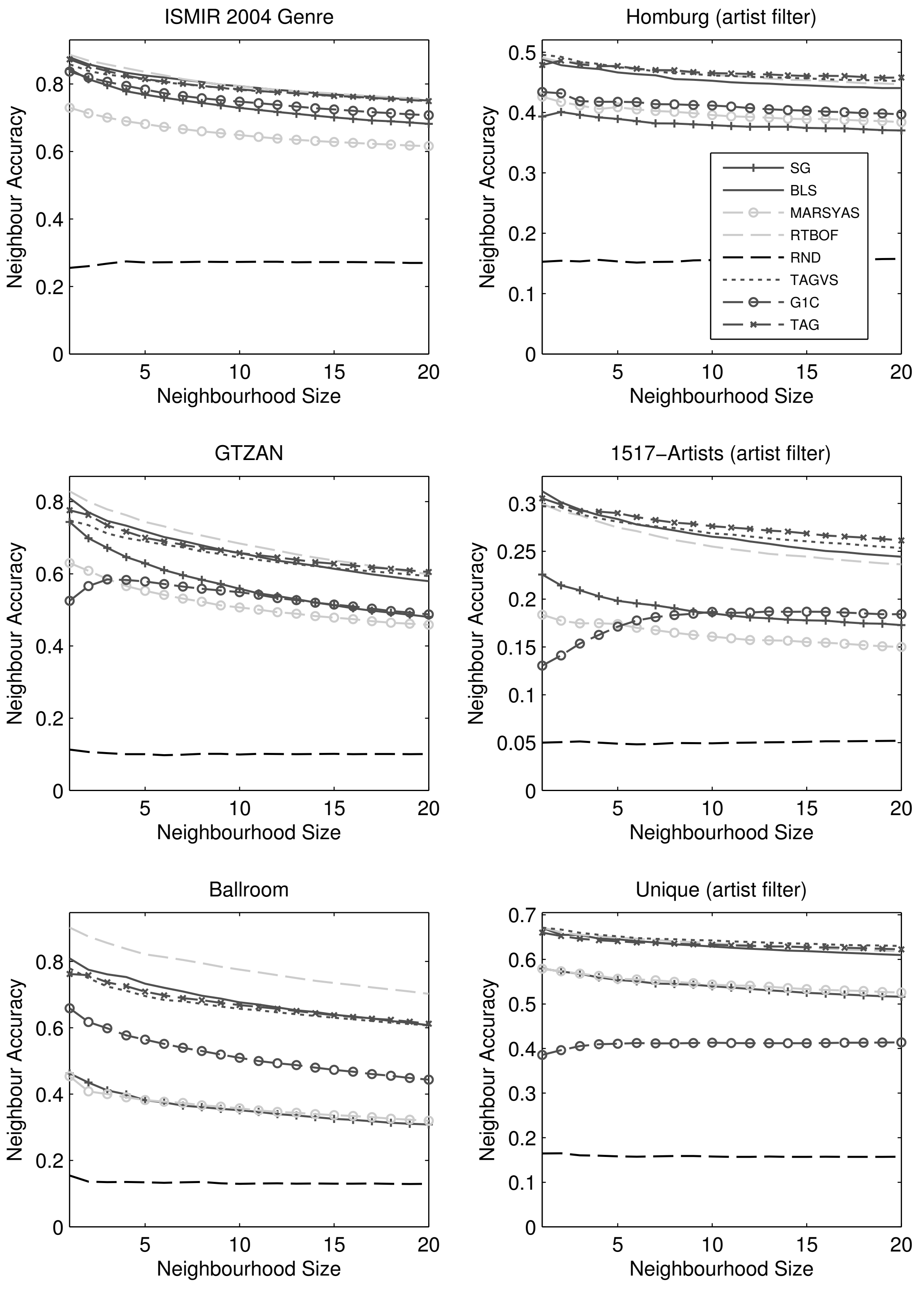


Figure 5.10: Comparison of music similarity algorithms (k-NA).

Furthermore, to quantify any existing portfolio effects all algorithms have been also evaluated using the even more restrictive portfolio filter (see 3.3.2). These results are visualized in figure **??**. As artist information is not available for all datasets, portfolio filtered results are only reported for the *"Homburg"*, the *"1517-Artist"* and the *"Annotated"* datasets. It is also worth mentioning that the results reported for the *"Unique"* dataset are automatically portfolio-filtered, as there is only one track per artist in this dataset. The same argument applies to the *"Annotated"* dataset. In figure **??** the plots to the left hand side report the k-NN CA and the plots to the right hand side report the k-NA quality measure. At a first glance it seems that there is no big change in the evaluation results on datasets *"1517-Artists"* and *"Homburg"*. However, this is not a big surprise as the average number of tracks per artist is very low on both datasets. On the average there are only 2.1 tracks per artist in the *"1517-Artists"* dataset and 1.29 tracks per artist in the *"Homburg"* dataset. Hence, the results on these two datasets are almost portfolio filtered if an artist filter is used. Still, by looking more closely at the results one can observe a difference to the artist filtered results. For both quality measures one can observe a general decrease in classification accuracy. For the k-NN CA this decrease is rather marginal in the range of 0.6 to 0.8 percentage points for dataset *"1517-Artists"* and about 0.4 percentage points for dataset *"Homburg"*. For the k-NA this difference is bigger and in the range of 1 to 1.4 percentage points on dataset *"1517-Artists"* and about 0.2 percentage points for dataset *"Homburg"* respectively. Also the relative difference between the algorithms has slightly changed. Altogether, one can conclude that there is a perceivable impact of the portfolio filter on the obtained results and one can expect that this impact would increase if the average number of tracks per artist were higher.

Last but not least, what jumps to the eye is that on the rather tiny *"Annotated"* dataset the auto-tag based algorithms clearly outperform all other algorithms. This could be an indicator that for smaller datasets auto-tag based similarity is more adequate. One explanation is that no perfectly similar songs exist in small datasets, which would be more easily identified by algorithms that perform a direct feature matching, while auto-tag based algorithms can also infer broader similarities among songs.

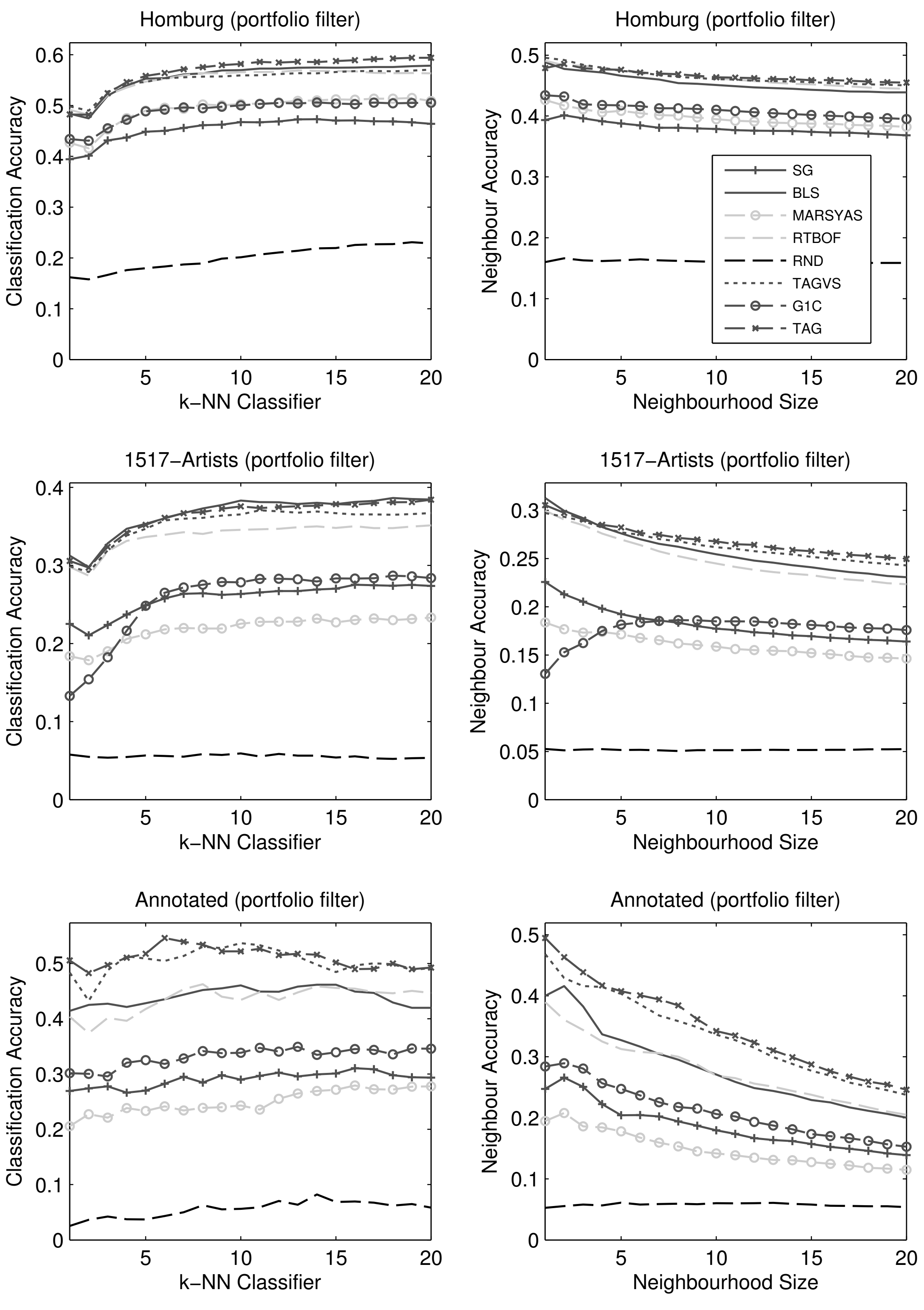


Figure 5.11: Comparison of music similarity algorithms (portfolio filter).

Altogether, the conducted evaluation shows that the proposed block-level algorithms (BLS, TAG, TAGVS) perform at least comparable to state-of-the-art algorithms with respect to their recommendation quality. In the next subsection all possible combinations of the three best performing algorithms (BLS, TAG, RTBOF) are evaluated.

5.5.3 Combinations of Algorithms

In this subsection any further potential for improvement of content-based music recommendation algorithms is identified. This is realized by evaluating all possible combinations of the best performing algorithms of the conducted evaluation. To combine two or more algorithms the distance matrices produced by the individual algorithms are combined using the extended distance space normalization (EDSN) approach presented in section 5.3.1. Figure **??** visualizes the result of the original algorithms and the four evaluated combinations. Note that compared to the previous plots the colors of the algorithms have changed. Darker colors indicate combinations of algorithms, while lighter colors mark the original algorithms. One can observe that in general combinations of algorithms outperform individual algorithms. Over all datasets the combination of all three algorithms (CMB4) seems to achieve the best and most stable classification performance. However, the difference to CMB2 and CMB3 is not huge. CMB1 yields the worst results of all the evaluated combinations. Altogether, we can conclude that there is still some potential for improvement and that the evaluated combinations are all an improvement over the current state-of-the-art. In the next subsection the results obtained by the submissions to the MIREX 2010 evaluation, which are all based on block-level features, are presented and discussed.

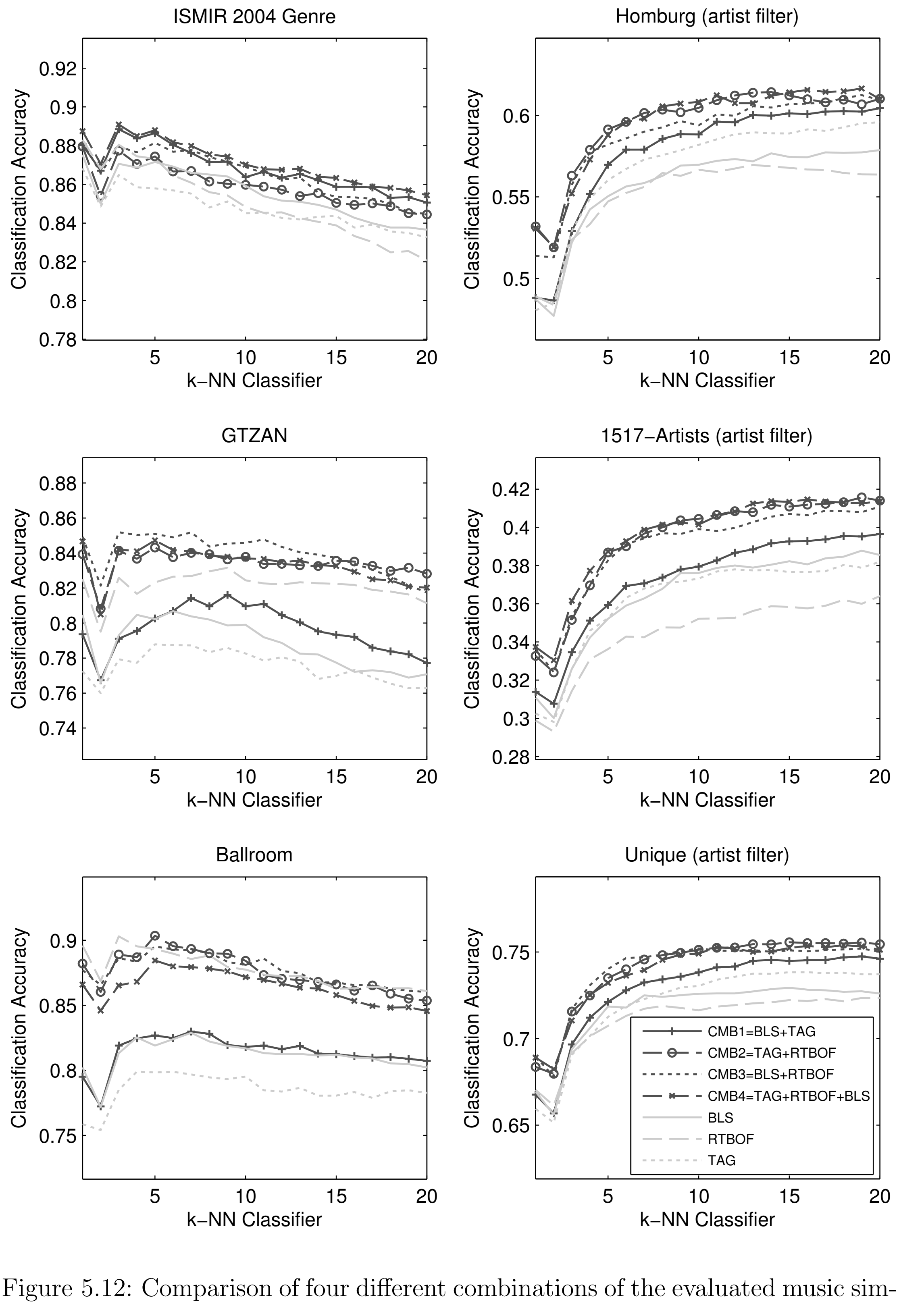


Figure 5.12: Comparison of four different combinations of the evaluated music similarity algorithms.

