A content-based music recommender system

Juuso Kaitila

University of Tampere
Faculty of Natural Sciences
Computer Science
Master's thesis
Supervisor: Martti Juhola
18.5.2017

University of Tampere
Faculty of Natural Sciences
Juuso Kaitila: A content-based music recommender system
Master's thesis, 65 pages
May 2017

Abstract
Music recommender have become increasingly relevant due to increased accessibility provided by various music streaming services. Some of these streaming services, such as Spotify, include a recommender system of their own. Despite many advances in recommendation techniques, recommender systems still often do not provide accurate recommendations.

This thesis provides an overview of the history and developments of music information retrieval from a more content-based perspective. Furthermore, this thesis describes recommendation as a problem and the methods used for music recommendation with special focus on content-based recommendation by providing detailed descriptions on the audio content features and content-based similarity measures used in content-based music recommender systems. Some of the presented features are used in our own content-based music recommender.

Both objective and subjective evaluation of the implemented recommender system further confirm the findings of many researchers that music recommendation based solely on audio content does not provide very accurate recommendations.

Keywords: music recommendation, recommender system, music information retrieval, content-based recommendation, music similarity

Contents

1 Introduction

Music recommender systems have become increasingly relevant with the advent of music streaming services such as Spotify and Tidal, which have made music more accessible. The recommender systems help users discover new music by providing recommendations, for example, in the form of automatically generated playlists containing songs the recommender system thinks the user might like.

Current state-of-the-art music recommender systems use user-generated metadata, such as previous purchases and listening history, as the basis for the recommendations. However, such metadata-based systems cannot recommend artists or songs for which there is no data available (i.e., new songs or artists). This ”cold start” problem has made researchers focus on improving content-based recommender systems, which use audio features extracted automatically from the audio content as the basis for recommendations. McFee, Barrington and Lanckriet (2012) state that the construction of features and the definition of similarity in such systems are frequently ad-hoc and not explicitly optimized for the specific task.

The varying definitions of similarity can be explained by the lack of a ground truth, which is due to music similarity being subjective, multifaceted and a moving target (Berenzweig, Logan, Ellis, & Whitman, 2004). Berenzweig *et al.* (2004) note that subjective judgments of the similarity between specific pairs of artists are not consistent between listeners and may vary with an individual's mood or evolve over time. Additionally, the similarity between two artists can be answered from multiple perspectives as music may be similar or distinct in terms of virtually any property that can be used to describe music such as genre, melody, rhythm, geographical origin and instrumentation. These factors make the problem of music recommendation complicated.

This thesis focuses on content-based music recommendation and describes the some of the key audio features and similarity measures used in content-based recommender systems. Chapter 2 gives a formal definition of recommendation as a problem and lists the factors that affect recommendations. Challenges specific to music recommendation are also discussed. Chapter 3 discusses the history and development of the music information retrieval research field with focus on content-based recommendation. Some research related to other recommendation techniques is briefly discussed. Chapter 4 details the recommendation techniques that have been used for music recommendation and how they work. Chapter 5 describes how commonly used features describing the audio content can be computed from the audio signal. Chapter 6 details some of the more common content-based similarity measures used by researchers. Finally, Chapter 7 describes the dataset, implementation and evaluation of our own content-based music recommender system.

2 Recommendation as a problem

According to Celma (2010, p. 15), the recommendation problem can be split into two subproblems: a prediction problem and a recommendation problem. The first one is about the estimation of the items' likeliness for a given user and the second problem is to recommend a list of N items, which is reduced to list the top-N items once the system can predict items into a totally ordered set.

Sarwar *et al.* (2001) formalize the prediction problem as follows: let $U = \{u_1, u_2, \ldots, u_m\}$ be the set of m users, and let $I = \{i_1, i_2, \ldots, i_n\}$ be the set of n recommendable items. Each user u_i has a list of items I_{u_i}, which represents the items that the user has expressed his or her opinion about through explicit or implicit feedback. Note that $I_{u_i} \subseteq I$, and that I_{u_i} can be empty, $I_{u_i} = \emptyset$. The function P_{u_a,i_j} is the predicted likeliness of item $i_j \notin I_{u_a}$ for the active user u_a.

Sarwar *et al.* (2001) also formalize the recommendation problem as follows: Recommendation is a list of N items, $I_r \subset I$, that the user will like the most, i.e., the N items with the highest P_{u_a,i_j} values. The recommended list should not contain items from the user's interests, i.e., $I_r \cap I_{u_a} = \emptyset$.