5.5.4 MIREX 2010 Results

The last subsection of the evaluation section in this chapter focuses on the results of the 2010 Music Information Retrieval Evaluation eXchange (MIREX), a yearly held competitive evaluation of MIR algorithms. The results of the MIREX 2010 evaluation are complementing the in-house evaluation of the proposed block-level features and similarity algorithms, as they represent an external and objective evaluation of the proposed feature set and techniques.

Three algorithms based on block-level features were submitted. The corresponding MIREX categories were the *Audio Music Similarity and Retrieval Task*, the *Audio Train/Test Task* and the *Audio Tag Classification*. The following subsections will report and discuss the obtained results in the different categories. For a detailed description of the submitted block-level feature based algorithms the interested reader is referred to the accompanying algorithm description of the submission [Seyerlehner and Schedl, 2010].

5.5.4.1 Audio Music Similarity and Retrieval Task

The goal of the *Audio Music Similarity and Retrieval Task* is to automatically identify, for a number of query songs, similar songs within a large music collection containing 7000 songs. Then the most similar songs according to the participating algorithms are rated by human evaluators on two different scales. The *fine score* is a continuous value in the range from 0 (failure) to 100 (perfection). For the *broad score* the graders can choose between three categories *Not Similar*(NS=0), *Somewhat Similar*(SS=1) and *Very Similar* (VS=2). These scores are averaged over all queries and over all graders. Figure **??** visualizes the obtained scores of the participating algorithms. The abbreviation SSPK2 denotes the submitted algorithm and is highlighted in the figure. The submitted algorithm (SSPK2) is basically the same as the combination of the auto-tag based algorithm (TAG) and the direct block-level similarity algorithm (BLS). In the previously conducted evaluation this combination was named CMB1. For both broad and fine scores the submitted algorithms obtained the highest scores. This result supports the findings of the evaluation from the previous section and indicates that the proposed algorithms in this thesis are state-of-the-art methods. It is however also important to mention that the difference between the first four algorithms of this human evaluation was not found to be statistically significant.

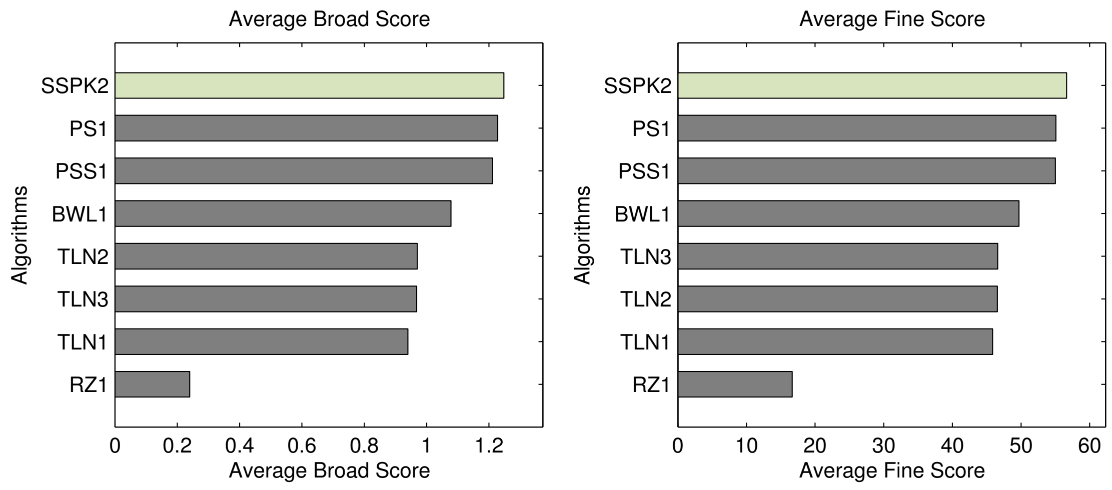


Figure 5.13: Visualization of the results of the MIREX 2010 Audio Music Similarity and Retrieval Task.

Another important aspect of this human evaluation is that the results are pretty much in line with the results of the conducted automatic evaluation. Thus, one can conclude that the results from automatic evaluations are quite reliable quality predictors (see also chapter 3.2.2).

5.5.4.2 Audio Train/Test Task

Automatic music genre classification is a performance task to evaluate the power of feature sets and classification approaches. The precise MIREX task name is *Audio Train/Test Tasks.* There exist four different subtasks that correspond to four different datasets:

- Classical Composer Identification

- Latin Genre Classification
- Music Mood Classification
- Mixed Popular Genre Classification

Figure **??** visualizes the obtained classification accuracies of all submitted algorithms. The submitted algorithm by the author is denoted *SSPK1* and is highlighted in the visualization. The presented block-level feature in combination with the support vector machine implementation of the machine learning toolbox WEKA were used in this submission. In figure **??** on can see that the number of participating algorithms varies depending on the dataset. This is because some of the algorithms failed to produce results on individual datasets. Comparing the block-level feature based algorithm to the other submitted algorithms one can see that the proposed algorithm either ranks first or second on all datasets. This is especially remarkable, as many other algorithms perform well only on individual datasets, but not on all datasets. Altogether, these results clearly show that the proposed block-level feature set is a highly descriptive feature set that is especially suitable for audio classification because of its vector space representation.

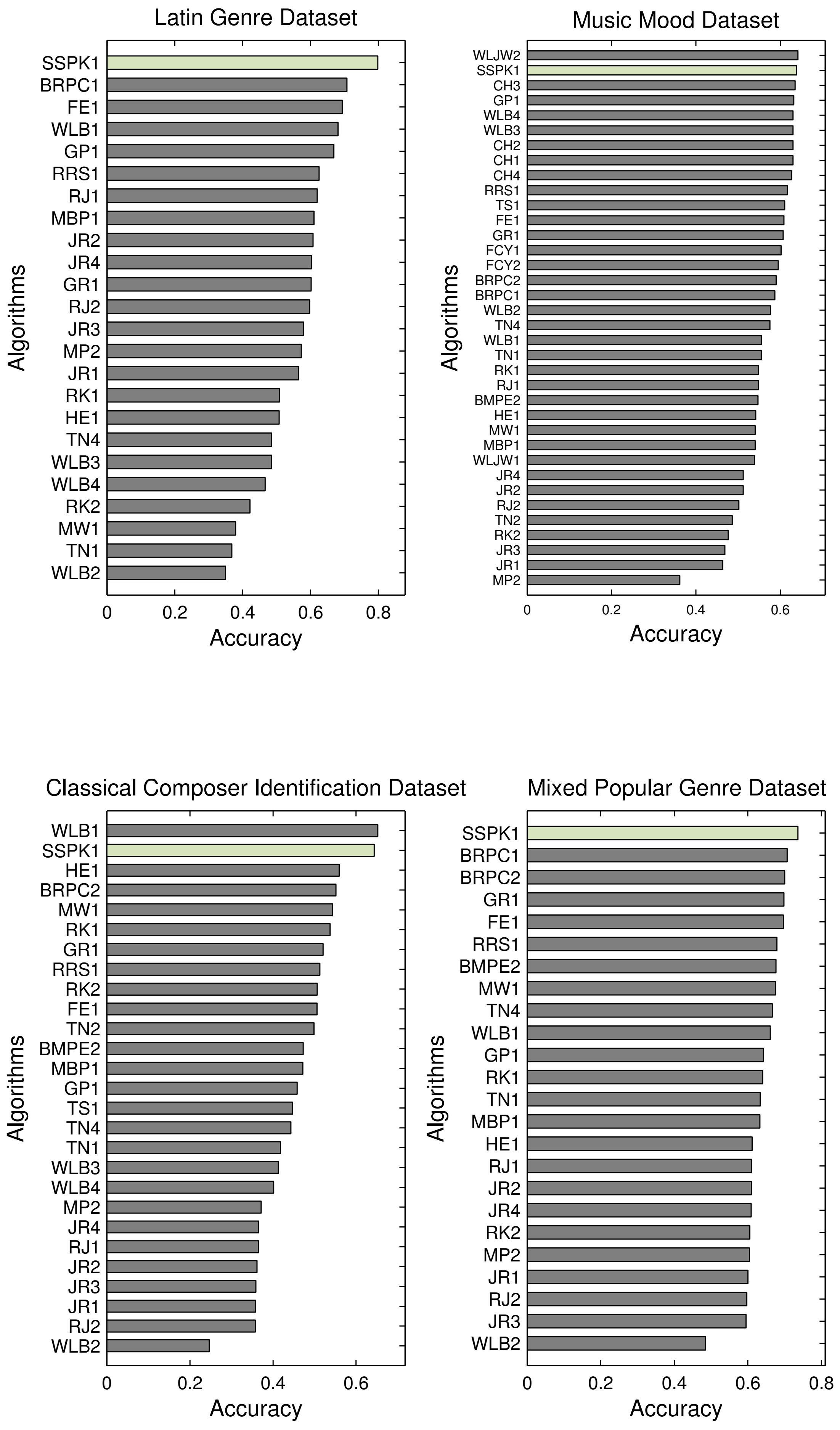


Figure 5.14: Visualization of the results of the MIREX 2010 Genre Classification

5.5.4.3 Audio Tag Classification

To evaluate the block-level feature based auto-tagging approach a variant of the presented tag classification algorithm in section **??** was submitted to the MIREX Audio Tag Classification task. In contrast to the evaluated tag classification algorithm, where a two stage classification approach called *stacked generalization* was used, in the submission a simple support vector machine with a polynomial kernel serves as tag classifier. One reason for this change in the classification procedure was that the submission should be platform independent and therefore the binding to the MARSYAS framework had to be removed. Regarding the evaluation procedure the task for the algorithms is to produce two outputs: for each song and each learned tag classifier an algorithm should report the probability that the tag applies to the given song and also the binary classification results, i.e., if the tag is set or not for the given song. To train the tag classifiers the evaluation datasets are first split into a training and testing set. Two different tag datasets are used in the evaluation: the *MajorMiner Tag Dataset* and the *Mood Tag Dataset*. In figure **??** the results of this MIREX task are visualized for both datasets. For the binary classification results the *f-Score* is plotted as global evaluation metric, while for the predicted tag affinities the *Area Under the Receiver Operating Characteristic Curve* (AUC-ROC) is plotted. Depending on the chosen metric on the MajorMiner dataset the block-level feature based submission (SSPK3) took rank 4 or rank 8. For the Mood dataset the SSPK3 algorithm achieved rank 6 or rank 9. Still, the proposed algorithm belongs to a larger group of algorithms, where the absolute difference in quality is marginal. Consequently, this external evaluation is in line with the obtained results of the conducted evaluation and we can conclude that block-level features are also well-suited for automatic tag prediction.

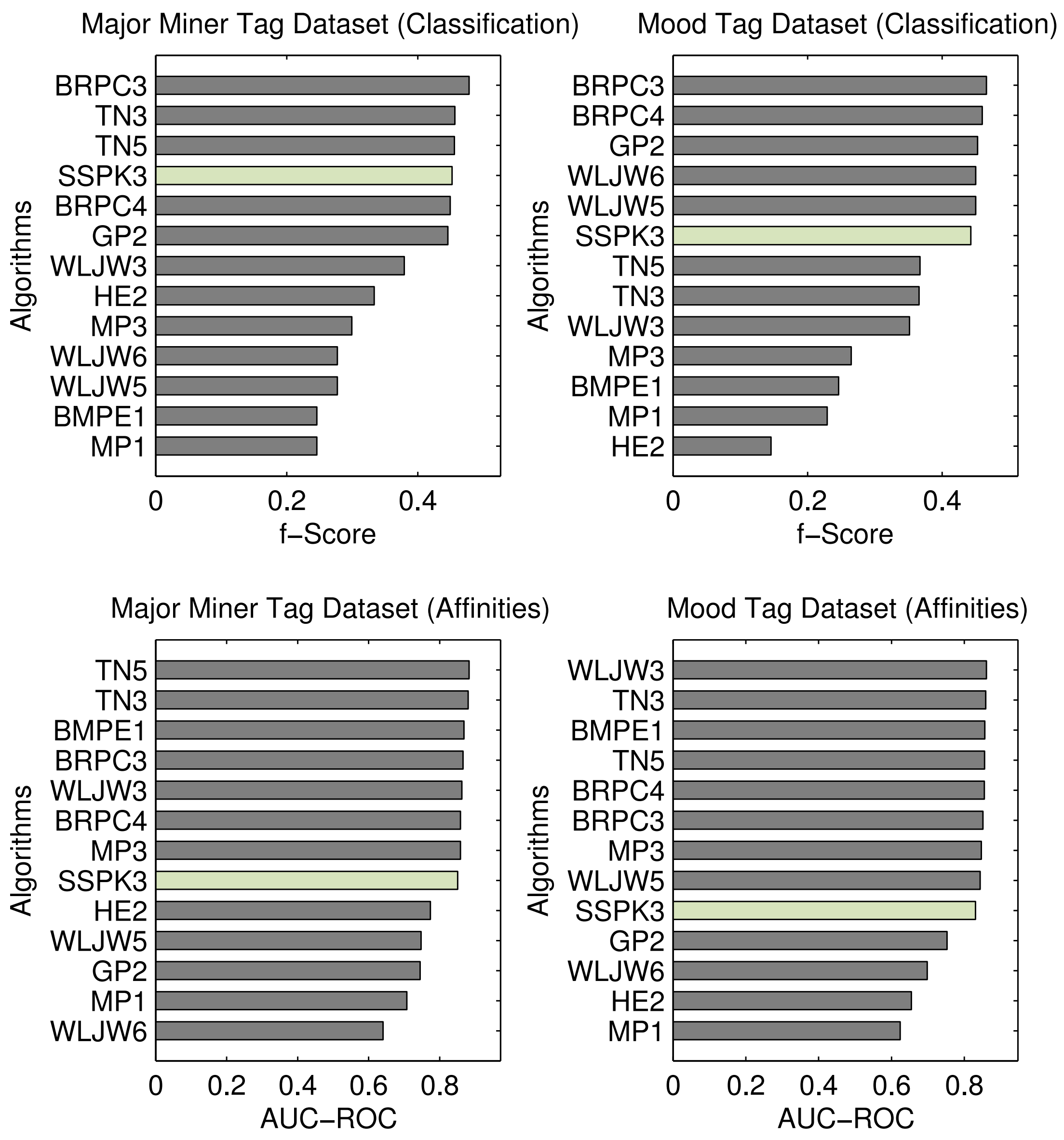


Figure 5.15: Visualization of the results of the MIREX 2010 Audio Tag Classification Results.

5.6 Conclusions

In this chapter a set of block-level features has been introduced that in contrast to frame-level features is especially designed to capture some local temporal information within the features themselves. The proposed feature set is significantly different from other feature sets and all the individual features have a simple vector space representation. This is an essential advantage not only because song models can easily be visualized, but also because this allows to make use of the whole mathematical toolbox that is available for vector spaces.

It could be experimentally shown that these block-level features are also a powerful feature set with respect to the task of automatic genre classification and automatic tag classification. To make them also applicable to music recommendation three music similarity measures based on this block-level features have been defined (BLS, TAG, TAGVS). The first approach (BLS) was to directly define a music similarity measure based on the block-level features using a complex combination approach. The second approach is a semantic space approach, where each song is represented by a number of automatically estimated tags. For this approach two variants have been proposed (TAG, TAGVS).

The bock-level feature based music similarity algorithms have been evaluated with respect to their ability to identify non-trivial music recommendations — i.e. songs not from the same artist as the query songs. This evaluation reveals that all three proposed algorithms perform at least comparably to the state-of-the-art. However, especially the auto-tag based variants seem to be an adequate choice to generate music recommendation. These algorithms do not only allow to explain generated music recommendations and have a semantic interpretable model, but additionally are easy to improve and extend. To improve an auto-tag based algorithm one can either add new features to the training process, or one can learn additional semantic attributes from other tag or genre datasets. Also filtering and weighting of the learned semantic concepts before estimating pairwise song similarities should be considered in future. Another advantage of the auto-tag based approach is that the TAGVS approach can be used in combination with more powerful search strategies.

This way such an auto-tag based recommender system is suitable even for music catalogs containing millions of audio tracks.

Besides the evaluation of individual algorithms also several combinations of the best performing individual algorithms have been analyzed to identify further potential for improvements. The finding is that these combinations typically outperform the individual algorithms. Future research will focus on identifying exactly which components of the combined algorithms are responsible for the achieved improvement.

Another finding of the conducted evaluation concerns the evaluation itself. It could be shown that the evaluation results depend on the chosen dataset and also on the chosen quality indicators. The obtained results based on automatic classification can be inconsistent or contradicting. To provide a comprehensive overview of the recommendation quality of an algorithm it should be evaluated on a number of different datasets. Furthermore, different quality indicators should be used.

In contrast to this chapter, which basically focused on the underlying music similarity algorithms of content-based music recommender systems, the following chapter considers improvements on a more abstract level, where music recommender systems are interpreted as recommendation networks.

6 Analysis of Music Recommendation Networks

In contrast to the previous chapters, which basically tried to improve music recommender systems by improving the underlying music similarity measures, the underlying features and the evaluation methods thereof, in this chapter analysis and improvements on a more abstract level are considered. On this higher level of abstraction a music recommender system is represented by a directed graph, where the vertices of this graph correspond to songs and the edges in the graph represent the recommendations for each song. Based on this graph representation network properties like *reachability*, *connectedness* and *hubness* of the emerging complex network can be analyzed. These network properties are essential as music recommendation services are typically used to explore the online music catalog by moving from recommendation to recommendation. Thus, music recommendation is not a one step process, but a continuous process. Therefore it is not enough to focus on accuracy based quality criteria only when music recommender systems are analyzed, but it is also important to analyze if users will be able explore a music catalog via browsing and navigating within the emerging recommendation network, which is the focus of this chapter. The following section presents an overview on related work with respect to the analysis of music recommendation networks and introduces the related *Long Tail* phenomenon. Then the subsequent sections will focus on theoretical and empirical analysis of the *reachability* of songs in music recommendation networks and methods that reduce the identified navigation issues.

6.1 Related Work and the Long Tail

The analysis of emerging music recommendation networks is a relatively new research area. The first who analyzed music network structures were Aucouturier and Pachet [Aucouturier and Pachet, 2008, Aucouturier and Pachet, 2004]. Their research was driven by the observation that certain songs occur in a large number of playlists, although they do not share any perceptually meaningful similarity to the query songs, if these playlists were generated by frame-level music similarity algorithms (see Chapter 4). They named these songs *hub*-songs or simply *hubs* (see 6.2.3). Aucouturier and Pachet conducted many interesting experiments to better understand the emergence of hubs. They tried to understand why certain similarity measures and feature sets produce more hubs than others, but they did not explicitly conduct any analysis of music recommendation networks, but rather interpreted hubs as an issue related to the similarity algorithm. Celma [Celma and Cano, 2008, Celma and Herrera, 2008], in contrast, explicitly focused on the analysis of music recommendation networks. He compared the structure of music recommendation networks generated by a content-based, a collaborative-filtering and an expert-based music recommender system. He investigated different network properties related to navigation, connectivity and clustering. One of the main findings of Celma is that collaborative filtering recommenders typically follow a popularity rule, so that they recommend the most popular songs to all users, whereas unpopular or new songs in a music catalog that have just a few or no ratings are not or very rarely recommended. This behavior of collaborative filtering recommenders leads to a very typical song-popularity distribution within a music catalog (e.g. measured in play-counts or song-downloads). The resulting popularity distributions are in general characterized by a head containing the few popular songs and a long tail consisting of all unknown or unpopular items. This specific pattern has motivated the phrase *Long Tail* [Brynjolfsson et al., 2003, Anderson, 2006, Brynjolfsson et al., 2006]. Of course exploring a music catalog does not only mean browsing the few popular items, but more importantly means browsing and finding interesting niche products within a catalog, which unfortunately reside in the Long Tail. Interestingly, Celma could show that expert-based and content-based systems are not affected by such a popularity bias. Using these types of system songs sitting in the Long Tail would be accessible, which is a big motivation for building hybrid recommender systems.

Here in this chapter the network analysis of Celma is extended to especially analyze the navigability of content-based top-N music recommender systems. The special focus of this chapter will be on measuring the *catalog coverage* of music recommender systems [Herlocker et al., 2004]. The *catalog coverage* measures what percentage of items a recommender system ever recommends to users and is related to the *coverage*, which measures the percentage of a dataset that the recommender system is able to provide recommendations for. While traditionally catalog coverage was measured based on a set of recommendations formed at a single point in time — e.g. it was measured by the percentage of all items that were recommended to all users in the population over a specified time period — , in this chapter it is proposed to measure the catalog coverage by identifying the number of not reachable vertices in a recommendation network. These not reachable songs in the recommendation network are called *anti-hubs* [Godfrey and Chordia, 2008], *orphans* [Gasser et al., 2010] or *sources* [Seyerlehner et al., 2009b]. By identifying the number of *sources* in a recommendation network one obtains an inverse measure of the catalog coverage.

6.2 Complex Network Analysis

The analysis and experiments of the emerging music network structures conducted in this chapter do not only apply to music recommendation networks, but generalize to any arbitrary recommendation network that can be derived for any *top-N recommender system* [Seyerlehner et al., 2009a]. To measure the catalog coverage of an item-based top-N recommender system the recommender system is first transformed into an equivalent *recommendation network* or *recommendation graph*. Given a recommender system presenting a user a list of precisely l recommendations for each item in the database, the recommendation graph of this system can be represented as a directed graph $G = (V, E)$, where each vertex $v \in V$ corresponds to an item in the database. Furthermore, each vertex v has exactly

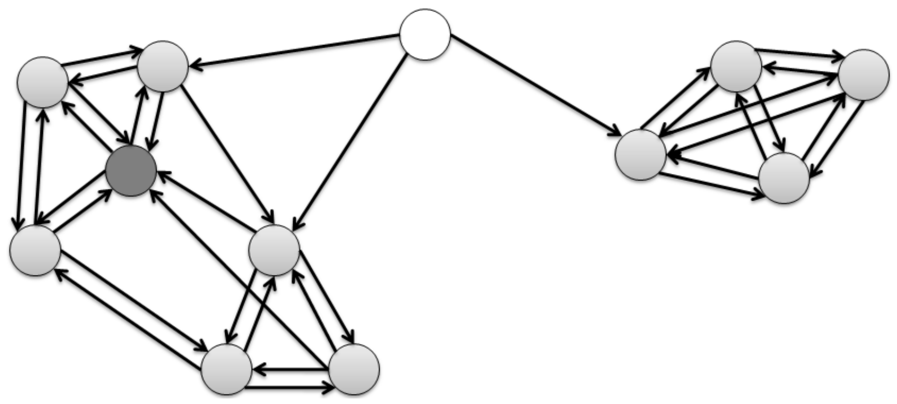


Figure 6.1: The recommendation graph of a top-N item-based recommender. Each vertex has exactly l outgoing edges. In this example $l = 3$. The number of incoming edges is an emergent property of the network. There might be hubs, i.e., vertices with a high indegree (dark), and sources, i.e., vertices with an indegree of zero (white).

l outgoing edges pointing to those items in the database that are recommended for item v. Thus, each vertex v has an outdegree of l ($\deg^+(v) = l$). Figure 6.1 visualizes such a directed recommendation graph.

The graph representation makes it easy to identify items that are not reachable via recommendation. Obviously, every vertex v with an indegree of zero ($\deg^-(v) = 0$) corresponds to such an item. In graph theory a vertex with an indegree of 0 is called a *source*. Vertices with a very high indegree are called *hubs* (see figure 6.1). The number of sources is an emergent property depending on the network structure. It is an inverse measure of the catalog coverage, as it measures the number of items that are not reachable, while catalog coverage informs about the number of items that are ever recommended. In this chapter it will be shown that sources naturally exist in any recommendation network and a lower bound on the number of sources one can expect will be established.

6.2.1 Reachability: A theoretical view

The basic principle of an item-based recommender system is to first estimate similarities among catalog items and then generate recommendations based on these similarities. Consequently, the estimates of the item-to-item similarities determine the respective recommendation graph as they define the edges between the vertices. The structure of the graph in turn determines the number of sources. Hence, a similarity function that is completely unbiased (with no preference of linking specific items) will produce the fewest sources in a recommendation graph. Clearly a random recommender is a realization of such a *"fair"* similarity function and represents a lower bound with respect to the number of sources. Therefore to identify the lower bound the recommendation graph of a random recommender is analyzed.

Random graphs [Newman, 2003, Erdős and Rényi, 1959] have been intensively studied in complex network analysis. The specific random graph model that is studied here is a directed graph $G = (V, E)$ with a fixed outdegree for all vertices, as defined previously. Furthermore, it is assumed that the recommended items are chosen randomly. Thus, each item in the recommendation graph will point to l other random items, but not to itself. If we suppose that there are N items in total in the database, then the recommendation graph will have N vertices and lN random edges. The probability of an item in this graph to be referenced by exactly k other items is then given by the binomial distribution:

$$P(X = k) = \binom{l(N-1)}{k} p^k (1-p)^{l(N-1)-k}, \tag{6.1}$$

where $p = \frac{1}{N-1}$ as each item will equally likely point to any other item, but not to itself. The probability of not being referenced by any other item in the graph is

$$P(X = 0) = (1 - \frac{1}{N-1})^{l(N-1)}. \tag{6.2}$$

Knowing the probability for a single item not to be referenced by any other item, we can now estimate the *expected number of sources* $E(X)$ (items that are not referenced) for the whole system containing N items.

$$E(X) = NP(X = 0) = N(1 - \frac{1}{N-1})^{l(N-1)} \tag{6.3}$$

To validate the proposed model, the number of sources of a real implementation of a random recommender was studied. In figure 6.2 the number of observed sources is plotted together with the expected number of sources from the model. The model perfectly fits the observed numbers. It essentially depends on the two parameters l and N. The number of sources seem to linearly increase with the collection size (see figure 6.2). Depending on l, the number of sources grows more or less slowly. If N approaches infinity ($N \to \infty$) the expected number of sources ($E(X) \to \infty$) approaches infinity as well. Deriving the limit, for N approaching infinity, of the probability for an item to be a source-item (equation 6.2), it can be seen that the probability converges:

$$\lim_{N\to\infty} P(X = 0) = \lim_{N\to\infty} (1 - \frac{1}{N-1})^{l(N-1)} = e^{-l} \tag{6.4}$$

In combination with equation 6.3 this implies that the percentage of sources will stay constant and that the number of sources increases linearly with the database size. More importantly, the percentage of sources exponentially decays with the length of the recommendation lists. Thus, it is in general desirable to have long recommendation lists, as this implies high catalog coverage. In practice, however, the number of recommended items is constrained by usability concerns and is typically rather low (usually below ten). Furthermore, the random recommender model is optimal in the sense that it will create the fewest sources that one can expect. For real world recommender systems the percentage of items inaccessible via recommendation can be much higher (see next section). One major theoretical result is that even in the best case, and independently of the recommendation approach, there will always be a percentage of items that will be inaccessible when browsing a top-N recommender system — unless it is specifically designed (see section 6.3).