The set I of possible items can be very large, which is also true for the user set U. In most recommender systems, the prediction function P_{u_a,i_j} is usually represented by a rating, which is given by the user either explicitly or implicitly through some measures, e.g., by tracking if a song is skipped. They are represented as triples $\langle u, i, r \rangle$ where r is the rating value assigned by the user u to a particular item i. The value is usually a real number (e.g., from 0 to 1), a value in a discrete range (e.g., from 1 to 5), or a binary variable (e.g., like/dislike). (Celma, 2010)

2.1 Factors that affect recommendations

There are several factors that affect the quality of recommendations:

- **Novelty** - A high rate of novel recommendations can make the quality of recommendations seem poor to the user. Recommending some familiar items increases the user's confidence in the recommender. (Herlocker, Konstan, Terveen, & Riedl, 2004)

- **Serendipity** - A recommender should help the user discover unexpected yet interesting items that they might not be able to discover otherwise. (Herlocker et al., 2004)

- **Explainability** - Giving explanations about recommended items can improve the user's trust in the recommender system. (Herlocker, Konstan, & Riedl, 2000)

- **Cold start problem** - When a new user or item enters the system the lack of data prevents the system from giving useful recommendations. (Celma, 2010, p. 36)

- **Data sparsity and high dimensionality** - High dimensionality of both users and items can result in low coverage of users' interactions with the items. (Celma, 2010, p. 36)

- **Coverage** - Low coverage of the domain limits the space of possible items to be recommended. (Herlocker et al., 2004)

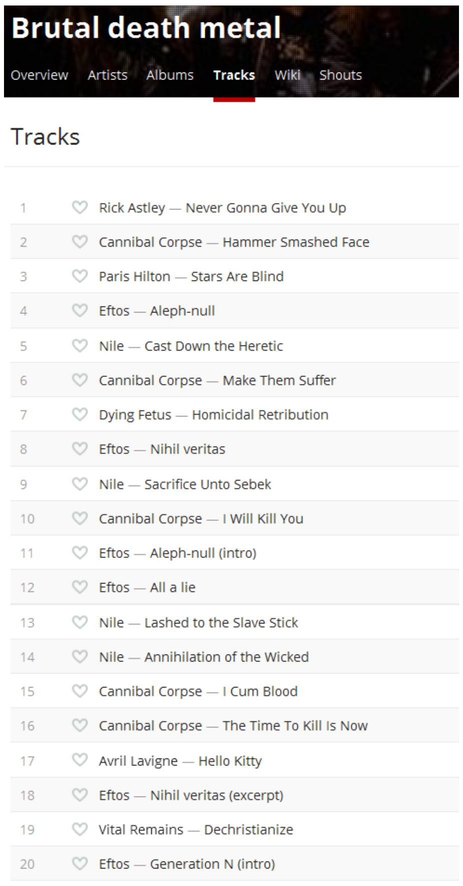
- **Trust** - Recommender systems that are trust-aware determine which users can be reliably used for recommendations, and which cannot. (Celma, 2010, p. 37)

- **Attacks** - Recommender systems can be attacked, which reduces the quality of recommendations (Celma, 2010, p. 37). An example of an attack is deliberate mistagging, which happens when a group of users tag an item using a false or malicious tag. For example on Last.fm, several users have tagged Rick Astley's Never Gonna Give You Up as brutal death metal, which has made it the top brutal death metal track[1]. Figure 1 is a screenshot taken of Last.fm's top brutal death metal tracks, which include three deliberately mistagged songs (Rick Astley's Never Gonna Give You Up, Paris Hilton's Stars Are Blind, and Avril Lavigne's Hello Kitty).

- **Temporal effects** - Recommender systems can treat older items as less relevant than the new ones. The system has to decide which items from a user profile are taken into account when computing the recommendations. (Celma, 2010, p. 37)

[1]https://www.last.fm/tag/brutal+death+metal/tracks

- **Psychological factors and musical taste** - Certain aspects of personality along with socio-economic status, age, and sex are correlated with music preference. Additionally, different cultural groups have different distributions for music preference. (Uitdenbogerd & van Schyndel, 2002)