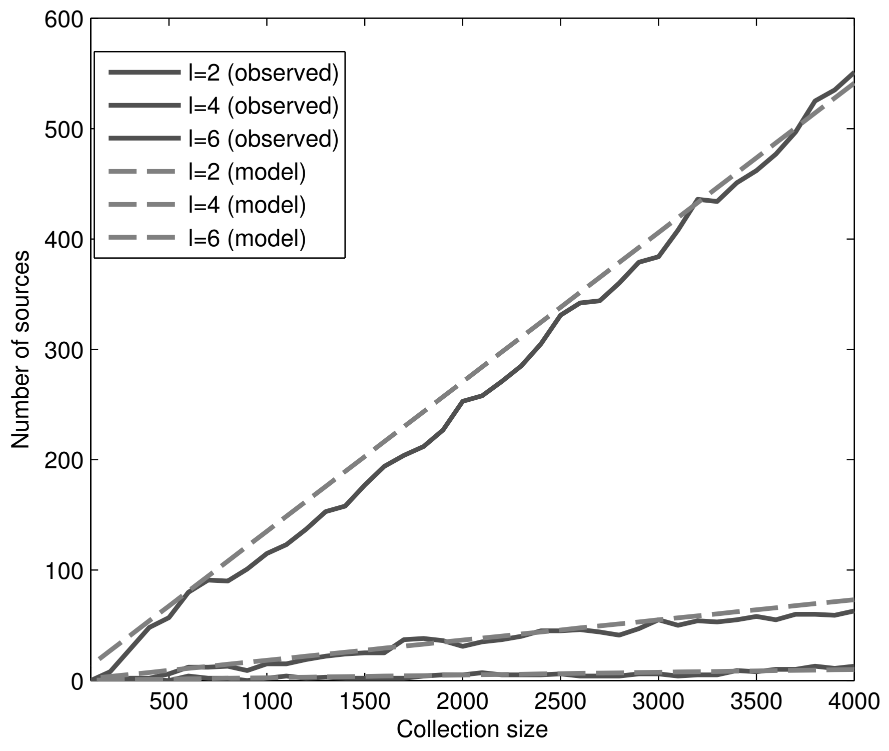


Figure 6.2: Number of sources observed for a random recommender and the predictions from the proposed model.

6.2.2 Reachability: An Empirical Study

Given the theoretical findings on the lower bound of the number of expected sources, it will now empirically be shown that for real world recommender systems, this lower bound is far too optimistic. Two different types of recommender system (see section 1.3) from different domains have been analyzed to demonstrate that the reachability issue is a fundamental problem of any recommender system: a real world *content-based* music recommender system and a prototypical *collaborative-filtering* based movie recommender system. To evaluate these two systems, the number of sources for various recommendation graphs has been determined: Systematically recommendation graphs for different recommendation list length l were generated. A recommendation list length of $l = 5$ implies that the five most similar items are recommended. Hence, the corresponding system is a top-5 recommender. Furthermore different collection sizes were simulated by randomly removing items from the respective datasets.

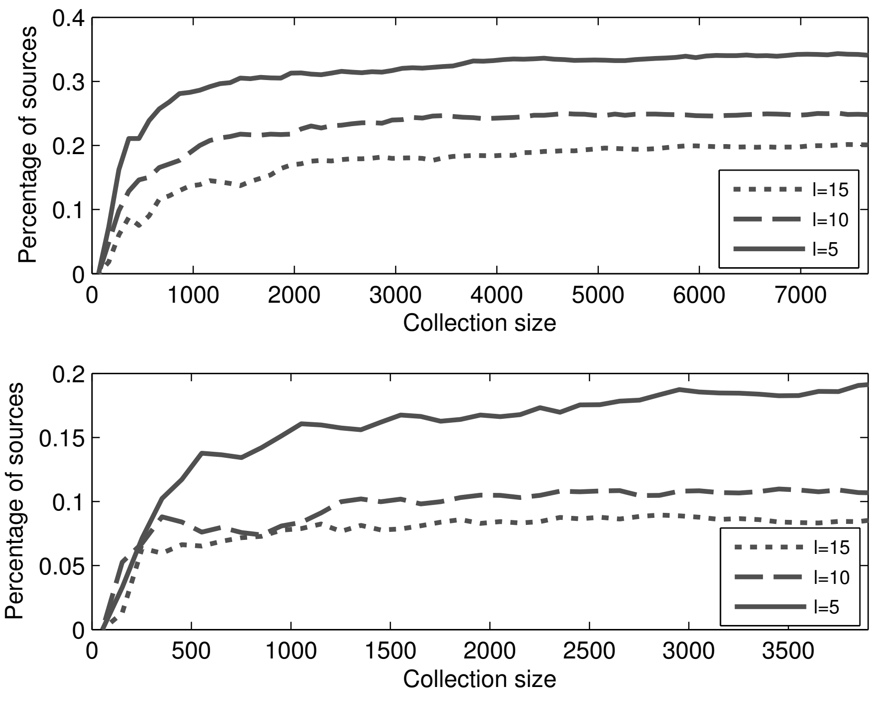


Figure 6.3: Analysis of a content-based music recommender system (upper plot) and a collaborative filtering based movie recommender (lower plot). The number of sources is illustrated for a top-5, a top-10 and a top-15 recommender system. The lower limit for the expected percentage of sources is 0.00673 for top-5, 0.000453 for top-10 and $2.06115 * 10^{-9}$ for top-15.

Content-Based Recommendation As an example of a content-based recommender system we analyze the real world music recommender attached to the *FM4 Soundpark*[1], an Austrian music portal. At the moment this platform contains about 10000 songs and is steadily growing. At the time of our experiments, 7665 songs were available. The recommender system of the FM4 Soundpark is

[1]`http://fm4.orf.at/soundpark/main`

based on a standard similarity measure for music audio files [Flexer et al., 2008], the Single Gaussian (SG) algorithm (see 4.2.2). Based on this similarity measure one can generate song recommendations of arbitrary length, ordered according to the similarity to the query song. Figure 6.3 (upper plot) shows that for recommendation lists of length $l = 5$ approximately 34% of all songs are sources and are therefore inaccessible via browsing.

Collaborative Filtering The analyzed collaborative filtering recommender is based on the well-known Movielens dataset[2], which contains 1 million ratings for 3900 movies by 6040 users. The simulated recommendation approach is similar to [Sarwar et al., 2001]. The cosine distance between item vectors of the user-item ratings matrix is used as a similarity measure. Based on these similarities the experimental recommender then simply proposes the top-N most similar items according to the cosine distance. Even if this is surely not a high quality recommender system, it will be useful as here we are not striving for improved classification accuracy. The sole goal is to demonstrate that even for collaborative filtering approaches, recommender systems based on real world datasets will produce many sources, far above the lower bound predicted by our model. The results of the analysis are visualized in figure 6.3 (lower plot). Obviously, in practice, the number of produced sources is far above the theoretical lower limit, which is significantly below one percent for all systems (see caption figure 6.3). Thus, a significant portion of the movies is never recommended by this experimental recommender system. For example, 19.23% of all movies will never be recommended when browsing the top-5 recommendations only. Consequently, for real world recommenders, sources can significantly reduce the utility of a recommender system. However, sources are not the only problem with respect to the browsabilty of music recommendation networks. Also hub-songs are problematic in this context.

6.2.3 *Hubs*: Highly Reachable Songs

In a music recommendation network *hubs*, in contrast to sources, are vertices with a very high indegree i.e., items that are in the recommendation lists of a large

[2]www.grouplens.org

number of other items (see figure 6.1). While hub-songs were initially seen as *bad* or *incorrect* recommendations, hubs also significantly decrease the browsability of music recommender systems. In this context especially large hubs are problematic as the users of the recommender system will likely revisit hub-songs many times when browsing the recommendations, which can be extremely annoying.

However, it is not as straightforward as for sources to define which vertices in a graph are actually hub vertices. The first definition of *hubs* was coined by Aucouturier [Aucouturier and Pachet, 2008, Aucouturier and Pachet, 2004]. He defined a song to be a so-called *hub song* if

- the song appears in most of the recommendation lists of the other songs in a music catalog

  and

- most of these appearances do not correspond to any meaningful perceptual similarity.

To quantify the *"hubness"* of a song Aucouturier introduced the notion of n-Occurrence[3] of a song, which is the same as the indegree $\deg^-(v)$ of a vertex $v \in V$ of the recommendation graph. However, Aucoututier's definition of hubs is too imprecise. In this thesis we follow the definition of Gasser et al. [Gasser et al., 2010] and define that hubs are outlier vertices where the indegree is higher than three times the *expected number of incoming edges*:

$$\text{isHub}(v) = \begin{cases} 1 & \deg^-(v) > 3l \\ 0 & \text{otherwise} \end{cases} \tag{6.5}$$

The expected number of incoming edges can be derived from equation **??** by computing the expected value of the binomial distribution $E(X) = np = l(N-1)\frac{1}{N-1} = l$.

[3]The n-Occurrence is the number of times a song occurs in the first n nearest neighbors of all the other songs in the dataset.

Figure 6.5 visualizes the indegree of each song and the indegree distribution for the Single Gaussian (SG) algorithms on dataset *"1517-Artists"* for recommendation list of length $l = 15$. The vertices with an indegree higher than the dotted line are the hub vertices according to the above definition. As already observed by Aucouturier the indegree distribution resembles an exponential distribution, which led him to the conclusion that the recommendation networks of frame-level algorithms belong to the category of so called *scale-free networks*. However, taking a look at the indegree distribution of the proposed MLVQ algorithm (see 4.3.1.1) also visible in figure 6.5, one can conclude that this assumption is not true for all frame-level similarity algorithms. Especially non-parametric modeling of local spectral features seems to reduce the number of hubs, which was simultaneously found by [Hoffman et al., 2008] and [Seyerlehner et al., 2008]. Furthermore, using the Nearest Neighbour (NN) density estimation algorithm presented in section 4.3.2.1 one can investigate the relation between the smoothness of the distribution of the local features and the hubness of songs. Figure 6.4 shows the number of hubs, the number of sources and the maximum hub. Additionally, the neighbour radius around a MFCC vector that is used in the density estimation and controls the smoothness of the distribution model has been varied between 20 to 60. From these results one can see that, on the one hand, very multi-modal and spiky distributions generated by a low neighbour radius produce many hubs and sources. On the other hand, very smooth distributions generated by a large neighbour radius, also produce more hubs than the optimal configuration (a neighbour radius of 45). Based on this result one can conclude that the smoothness of the distribution used to model the MFCC vectors has a significant influence on the hubness of the resulting recommendation network of frame-level similarity algorithms. This relation of hubness and smoothness of a distribution will also be discussed in the following section focusing on the relation between hubs and sources in more detail.

6.2.4 Relation between Hubs and Sources

To investigate the relation of hubs and sources the graph representations of the top-1 up to the top-100 recommender ($l = 1 \ldots 100$) for the FM4 Soundpark have been

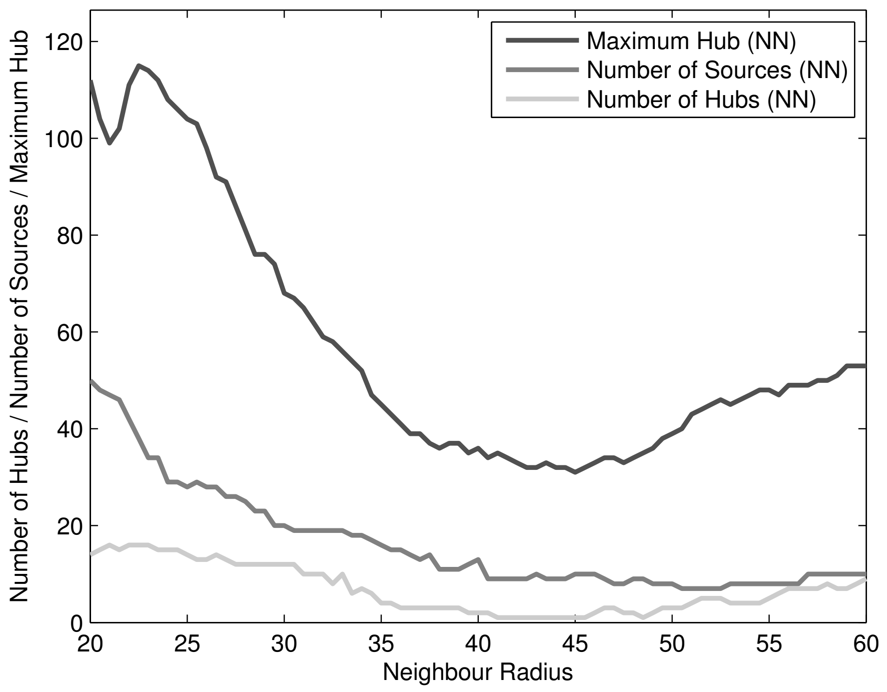


Figure 6.4: Visualization of the number of hubs, the number of sources and the maximum indegree for distribution models of different smoothness. Dataset *"Annotated"* and recommendation list containing 10 songs are used.

created systematically. For each of these graphs the number of sources, the number of hubs and additionally the *"biggest"* hub, the vertex with the maximum indegree, were determined. Figure 6.6 shows that for short recommendation lists the number of sources is extremely high. When increasing the length of the recommendation list, the number of sources decreases exponentially — as expected. Furthermore, the number of hubs exponentially decreases as well. In contrast, the indegree of the maximum hub increases, while the number of hubs stays approximately constant. Therefore, simply increasing the recommendation list length will make more items reachable, but in turn will also create even extremer hubs. This implies that when browsing such a recommendation network, one will often visit or re-visit hub items, which degrades the browseability. Furthermore it seems that hubs and sources are related phenomena. For small recommendation lists extreme hubs imply that

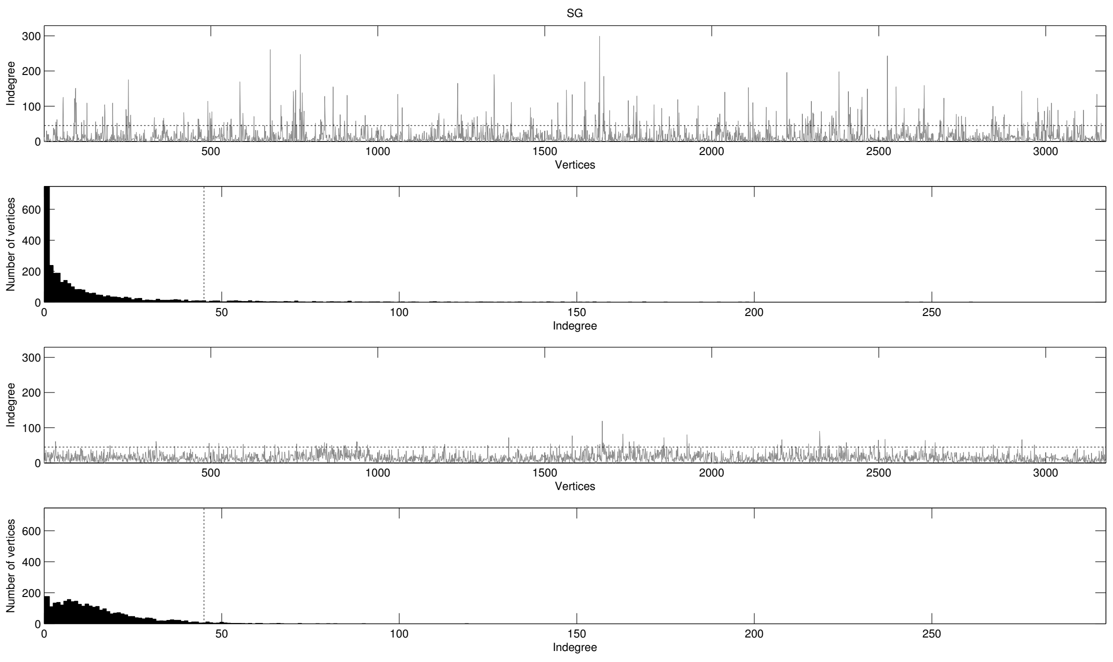


Figure 6.5: Visualization of the indegree of the Single Gaussian (SG) measure (upper plots) and the Multi-Level Vector Quantization (MLVQ) approach (lower plots). Any vertices with an indegree higher than indicated by the dotted line are hub songs by definition.

there are many sources and vice versa (see figure 6.4). A very informal but intuitive explanation is that hub-vertices are some sort of attractors *"stealing"* the links from other vertices, which may even lead to a vertex having no incoming links at all. Recent research on hubs [Radovanovic et al., 2009] indicates that the emergence of hubs is a direct consequence of the high (intrinsic) dimensionality of the underlying data. This finding explains why the smoothness of the distribution model that is used to model the observed MFCC vectors is so crucial, which was observed in the previous section (see figure 6.4). Intuitively, smoother distribution models will implicitly define a music similarity space of lower intrinsic dimensionality, which in turn will create fewer hubs and fewer sources in the emerging music recommendation network. All in all, one can conclude that hubs are a consequence of the curse of dimensionality, and sources are the counterpart of hubs.

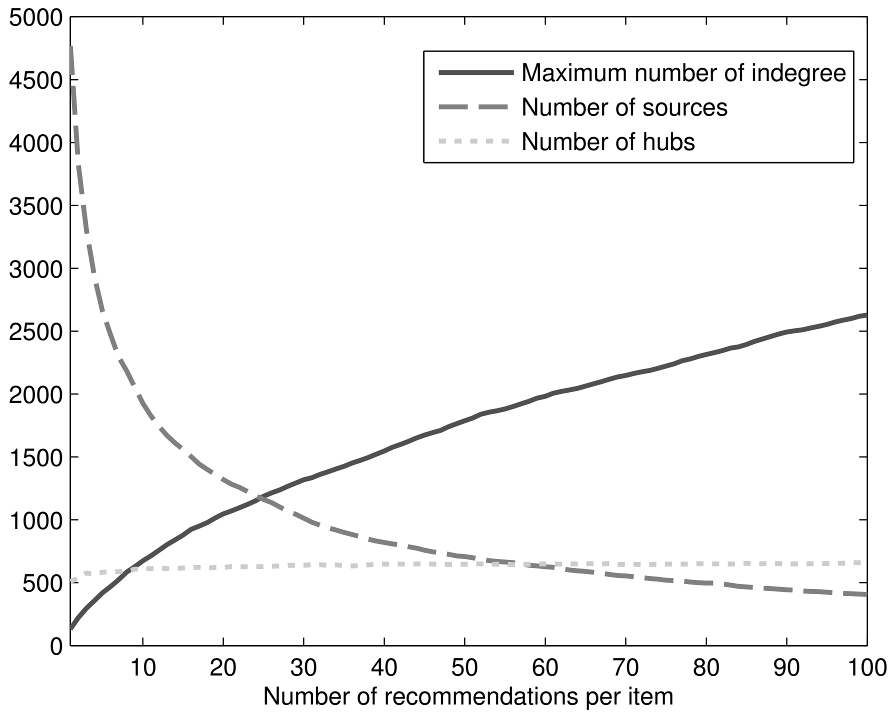


Figure 6.6: Analysis of the FM4 Soundpark. For small recommendation lists, the number of sources is extremely large. It decreases with a growing number of recommendations per song, whereas the maximum indegree over all vertices in each graph increases. For a result set size of 100, there is one song that appears in the recommendation list of 2628 other songs, or in 34.29% of all recommendation lists.

6.3 Improving Music Recommendation Networks: Two Approaches

As shown in the previous section both *hubs* and *sources* significantly reduce the browsability of music recommendation networks. In this section two methods that improve the navigability of recommendation networks are discussed. The first approach is a graph transformation algorithm that transforms any directed recommendation graph into a undirected so-called *browsing graph* that under some conditions ensures reachability of all songs of the underlying music catalog. The second method is based on the *hubness* analysis of the state-of-the-art algorithms presented in section 5.5. The finding is that the distance space normalization (DSN) operation that is used to combine the individual components of the similarity measure significantly reduces the number of hubs and sources in the respective music recommendation network.

6.3.1 A Graph Transformation Algorithm

The basic idea of the graph transformation algorithm presented in this section is to modify a given directed recommendation graph in such a way that it is more adequate to browse and explore a music recommendation network. One main question is: *Under which conditions is a graph well-suited to browse a recommendation network?*

In the following three reasonable graph properties are proposed to define a subset of recommendation graphs that are well-suited for browsing, which will then be called *browsing graphs*.

6.3.1.1 Properties of a Browsing Graph

Up to this point reachability as the only important property of a recommendation graph has been considered. In this section additional desirable properties of recommendation graphs are derived by analyzing user requirements of music recommender systems.

A typical requirement of many real world recommender systems is that the result set should be relatively small — first of all, because the display space for recommendations is in general limited on output devices, and secondly, because too large result sets would confuse the user and make for a very unfocused search. Thus, it is a natural constraint that the size of the result set should not exceed a maximum number of recommendations k_{max}. For the corresponding recommendation graph

$G = (V, E)$ this implies that the outdegree of all vertices is less or equal to $k_{\max}$. This property is called **maximum outdegree property**.

$$\forall v \in V : \deg^{+}(v) \leq k_{\max} \tag{6.6}$$

The second constraint is that if item B is a recommendation for item A then item A should also be a recommendation of item B. This corresponds not only to humans' intuition that similarity relations are symmetric, but also allows to easily go back each recommendation step. The **symmetry property** as defined in (6.7) implies that the browsing graph is an undirected graph.

$$\begin{aligned} &\forall e_1 = (v_1, u_1) \in E : \\ &\exists e_2 = (v_2, u_2) \in E : \\ &\quad v_1 = u_2 \wedge u_1 = v_2 \end{aligned} \tag{6.7}$$

Finally, the notion of reachability is extended. Reachability just ensures that starting from an arbitrary vertex there is at least a single path to each other vertex. This could make it rather difficult to find this path. Therefore, we require each vertex to have a minimum number of incoming edges. For the browsing graph this implies that each vertex has a minimum indegree $k_{\min}$ and means that each item is reachable by recommendations from at least $k_{\min}$ other items. This property is called **minimum indegree property**.

$$\forall v \in V : \deg^{-}(v) \geq k_{\min} \tag{6.8}$$

The result from this requirement analysis is the claim that a recommendation graph is better suited for browsing a music archive if additionally to reachability also these three properties are ensured. Such a graph is then no longer called a *recommendation graph*, but a *browsing graph* instead. In the next section a heuristic algorithm is presented that allows to transform a recommendation graph into such a browsing graph and the effect of this transformation on the recommendation quality will be evaluated.

6.3.1.2 Constructing a Browsing Graph

The main idea of the transformation algorithm is to transform a recommendation graph into a browsing graph, simply by replacing all directed edges by undirected edges and then iteratively (and heuristically) removing edges from the resulting graph so that the *maximum outdegree* and the *minimum indegree* properties are satisfied for all vertices. The *symmetry property* is automatically ensured because the graph is undirected. Furthermore, reachability is guaranteed if the resulting graph is connected.

The proposed algorithm has three important parameters. There is the minimum indegree k_{min} and the maximum outdegree k_{max}, which directly result from the required properties. It is easy to see that in combination with the symmetry property this implies that each vertex in the final browsing graph will have to have an edge degree between k_{min} and k_{max}. The proposed algorithm starts from the directed version of the recommendation graph. One could of course run the algorithm based on a recommendation graph with outdegree k_{max}, but since we want to give our algorithm additional flexibility during the process of removing edges, it is required that the original recommendation graph has an outdegree of at least k_{start} for all vertices. This implies that one can generate for each item at least k_{start} recommendations. The three parameters are related to each other as stated in (6.9).

$$k_{\text{min}} < k_{\text{max}} < k_{\text{start}} \tag{6.9}$$

The only thing left to do is to remove edges till each vertex has a degree in between k_{min} and k_{max}. This should be done in such a way that each vertex tries to remove its *"weakest"* links (i.e., those with the lowest degree of relatedness), since the recommendations should be as good as possible. This can be done as follows:

1. Put all vertices into a priority queue q, where all vertices are sorted according to their degree $\deg(v)$; break ties among same-degree nodes randomly;

2. Pop the vertex with the highest degree from the queue.

3. If this vertex already has a degree smaller than or equal to k_{max}, then all vertices in the queue have a degree smaller or equal to k_{max}. We are done.

4. As the current vertex has too many edges, remove an edge that connects this vertex to another vertex having a degree greater than k_{min}. Choose the edge to remove according to the indegree of the neighboring vertices.

Remove the edge connecting to the vertex with the highest indegree and if there are several vertices of the same indegree remove the vertex with the weakest (lowest similarity) edge. If this vertex is not connected to any other vertex having a degree greater than k_{min}, then we are not able to ensure the maximum indegree property for this node. Stop in this case.

5. Since we have removed an edge, the indegrees of the two vertices connected by the edge have changed. Remove them from the queue and reinsert them so that the queue is up to date.

6. Go back to step 2.

One important remark is that in some cases the proposed algorithm might find a solution where individual vertices have an edge degree higher than k_{max}, violating the maximum outdegree property. This can be due to the fact that for given constraints there simply does not exist any solution. In such a case weakening the constraint till enough solutions to the problem exist can help. If there are enough solutions, simply rerunning the algorithm might help, as vertices of the same edge count are inserted into the priority queue in random order and the algorithm might find different solutions for different runs. However, our experiments indicate that it is quite easy to find a valid solution. Furthermore, the proposed algorithm does not guarantee that the resulting graph is connected, but in all our conducted experiments the resulting browsing graph turned out to be connected.

6.3.1.3 Validation of the Transformation Algorithm

To validate the proposed algorithm the resulting graph after the transformation of the FM4 Soundpark into a browsing graph has been analyzed. The parameters used to transform the graph were $k_{\text{min}} = 4$, $k_{\text{max}} = 7$ and $k_{\text{start}} = 9$. To verify that the recommendation quality is approximately the same before and after the graph transformation, the neighbour accuracy (NA) was computed for both graphs. For all query songs q we count the number of songs in the recommendation list $R(q)$ that have the same genre as the query song and compute the overall percentage relative to the number of recommended songs. That way one measures the accuracy of the recommendations independently of the number of the recommendations. The accuracy of the recommendations using result sets of length $k = 5$ was 35.39%, for $k = 6$ it was 34.86% and for $k = 7$ it was 34.32%. After the transformation using the above parameters the accuracy was 35.63% with an average degree of 5.918 per vertex. This result indicates that there is only a marginal change in recommendation quality. Furthermore, to evaluate how the reachability of songs

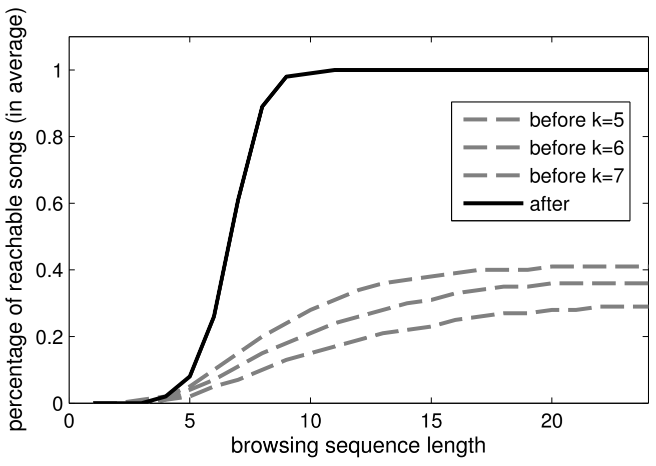


Figure 6.7: The average percentage of songs that can be reached by browsing sequences of different lengths. Before the transformation (for $k = 5, 6, 7$) and after the transformation.

has changed it was investigated how many songs can be reached on average by a recommendation sequence of length l. To do so, for each song the number of songs that can be reached by such a sequence was computed. This can be done by traversing the recommendation graph using the *breadth-first search (BFS)* algorithm up to a maximum depth of l. As a quality indicator for the whole network the average over all songs was used. As one can see from figure 6.7 the transformation algorithm radically improved the reachability of the songs in the recommendation graph.

6.3.1.4 Time Complexity

One major advantage of this algorithm is its low runtime complexity of $O(n\log(n))$. At most $n(k_{\text{start}} - k_{\text{min}})$ edges have to be removed. Therefore, we have to perform a maximum of $3n(k_{\text{start}} - k_{min})$ removal or insertion operations on the sorted priority queue. Sorting and removing elements from a priority queue can be done in $O(\log(n))$, e.g., by using a balanced red-black tree. Therefore, removing all the additional edges from the graph can be done in $O(n\log(n))$. The initial insertion operation of all elements in the priority queue is also of complexity $O(n\log(n))$. Thus, the overall complexity of this algorithm is $O(n\log(n))$, which allows to apply this transform even to very large music catalogs. The next section in this chapter presents the second method, namely *Distance Space Normalization* (DSN), that was found to improve the navigability of music recommendation networks.

6.3.2 Distance Space Normalization

While the previous sections have been mainly focusing on recommendation networks resulting from frame-level similarity algorithms, in this section all music recommendation algorithms presented in Chapters 4 and 5 are compared with respect to the reachability within the emerging recommendation network. Hence, the number of hubs and the number of sources were determined for each algorithm. Dataset *"1517-Artists"* and recommendation lists of length $l = 10$ were used. Table 6.3.2 summarizes the result. Obviously, those algorithms (BLS and TAG) that use the *Distance Space Normalization* (DSN) approach presented in **??** to combine individual components produce only a marginal number of hubs and sources. To verify the assumption that the lower number of hubs and sources is a consequence of the DSN approach, the distance matrices produced by all algorithms are finally normalized using the DSN method. It can be seen from table 6.3.2 that the DSN efficiently reduces both the number of hubs and the number of sources.