Tracks

1	Rick Astley — Never Gonna Give You Up
2	Cannibal Corpse — Hammer Smashed Face
3	Paris Hilton — Stars Are Blind
4	Eftos — Aleph-null
5	Nile — Cast Down the Heretic
6	Cannibal Corpse — Make Them Suffer
7	Dying Fetus — Homicidal Retribution
8	Eftos — Nihil veritas
9	Nile — Sacrifice Unto Sebek
10	Cannibal Corpse — I Will Kill You
11	Eftos — Aleph-null (intro)
12	Eftos — All a lie
13	Nile — Lashed to the Slave Stick
14	Nile — Annihilation of the Wicked
15	Cannibal Corpse — I Cum Blood
16	Cannibal Corpse — The Time To Kill Is Now
17	Avril Lavigne — Hello Kitty
18	Eftos — Nihil veritas (excerpt)
19	Vital Remains — Dechristianize
20	Eftos — Generation N (intro)

Figure 1. Top 20 brutal death metal tracks on Last.fm with three mistagged songs. Screenshot taken on March 12, 2017.

2.2 Music recommendation

The recommendation problem in the music domain has additional challenges as individual's music perception depends on many factors. Lesaffre *et al.* (2006) discovered that music perception is affected by the context of the user. They found subject dependencies for age, music expertise, musicianship, music taste and familiarity with the music. Furthermore, Berenzweig *et al.* (2004) state that subjective judgments of similarity between artists are not consistent between listeners and may vary with individual's mood or evolve over time. They emphasize that music which holds no interest for a given subject very frequently "sounds the same." Music can be similar or distinct in terms of virtually any property that can be used to describe music such as genre, melody, rhythm, geographical origin and instrumentation, which makes it possible to answer the question of similarity between two artists from multiple perspectives.

The UK-based *Phoenix 2* Project (Jennings, 2007) analyzed the different types of listeners with an age group ranging from 16 to 45. The project classified the listeners based on four degrees of interest in music as follows:

- **Savants.** Everything in life seems to be tied up with music and their musical knowledge is very extensive. They represent 7% of the 16-45 age group.
- **Enthusiasts.** Music is a key part of life but is also balanced by other interests. They represent 21% of the 16-45 age group.
- **Casuals.** Music plays a welcome role, but other things are far more important. They represent 32% of the 16-45 age group.
- **Indifferents** would not lose much sleep if music ceased to exist. They represent 40% of the 16-45 age group and they are the predominant type of listeners in the whole population.

According to Celma (2010, p. 46), each type of listener needs different type of recommendations. Savants are very exigent and are thus the most difficult listeners to provide recommendations to. They need risky and clever recommendations instead of popular ones. Enthusiasts on the other hand appreciate a balance between interesting, unknown, and familiar recommendations. Casuals and indifferents, who represent 72% of the population, do not need complicated recommendations and popular mainstream music that they can easily identify with would fit their musical needs. Thus, it is important for a recommender system to be able to detect the type of user and act accordingly.

Some researchers, e.g., (Celma & Serra, 2008) assert that there exists a "semantic gap" between content object descriptions and concepts that humans use to relate to music in content-based music recommendation that makes it more difficult to provide accurate recommendations. The term has received some criticism, e.g., Wiggins (2009), as being misleading due to there existing plenty of musical syntax that is perceived and relevant to the listening experience, but it is not explicit in the audio signal. The term also has many different interpretations, which makes the concept slippery. Wiggins (2009) agrees that there exists a "semantic gap" from the perspective of the audio domain, but such "gap" is not visible from the perspective of the auditory domain, in which is a discrete spectrum of structure that is realized in or stimulated by an audio signal and is theoretically explicable in both psychological and musicological terms.

Celma and Serra (2008) define three levels of abstraction for describing multimedia objects: low-level basic features, mid-level semantic features and human understanding. The low-level includes physical features of the object, such as the bit depth of an audio file, and basic features such as the pitch salience of an audio frame. The mid-level abstraction aims to describe concepts such as the genre of a song.

These abstractions can also be used to describe a the music information plane (Figure 2), in which one dimension represents the different media types that serve as input data and the other dimension is the level of abstraction in the information extraction process of this data. The semantic gap is between the mid-level abstraction (content objects) and higher-level information related to the users representing the distance between the descriptions that can be extracted from the input sources and the end user.