Still some hubs and sources remain in the recommendation network, but their number is significantly reduced. Furthermore, it seems that those algorithms that combine many components (BLS and TAG) produce the fewest sources and hubs.

	unmodified		using final DSN	
Approach	**#hubs**	**#sources**	**#hubs**	**#sources**
BLS	10	23	6	13
RTBOF	104	236	16	62
SG	244	599	156	224
MLVQ	78	145	4	46
G1C	143	517	65	81
MARSYAS	144	208	39	13
TAG	10	18	8	17
TAGVS	82	121	9	16

Table 6.1: Number of hubs for unmodified similarity algorithms and after finally applying a DSN measured on dataset *"1517-Artists"*.

Thus, not only DSN reduces the number of hubs, but also combining different similarity components leads to a reduction of these navigation issues. This finding that combining many components reduces the number of hubs and sources is in line with some recent research work of Flexer et al. [Flexer et al., 2010], who show that the combination of two well-known similarity measures reduces the number of hubs of the combined similarity algorithm. Therefore, also from a network perspective it seems advisable to combine many components using the DSN operation and as a last step once more normalize the combined distance matrix, which is then called *Extended Distance Space Normalization* (see 5.3.1). However, the main drawback of the DSN approach is that the runtime of the DSN is $O(n^2)$ and consequently the DSN is not a practical solution for large music catalogs containing millions of songs. So one important future research direction will be to try to reduce the runtime complexity of the DSN operation to make similarity measures based on DSN applicable even for very large music catalogs.

6.4 Conclusions

In this chapter the reachability of top-N music recommender systems has been investigated. The theoretical findings are that vertices that are not reachable (called sources) naturally exist in any recommendation network independently of the recommendation approach. It could also be theoretically shown that the number of sources exponentially decreases with increasing number of recommendations per query. However, empirical analysis of recommendation networks shows that the number of unreachable items in a recommendation graph can be a problem for any real world recommender system, as the number of sources in a recommendation network depends on the intrinsic dimensionality of the underlying data. Additionally, it was argued that hubs and sources are related phenomena. Finally, two approaches to reduce these navigation issues of music recommendation networks have been proposed. While it could be shown that similarity algorithms that make use of the distance normalization approach do no longer suffer from the hub-problem that much, also a graph transformation algorithm has been proposed. In contrast to the DSN approach, which has a runtime complexity of $O(N^2)$ this graph transformation algorithm has only a runtime complexity of $O(N \log N)$. Therefore, for small and medium datasets algorithms that combine many components using the DSN method is recommended. For large and very large datasets it is advisable to use vector space music similarity measure, for example the TAGVS algorithm, in combination with the graph transformation algorithm to improve the navigability of the recommendation network.

7 Conclusions and Future Work

The goal of this thesis was to identify ways to further improve content-based music recommender systems. For this reason music recommender systems and their evaluation have been studied in detail on two different abstraction layers. The obtained results and conclusions can be summarized as follows:

- With respect to the evaluation of content-based music recommender systems it is important to note the difference between *music recommendation* and *music similarity*. In a recommendation context only non-trivial similarity relations among songs are of interest. For this purpose it is important to use an artist or portfolio filter, whenever content-based music recommender systems are evaluated via genre classification based quality indicators, so that no over-optimistic quality estimates are reported. Additionally, it turned out that within an evaluation different datasets and evaluation metrics should be considered to provide a comprehensive overview of the obtainable recommendation quality, as the obtained results can vary depending on both the used dataset and the used evaluation metric.

- The conducted experiments analyzing frame-level similarity revealed that BoF approaches are difficult to improve and are in general limited in their ability to identify interesting musical similarities among songs. Three limiting factors have been identified: BoF approaches are not pitch invariant, cannot identify existing similarity relations in case of multiple sound sources and do not capture any temporal information of the analyzed signal. The latter reason was addressed by introducing so-called *block-level* features that can also capture some temporal information of audio signals.

- To make this novel feature set applicable to music recommendation three music similarity measures based on this block-level feature set have been defined (BLS, TAG, TAGVS). Especially the auto-tag based variants (TAG, TAGVS) seem to be an adequate choice to generate music recommendation. In contrast to music similarity algorithms that directly estimate similarity using the extracted audio features, auto-tag based music recommendation algorithms have a semantic interpretable representation. This, for example, allows to explain the generated recommendations and also allows to support exploring the music space using the predicted tags. Furthermore, auto-tag based music similarity algorithms are easy to improve and extend. To improve the quality of an auto-tag based algorithm one can either add new features to the training process, or one can learn new semantic attributes from other tag or genre datasets. Another advantage of the auto-tag based approach (TAGVS) is that it can be used in combination with more powerful search strategies. Consequently, this method is extremely scalable, which makes it a feasible solution even for very large music collections. Altogether, it seems that auto-tag music similarity algorithms will be the future trend in the field of content-based music recommendation.

- However, the listening experiment presented in chapter 3.4.2 revealed that in general collaborative approaches outperform both automatic methods as well as individual human performances with respect to the task of music categorization. This indicates that collaborative music recommender systems outperform content-based music recommender systems in terms of recommendation quality. This raises the question about the future role of content-based music recommender systems. It seems that purely content-based music recommender systems will only be useful in an application context where no access to collaboratively collected metadata is possible or no metadata is available at all. Nevertheless, there will be a growing interest in content-based recommender systems as content-based techniques will be one important component of more complex hybrid music recommenders, for example to resolve the cold-start problem.

- Finally, also the navigability of music recommender systems has been studied. The empirical analysis of music recommendation networks shows that the number of unreachable items in the corresponding recommendation graph can be a problem for any real world recommender. Even more important are the obtained theoretic findings. It could be shown that vertices that are not reachable (called sources) naturally exist in any recommendation network (not only in music recommendation networks) and are *not* a specific property of the underlying recommendation approach. Another important theoretic finding is that the number of sources exponentially decreases with an increasing number of recommendations per query. As a consequence, in general it is advisable to present users longer recommendation lists. Altogether, the conducted network analysis showed that a straightforward top-N recommendation approach is not ideal to explore an item catalog and that navigation issues should be considered already during the development phase of recommendation engines.

Regarding future research work there are at least two major research directions that should be investigated:

One future research direction could focus on improving the quality of automatic tag predictions, as this would directly improve the quality of music recommendation engines that are based on auto-tags. Thus, one idea would be to collect user feedback on auto-tags for songs that have not yet been manually annotated. This way the user feedback could be used to acquire additional ground truth data that could then be used to retrain the tag classifiers.

Another future research direction would be to concentrate on the audio feature side. To improve the quality of audio features decomposing music signals into multiple sources would be an important step. As a first starting point it should be explored how simple decomposition techniques like the percussive-harmonic decomposition presented in [**?**] would affect the quality of audio features.

8 Bibliography

[Adomavicius and Tuzhilin, 2005] Adomavicius, G. and Tuzhilin, A. (2005). "Towards the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions". *IEEE Transactions on Knowledge and Data Engineering*, 17(6), pp. 734–749.

[Anderson, 2006] Anderson, C. (2006). *The Long Tail: Why the Future of Business Is Selling Less of More.* Hyperion.

[Arthur and Vassilvitskii, 2007] Arthur, D. and Vassilvitskii, S. (2007). "k-means++: The Advantages of Careful Seeding". In *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035, New Orleans, Louisiana.

[Ash, 2008] Ash, T. (2008). *Landing Page Optimization: The Definitive Guide to Testing and Tuning for Conversions.* Wiley Verlag.

[Aucouturier, 2006] Aucouturier, J.-J. (2006). *Ten experiments on the modelling of polyphonic timbre.* PhD thesis, University of Paris 6.

[Aucouturier et al., 2007] Aucouturier, J.-J.; Defreville, B.; and Pachet, F. (2007). "The bag-of-frames approach to audio pattern recognition: A sufficient model for urban soundscapes but not for polyphonic music". *The Journal of the Acoustical Society of America*, 122(2), pp. 881–91.

[Aucouturier and Pachet, 2004] Aucouturier, J.-J. and Pachet, F. (2004). "Improving timbre similarity: How high's the sky?". *J. Negative Results Speech Audio Sci.*, 1(1).

[Aucouturier and Pachet, 2008] Aucouturier, J.-J. and Pachet, F. (2008). ”A scale-free distribution of false positives for a large class of audio similarity measures”. *Pattern Recogn.*, 41(1), pp. 272–284.

[Aylon, 2006] Aylon, E. (2006). ”Automatic detection and classification of drum kit sounds.”. Master’s thesis, Universitat Pompeu Fabra.

[Barneveld and Setten, 2004] Barneveld, J. and Setten, M. (2004). ”Designing Usable Interfaces for TV Recommender”. *Human-Computer Interaction Series*, 6, pp. 259–286.

[Barrington et al., 2009] Barrington, L.; Oda, R.; and Lanckriet, G. (2009). ”Smarter than Genius? Human Evaluation of Music Recommender Systems.”. In *Proc. of the Int. Society for Music Information Retrieval.*

[Bella and Peretz, 2005] Bella, S. D. and Peretz, I. (2005). ”Differentiation of classical music requires little learning but rhythm”. *Cognition.*

[Bergstra et al., 2006] Bergstra, J.; Casagrande, N.; Erhan, D.; Eck, D.; and Kégl, B. (2006). ”Aggregate features and ADABOOST for music classification”. *Machine Learning.*

[Bertin-Mahieux et al., 2008] Bertin-Mahieux, T.; Eck, D.; Maillet, F.; and Lamere, P. (2008). ”Autotagger: A Model For Predicting Social Tags from Acoustic Features on Large Music Databases”. *Journal of New Music Research*, 37(2), pp. 115–135.

[Bishop, 2006] Bishop, C. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer.

[Brynjolfsson et al., 2003] Brynjolfsson, E.; Hu, Y. J.; and Smith, M. D. (2003). ”Consumer Surplus in the Digital Economy: Estimating the Value of Increased Product Variety at Online Booksellers”. *Management Science*, 49(11).

[Brynjolfsson et al., 2006] Brynjolfsson, E.; Hu, Y. J.; and Smith, M. D. (2006). ”From Niches to Riches: Anatomy of the Long Tail.”. *Sloan Management Review*, 47(4).

[Burke, 2002] Burke, R. (2002). "Hybrid Recommender Systems: Survey and Experiments". *User Modeling and User-Adapted Interaction*, 12(4), pp. 331–370.

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

[Celma and Cano, 2008] Celma, O. and Cano, P. (2008). "From hits to niches? or how popular artists can bias music recommendation and discovery". In *2nd Workshop on Large-Scale Recommender Systems (ACM KDD)*, Las Vegas, USA.

[Celma and Herrera, 2008] Celma, O. and Herrera, P. (2008). "A New Approach to Evaluating Novel Recommendations". In *2008 ACM Conference on Recommender Systems*, Lausanne, Switzerland.

[Claypool et al., 1999] Claypool, M.; Gokhale, A.; Miranda, T.; Murnikov, P.; Netes, D.; and Sartin, M. (1999). "Combining Content-Based and Collaborative Filters in an Online Newspaper". In *Proc. of ACM SIGIR Workshop on Recommender Systems.*

[Craft et al., 2007] Craft, A.; Wiggins, G. A.; and Crawford, T. (2007). "How Many Beans Make Five? The Consensus Problem in Music-Genre Classification and a New Evaluation Method for Single-Genre Categorisation Systems". In *Proc. Int. Sym. on Music Information Retrieval (ISMIR-07).*

[Donaldson, 2007] Donaldson, J. (2007). "A hybrid social-acoustic recommendation system for popular music". In *Proc. of the 2007 ACM Conference on Recommender Systems (RecSys-07)*, pages 187–190, New York, NY, USA. ACM.

[Duda et al., 2000] Duda, R. O.; Hart, P. E.; and Stork, D. G. (2000). *Pattern Classification (2nd Edition).* Wiley-Interscience.

[Ellis et al., 2002] Ellis, D.; Whitman, B.; Berenzweig, A.; and Lawrence, S. (2002). "The Quest for Ground Truth in Musical Artist Similarity". In *Proc. of the 3rd international Conference on Music Information Retrieval (ISMIR-02).*

[Erdős and Rényi, 1959] Erdős, P. and Rényi, A. (1959). "On random graphs.". *Publ. Math. Debrecen*, 6, pp. 290–297.

[Flexer and Schnitzer, 2009] Flexer, A. and Schnitzer, D. (2009). "Album and Artist effects for Audio Similarity at the Scale of the Web". In *Proc. of the 6th Sound and Music Computing Conference (SMC-09).*

[Flexer et al., 2010] Flexer, A.; Schnitzer, D.; Gasser, M.; and Pohle, T. (2010). "Combining Features Reduces Hubness in Audio Similarity". In *Proc. of the 11th Int. Society for Music Information Retrieval Conference (ISMIR-2010).*

[Flexer et al., 2008] Flexer, A.; Schnitzer, D.; Gasser, M.; and Widmer, G. (2008). "Playlist Generation Using Start and End Songs". In *Proc. Int. Symposium on Music Information Retrieval (ISMIR-08).*

[Foote, 1997] Foote, J. T. (1997). "Content-Based Retrieval of Music and Audio". In *In Multimedia Storage and Archiving Systems II, Proc. of SPIE*, pages 138–147.

[Gasser et al., 2010] Gasser, M.; Flexer, A.; and Schnitzer, D. (2010). "Hubs and Orphans - An Explorative Approach". In *Proc. of the 7th Sound and Music Computing Conference (SMC-2010).*

[Geleijnse et al., 2007] Geleijnse, G.; Schedl, M.; and Knees, P. (2007). "The Quest for Ground Truth in Musical Artist Tagging in the Social Web Era". In *Proc. of the 8th International Conference on Music Information Retrieval (ISMIR-07).*

[Gjerdingen and Perrott, 2008] Gjerdingen, R. and Perrott, D. (2008). "Scanning the Dial: The Rapid Recognition of Music Genres". *Journal of New Music Research.*

[Godfrey and Chordia, 2008] Godfrey, M. T. and Chordia, P. (2008). "Hubs and Homogeneity: Improving Content-Based Music Modelling". In *Proc. of the 9th Int. Conf. on Music Information Retrieval (ISMIR-2008).*

[Goto, 2003] Goto, M. (2003). "SmartMusicKIOSK: Music Listening Station with Chorus-Search Function". In *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology, UIST'03*, pages 31–40.

[Guaus and Herrera, 2006] Guaus, E. and Herrera, P. (2006). "Music Genre Categorization in Humans and Machines". In *121th AES Convention.*

[Hawley, 1993] Hawley, M. (1993). *Structure Out of Sound.* PhD thesis, Massachusetts Institute of Technology. Dept. of Architecture. Program in Media Arts and Sciences.

[Herlocker et al., 2004] Herlocker, J. L.; Konstan, J. A.; Terveen, L. G.; and Riedl, J. T. (2004). "Evaluating collaborative filtering recommender systems". *ACM Trans. Inf. Syst.*, 22(1), pp. 5–53.

[Hoffman et al., 2008] Hoffman, M.; Blei, D.; and Cook, P. (2008). "Content-Based Musical Similarity Computation using the Hierarchical Dirichlet Process". In *In Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR'08)*, pages 349–354, Philadelphia, USA.

[Hoffman et al., 2009] Hoffman, M.; Blei, D.; and Cook, P. (2009). "Easy As CBA: A Simple Probabilistic Model for Tagging Music". In *Proc. of the Int. Sym. on Music Information Retrieval.*

[Holzapfel and Stylianou, 2008] Holzapfel, A. and Stylianou, Y. (2008). "Musical Genre Classification Using Nonnegative Matrix Factorization-Based Features". *IEEE Transactions on Audio, Speech, and Language Processing (TASLP-08).*

[Holzapfel and Stylianou, 2009] Holzapfel, A. and Stylianou, Y. (2009). "A scale transform based method for rhythmic similarity of music". In *Proc. of the 2009 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP-09).*

[Homburg et al., 2005] Homburg, H.; Mierswa, I.; Morik, B. M. K.; and Wurst, M. (2005). "A benchmark dataset for audio classification and clustering". In *Proc. of the 6th Int. Sym. on Music Information Retrieval (ISMIR-05).*

[Jensen et al., 2009] Jensen, J. H.; Christensen, M. G.; and Jensen, S. (2009). "A tempo-insensitve Representation of Rhythmic Patterns". In *Proc. of the 17th European Signal Processing Conf. (EUSIPCO-09).*

[Jiang et al., 2002] Jiang, D.; Lu, L.; Zhang, H.; Tao, J.; and Cai, L. (2002). "Music Type Classification by Spectral Contrast Feature". In *Proc. IEEE Int. Conf. on Multimedia and Expo (ICME-02).*

[Karypis, 2001] Karypis, G. (2001). "Evaluation of Item-Based Top-N Recommendation Algorithms". In *Proc. of the 10th Int. Conf. on Information and Knowledge Management (CIKM-'01).*

[Kay et al., 2001] Kay, J.; Kummerfeld, B.; and Lauder, P. (2001). "Foundations for personalised documents: a scrutable user model server.". In *Proc. of the Australian Document Computing Symposium (ADCS' 2001).*

[Keum and Lee, 2006] Keum, J. and Lee, H. (2006). "Speech/Music Discrimination Based on Spectral Peak Analysis and Multi-layer Perceptron". In *Proc. of the 9th International Conference on Hybrid Information Technology (ICHIT'06), Cheju Island, Korea*, pages 56–61.

[Knees, 2007] Knees, P. (2007). "Search and Select - Intuitively Retrieving Music from Large Collections". In *Proc. of the 8th International Conference on Music Information Retrieval (ISMIR-07)*, Vienna, Austria.

[Knees et al., 2008] Knees, P.; Pohle, T.; Schedl, M.; Schnitzer, D.; and Seyerlehner, K. (2008). "A Document-centered Approach to a Natural Language Music Search Engine". In *Proc. of the 30th European Conference on Information Retrieval (ECIR-08)*, Glasgow, Scotland, UK.

[Kullback and Leibler, 1951] Kullback, S. and Leibler, R. A. (1951). "On information and sufficiency". *Annals of Mathematical Statistics*, 22(1), pp. 79–86.

[Lam. and Riedl, 2004] Lam., S. K. and Riedl, J. (2004). "Shilling recommender systems for fun and profit". In *Proc. of the 13th Int. Conf. on World Wide Web (WWW-04).*

[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*, pages 27–36, Santa Barbara, California, USA.

[Li et al., 2003] Li, T.; Ogihara, M.; and Li, Q. (2003). "A comparative study on content-based music genre classification". In *Proc. of the 26th ACM SIGIR Conf. on Research and Development in Informaion Retrieval.*

[Lidy and Rauber, 2005] Lidy, T. and Rauber, A. (2005). "Evaluation of feature extractors and psycho-acoustic transformations for music genre classification.". In *Proc. of the 6th Int. Conf. on Music Information Retrieval (ISMIR-05).*

[Lidy et al., 2007] Lidy, T.; Rauber, A.; Pertusa, A.; and Inesta, M. (2007). "Improving Genre Classification by Combination of Audio and Symbolic Descriptors Using a Transcription System". In *Proc. of the 8th International Conference on Music Information Retrieval (ISMIR-07).*

[Lippens et al., 2004] Lippens, S.; Martens, J.; Mulder, T. D.; and Tzanetakis, G. (2004). "A Comparison of Human and Automatic Musical Genre Classification". In *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP-04).*

[Lloyd, 1982] Lloyd, S. (1982). "Least squares quantization in PCM". *IEEE Transactions on Information Theory*, 28(2), pp. 129–137.

[Logan and Salomon, 2001] Logan, B. and Salomon, A. (2001). "A music similarity function based on signal analysis". In *In proceedings IEEE International Conference on Multimedia and Expo (ICME)*, Tokyo, Japan.

[Mandel and Ellis, 2005] Mandel, M. and Ellis, D. (2005). "Song-Level features and SVMs for music classification". In *Proc. of the 6th Int. Conf. on Music Information Retrieval.*

[McKay and Fujinaga, 2006] McKay, C. and Fujinaga, I. (2006). "Musical genre classification: Is it worth pursuing and how can it be improved?". In *Proc. of the 7th Int. Conf. on Music Information Retrieval (ISMIR-06).*

[Minami et al., 1997] Minami, K.; Akutsu, A.; Hamada, H.; and Tonomura, Y. (1997). "Enhanced Video handling based on Audio Analysis". In *Proc. of the 1997 International Conference on Multimedia Computing and Systems (ICMCS'97), Washington, DC, USA*, pages 219–226.

[Minami et al., 1998] Minami, K.; Akutsu, A.; Hamada, H.; and Tonomura, Y. (1998). "Video Handling with Music and Speech Detection". *IEEE MultiMedia*, 5(3), pp. 17–25.

[Moore, 1998] Moore, B. C. J. (1998). *Cochlear hearing loss.* Whurr Publishers Ltd.

[Ness et al., 2009] Ness, S. R.; Theocharis, A.; Tzanetakis, G.; and Martins, L. G. (2009). "Improving automatic music tag annotation using stacked generalization of probabilistic SVM outputs". In *Proc. of the 17th ACM Int. Conf. on Multimedia (MM-09).*

[Newman, 2003] Newman, M. E. J. (2003). "The Structure and Function of Complex Networks". *SIAM Review*, 45, pp. 167–256.

[Oppenheim et al., 2004] Oppenheim, A. V.; Schafer, R. W.; and Buck, J. R. (2004). *Zeitdiskrete Signalverarbeitung.* Pearson Studium.

[Pachet and Cazaly, 2000] Pachet, F. and Cazaly, D. (2000). "A Taxonomy of Musical Genre". In *Proc. of Content-Based Multimedia Information Access Conference (RIOA).*

[Pampalk, 2010] Pampalk, E. (2010). *Computational Models in Music Similarity and their Application in Music Infromation Retrieval.* PhD thesis, Vienna University of Technology.

[Pampalk et al., 2005] Pampalk, E.; Flexer, A.; and Widmer, G. (2005). "Improvements of Audio-Based Music Similarity and Genre Classification". In *In Proc of the 6th Int.l Conf. 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 *Proc. of the 10th ACM Int. Conf. on Multimedia.*

[Panagakis et al., 2008] Panagakis, I.; Benetos, E.; and Kotropoulos, C. (2008). "Music Genre Classification: A Multilinear Approach". In *Proc. of the 9th International Conference on Music Information Retrieval (ISMIR-08).*

[Panagakis et al., 2009] Panagakis, Y.; Kotropoulos, C.; and Arce, G. (2009). "Music Genre Classification using Locality Preserving Non-Negative Tensor Factorization and Sparse Representations". In *10th International Society for Music Information Retrieval Conference (ISMIR 2009).*

[Peeters, 2004] Peeters, G. (2004). "A large set of audio features for sound description (similarity and classification) in the CUIDADO project". Technical report, Institut de Recherche et Coordination Acoustique/Musique (IRCAM).

[Pohle, 2010] Pohle, T. (2010). *Automatic Characterization of Music for Intuitive Retrieval.* PhD thesis, Johannes Kepler University.

[Pohle and Schnitzer, 2007] Pohle, T. and Schnitzer, D. (2007). "Striving for an Improved Audio Similarity Measure". In *online Proc. of the 4th Annual Music Information Retrieval eXchange (MIREX-07).*

[Pohle et al., 2009] Pohle, T.; Schnitzer, D.; Schedl, M.; Knees, P.; and Widmer, G. (2009). "On Rhythm and General Music Similarity". In *Proc. of the 10th Int. Society for Music Information Retrieval Conf. (ISMIR-09).*

[Pye, 2006] Pye, D. (2006). "Content-based methods for the management of digital music". In *ICASSP '00: Proceedings of the Acoustics, Speech, and Signal Processing, 2000. on IEEE International Conference*, pages 2437–2440, Washington, DC, USA.

[Radovanovic et al., 2009] Radovanovic, M.; Nanopoulos, A.; and Ivanovic, M. (2009). "Nearest Neighbors in High-Dimensional Data: The Emergence and Influence of Hubs". In *Proc. of the 26th Int. Conf. on Machine Learning.*

[Rauber et al., 2003] Rauber, A.; Pampalk, E.; and Merkl, D. (2003). *The SOM-enhanced JukeBox: Organization and visualization of music collections based on perceptual models.* Journal of New Music Research.

[Rubner et al., 1998] Rubner, Y.; Tomasi, C.; and Guibas, L. J. (1998). "A Metric for Distributions with Applications to Image Databases". In *In Proceedings of the 1998 IEEE International Conference on Computer Vision, ICCV'98*, pages 59–66, Bombay, India.

[Sarwar et al., 2001] Sarwar, B.; Karypis, G.; Konstan, J.; and Reidl, J. (2001). "Item-based collaborative filtering recommendation algorithms". In *WWW '01: Proc. of the 10th Int. Conf. on World Wide Web.*

[Schedl and Knees, 2009] Schedl, M. and Knees, P. (2009). "Context-based Music Similarity Estimation". In *Proceedings of the 3rd International Workshop on Learning the Semantics of Audio Signals (LSAS-09)*, Graz, Austria.

[Schein et al., 2002] Schein, A.; Popescul, A.; Ungar, L. H.; and Pennock, D. M. (2002). "Methods and metrics for cold-start recommendations". In *SIGIR '02: Proc. of the 25th Int. ACM SIGIR Conf. on Research and development in information retrieval*, New York, NY, USA.

[Schwarz and Rodet, 1999] Schwarz, D. and Rodet, X. (1999). "Spectral Envelope Estimation and Representation for Sound Analysis-Synthesis". In *Proceedings of the ICMC*, pages 351–354.

[Seyerlehner et al., 2009a] Seyerlehner, K.; Flexer, A.; and Widmer, G. (2009a). "On the limitations of browsing top-N recommender systems". In *Proc. of the third ACM Conf. on Recommender systems (RecSys-09).*

[Seyerlehner et al., 2009b] Seyerlehner, K.; Knees, P.; Schnitzer, D.; and Widmer, G. (2009b). "Browsing Music Recommendation Networks". In *10th International Society for Music Information Retrieval Conference.*

[Seyerlehner et al., 2009c] Seyerlehner, K.; Pohle, T.; Widmer, G.; and Schnitzer, D. (2009c). "Infromed Selection of Frames for Music Similarity Computation". In *Proc. of the 12th Int. Conf. on Digital Audio Effects (DAFx-09).*

[Seyerlehner and Schedl, 2009] Seyerlehner, K. and Schedl, M. (2009). "Block-Level Audio Feature for Music Genre Classification". In *online Proc. of the 5th Annual Music Information Retrieval Evaluation eXchange (MIREX-09).*

[Seyerlehner and Schedl, 2010] Seyerlehner, K. and Schedl, M. (2010). "Using Block-Level Features for Genre Classification, Tag Classification and Music Similarity Estimation". In *online Proc. of the 6th Annual Music Information Retrieval Evaluation eXchange (MIREX-10).*

[Seyerlehner et al., 2008] Seyerlehner, K.; Widmer, G.; and Knees, P. (2008). "Frame level audio similarity - A codebook approach". In *Proc. of the 11th International Conference on Digital Audio Effects (DAFx-08).*

[Seyerlehner et al., 2010a] Seyerlehner, K.; Widmer, G.; and Knees, P. (2010a). "A Comparison of Human, Automatic and Collaborative Music Genre Classification and User Centric Evaluation of Genre Classification Systems". In *Proc. of the 8th Int. Workshop on Adaptive Multimedia Retreival (AMR-10).*

[Seyerlehner et al., 2010b] Seyerlehner, K.; Widmer, G.; and Pohle, T. (2010b). "Fusing Block-level Features for Music Similarity Estimation". In *Proc. of the 13th International Conference on Digital Audio Effects (DAFx-10).*

[Soltau et al., 2010] Soltau, H.; Schultz, T.; Westphal, M.; and Waibel, A. (2010). "Recognition of Music Types". In *Proc. of the 23th IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP-98).*

[Sordo et al., 2008] Sordo, M.; Celma, O.; Blech, M.; and Guaus., E. (2008). "The Quest for Musical Genres: Do the Experts and the Wisdom of Crowds Agree?". In *Proc. of the 9th International Conference on Music Information Retrieval (ISMIR-08).*

[Stanley et al., 1937] Stanley, S. S.; Volkman, J.; and Newman, E. (1937). "A scale for the measurement of the psychological magnitude of pitch". *The Journal of the Acoustical Society of America*, 8(3).

[Swain and Ballard, 1991] Swain, M. and Ballard, D. (1991). "Color Indexing". *International Journal of Computer Vision*, 7(1), pp. 11–32.

[Tiemann and Pauws, 2007] Tiemann, M. and Pauws, S. (2007). "Towards ensemble learning for hybrid music recommendation". In *RecSys '07: Proceedings of* *the 2007 ACM conference on Recommender systems*, pages 177–178, New York, NY, USA. ACM.

[Tintarev and Masthoff, 2006] Tintarev, N. and Masthoff, J. (2006). "Similarity for News Recommender Systems". In *Proc. of the 4th Int. Conf. of Adaptive Hypermedia and Adaptive Web-Based Systems (AH-06).*

[Tintarev and Masthoff, 2007] Tintarev, N. and Masthoff, J. (2007). "A Survey of Explanations in Recommender Systems". In *Proc. of the 2007 IEEE 23rd International Conference on Data Engineering Workshop.*

[Turnbull et al., 2008] Turnbull, D.; Barrington, L.; Torres, D.; and Lanckriet, G. (2008). "Semantic Annotation and Retrieval of Music and Sound Effects". *Audio, Speech, and Language Processing, IEEE Transactions on*, 16(2), pp. 467–476.

[Tzanetakis and Cook, 2002] Tzanetakis, G. and Cook, P. (2002). "Musical Genre Classification of Audio Signal". *IEEE Transactions on Audio and Speech Processing*, 10(5), pp. 293–302.

[Vignoli and Pauws, 2005] Vignoli, F. and Pauws, S. (2005). "A music retrieval system based on user-driven similarity and its evaluation". In *In Proceedings of the 6th International Conference on Music Information Retrieval, ISMIR'05*, London, UK.

[Wang et al., 2006] Wang, F.; Ma, S.; Yang, L.; and Li, T. (2006). "Recommendation on Item Graphs". In *Proc. of the Sixth International Conference on Data Mining (ICDM-06)*, pages 1119–1123, Washington, DC, USA. IEEE Computer Society.

[West et al., 2006] West, K.; Cox, S.; and Lamere, P. (2006). "Incorporating machine-learning into music similarity estimation". In *Proc. of the 1st ACM workshop on Audio and music computing multimedia (AMCMM-06)*, New York, NY, USA.

[Yoshii et al., 2008] Yoshii, K.; Goto, M.; Komatani, K.; Ogata, T.; and Okuno, H. (2008). "An Efficient Hybrid Music Recommender System Using an Incrementally Trainable Probabilistic Generative Model". *Unbekannte Zeitschrift*, 16(2), pp. 435–447.

[Ziegler et al., 2005] Ziegler, C.-N.; McNee, S. M.; Konstan, J. A.; and Lausen, G. (2005). "Improving recommendation lists through topic diversification". In *Proc. of the 14th international conference on World Wide Web (WWW-05).*

[Zölzer, 2002] Zölzer, U. (2002). *DAFX - Digital Audio Effects.* WILEY, Southern Gate, Chichester, England.

[Zwicker, 1961] Zwicker, E. (1961). "Subdivision of the audible frequency range into critical bands". *Journal of the Acoustical Society of America*, 33(2), pp. 248–248.

9 APPENDIX A - Datasets

’103-Artists’

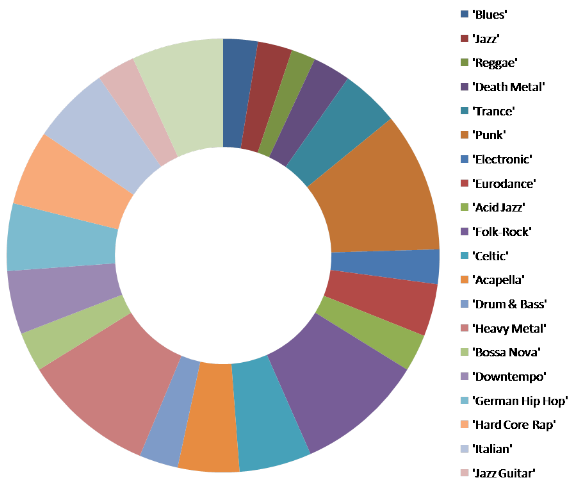


Figure 9.1: Visualization of the class distribution.

genre	**#tracks**	**#artists**
Blues	63	100
Jazz	63	103
Reggae	44	87
Death Metal	69	103
Trance	106	82
Punk	255	83
Electronic	63	46
Eurodance	96	98
Acid Jazz	68	86
Folk-Rock	234	117
Celtic	132	116
Acapella	112	92
Drum & Bass	71	113
Heavy Metal	242	76
Bossa Nova	72	76
Downtempo	116	71
German Hip Hop	123	74
Hard Core Rap	137	98
Italian	142	68
Jazz Guitar	70	72
Melodic Metal	167	72
total	**2445**	**1517**

Table 9.1: Detailed song and artist distribution.

'1517-Artists'

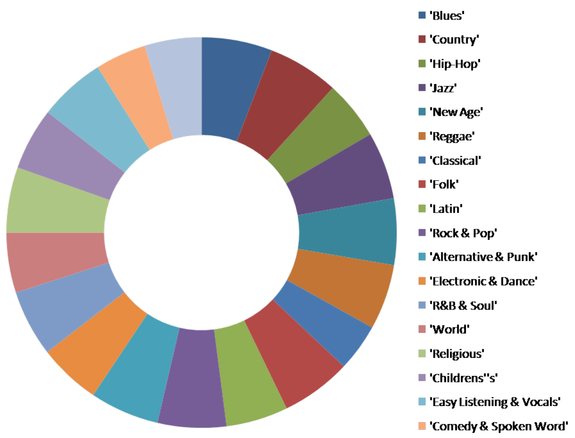


Figure 9.2: Visualization of the class distribution.

genre	#tracks	#artists
Blues	186	100
Country	187	103
Hip-Hop	155	87
Jazz	177	103
New Age	175	82
Reggae	172	83
Classical	125	46
Folk	185	98
Latin	163	86
Rock & Pop	181	117
Alternative & Punk	182	116
Electronic & Dance	164	92
R&B & Soul	175	113
World	158	76
Religious	172	71
Children's	164	74
Easy Listening & Vocals	175	98
Comedy & Spoken Word	134	68
Soundtracks & More	150	72
total	**3180**	**1517**

Table 9.2: Detailed song and artist distribution.

’Annotated’

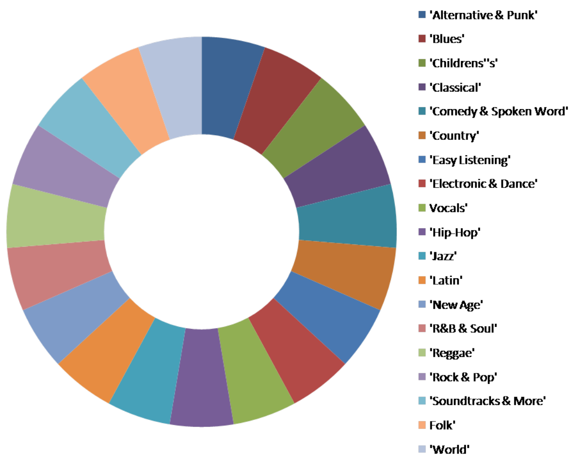


Figure 9.3: Visualization of the class distribution.

genre	**#tracks**	**#artists**
Alternative & Punk	10	10
Blues	10	10
Childrens’s	10	10
Classical	10	10
Comedy & Spoken Word	10	10
Country	10	10
Easy Listening	10	10
Electronic & Dance	10	10
Vocals	10	10
Hip-Hop	10	10
Jazz	10	10
Latin	10	10
New Age	10	10
R&B & Soul	10	10
Reggae	10	10
Rock & Pop	10	10
Soundtracks & More	10	10
Folk	10	10
World & More	10	10
total	**190**	**190**

Table 9.3: Detailed song and artist distribution.

'Unique'

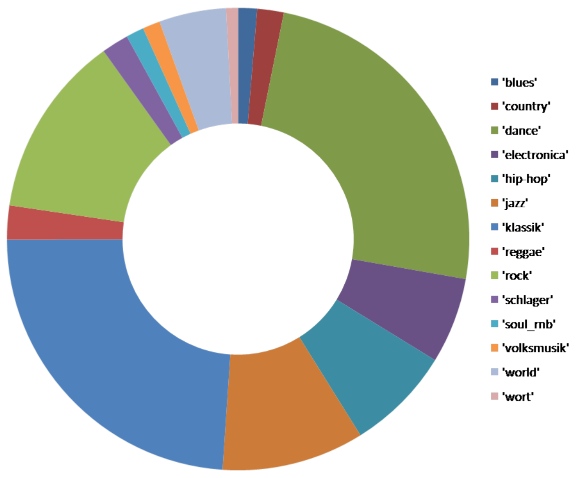


Figure 9.4: Visualization of the class distribution.

genre	**#tracks**	**#artists**
blues	41	41
country	58	58
dance	766	766
electronica	187	187
hip-hop	229	229
jazz	310	310
klassik	744	744
reggae	74	74
rock	398	398
schlager	59	59
soul_rnb	39	39
volksmusik	38	38
world	146	146
wort	26	26
total	**3115**	**3115**

Table 9.4: Detailed song and artist distribution.

'GTZAN'

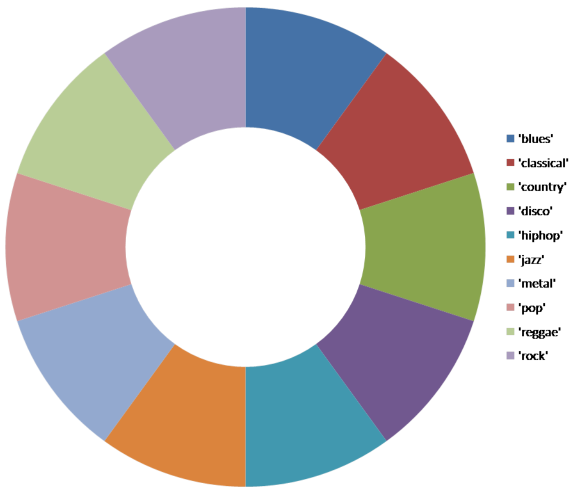


Figure 9.5: Visualization of the class distribution.

genre	**#tracks**	**#artists**
blues	10	N/A
classical	10	N/A
country	10	N/A
disco	10	N/A
hiphop	10	N/A
jazz	10	N/A
metal	10	N/A
pop	10	N/A
reggae	10	N/A
rock	10	N/A
total	**1000**	**N/A**

Table 9.5: Detailed song and artist distribution.

'Homburg'

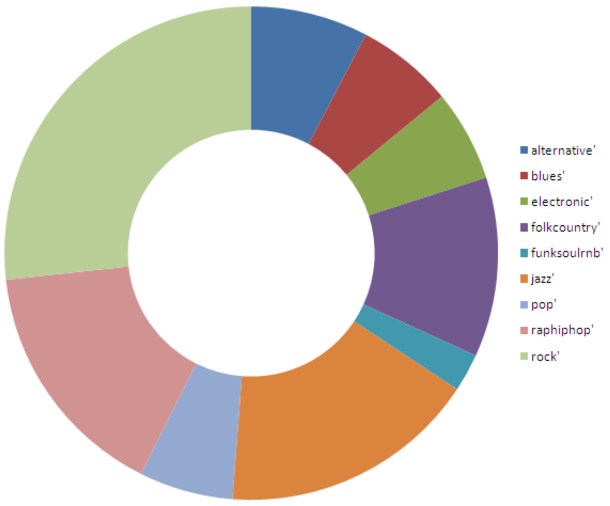


Figure 9.6: Visualization of the class distribution.

genre	#tracks	#artists
alternative	145	121
blues	120	80
electronic	113	97
folkcountry	222	177
funksoulrnb	47	39
jazz	319	214
pop	116	106
raphiphop	300	210
rock	504	448
total	**1886**	**1463**

Table 9.6: Detailed song and artist distribution.

’ISMIR 2004 Genre’

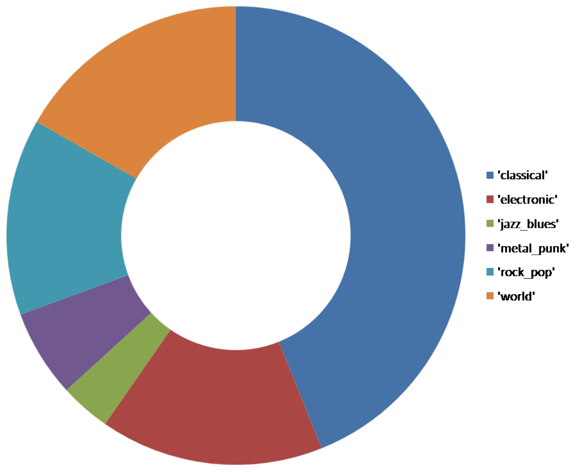


Figure 9.7: Visualization of the class distribution.

genre	#tracks	#artists
classical	640	N/A
electronic	229	N/A
jazz_blues	52	N/A
metal_punk	90	N/A
rock_pop	203	N/A
world	244	N/A
total	**1458**	**N/A**

Table 9.7: Detailed song and artist distribution.

Curriculum Vitae

Klaus Seyerlehner received his Master in Computer Science from the Johannes Kepler University in June 2006. Since 2006 he has been working as a research assistant at the Department of Computational Perception at the Johannes Kepler University. As a PhD candidate supervised by Prof. Dr. Gerhard Widmer he was engaged in three scientific research projects: *Music Retrieval Beyond Simple Audio Similarity* (L511-N15), *Operational Models of Music Similarity for Music Information Retrieval* (L112-N04) and *I2M — Interfaces to Music*. During his work as a research assistant he (co-)authored more than 18 refereed conference papers and was a member of Prof. Widmer's Wittgenstein team. Furthermore, Klaus Seyerlehner served as reviewer for several conferences (International Society for Music Information Retrieval (ISMIR), International Conference on Multimedia Modeling (MMM), International Symposium on Methodologies for Intelligent Systems (ISMIS)) and several journals (IEEE Transactions on Audio, Speech, and Language Processing, Journal on Advances in Signal Processing (EURASIP) and Pattern Recognition Letters (Elsevier)). He also supervised several student projects, bachelor and diploma theses. His main research interests are related to Digital Signal Processing, Pattern Recognition, Machine Learning, Statistics and Recommender Systems.

„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.“

Linz, am ....