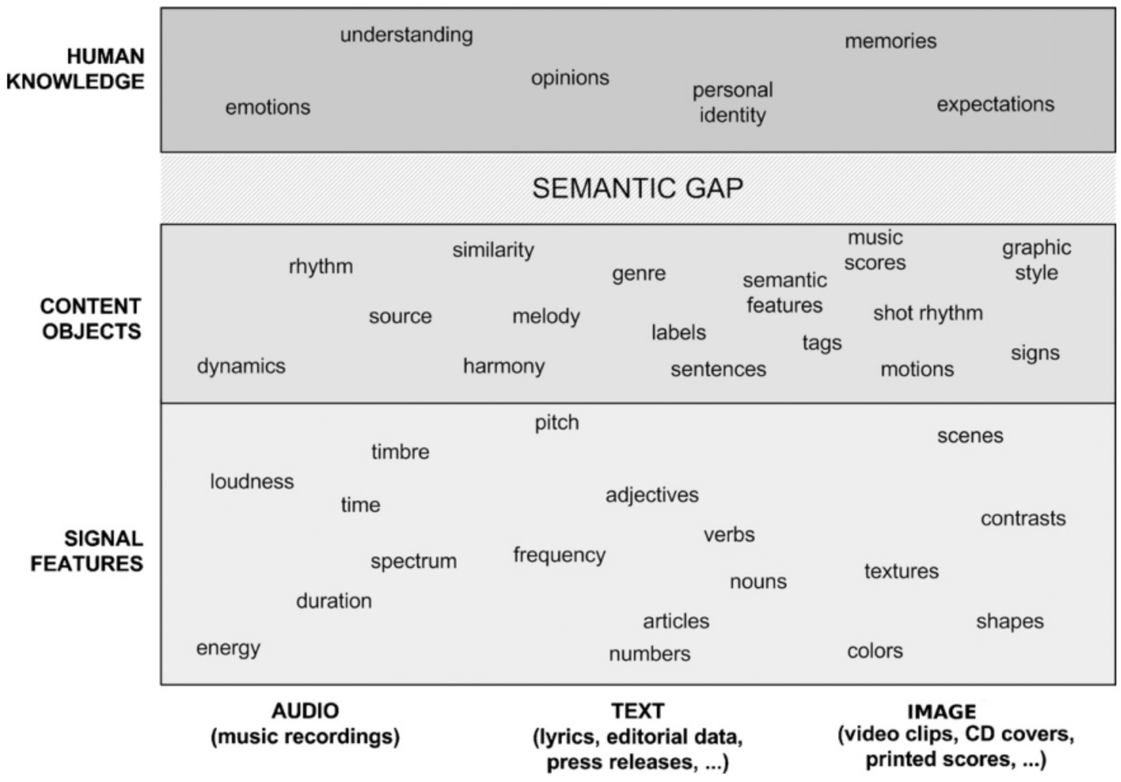


Figure 2. The music information plane and the semantic gap between human understanding and content object descriptions. (Celma & Serra, 2008)

3 Related work

Music information retrieval (MIR) is a research field that comprises several subfield and research tasks. The core applications, which drive the research, are music retrieval, music recommendation, automatic playlist generation, and music browsing interfaces.

One of the most important topic groups for MIR research is the automatic extraction of meaningful features from audio content and context. The extracted features are used to compute similarity between two songs or to classify music based on some criteria such as mood, instrumentation, or genre. The features, similarity measures, and classification methods are used extensively in music recommendation and automatic playlist generation.

The earliest work on content-based audio retrieval and content-based music similarity, such as that of Wold *et al.* (1996), used only very basic aspects of the sound, namely loudness, pitch, brightness, bandwidth, and harmonicity; for building a database of feature vectors that were classified with weighted Euclidean distance. The sounds could then be retrieved based on their classes such as "scratchy". Foote (1997) was among the first to use *Mel-frequency cepstral coefficients* (MFCCs) in the music domain. He built a music indexing system using histograms of MFCC features, which were derived from a discriminatively trained vector quantizer. He used Euclidean distance and cosine distance for measuring similarity between the histograms. Blum *et al.* (1999) also built a music indexing system that incorporated various audio features such as loudness, bass, pitch, and MFCCs. Welsh *et al.* (1999) built a system for searching songs that sound similar to a given query song. They used 1248 feature dimensions per song which modeled the tonal content, the noise and volume levels, and the tempo and rhythm of a song.

The research on music content similarity in the early 2000s used MFCCs extensively and focused on timbral similarity. Both Logan and Salomon (2001) and Aucouturier and Patchet (2002) modeled songs by clustering MFCC features and determined their similarity by comparing the models. Logan and Salomon used K-means clustering and Earth Mover's Distance while Aucouturier and Patchet used *Gaussian Mixture Models* (GMMs), which were initialized with K-means clustering and trained with the Expectation-maximization algorithm. For comparing the models, Aucouturier and Pachet used Monte Carlo sampling to approximate the likelihood of the MFCCs of one song given the model of another. Later, Aucouturier and Patchet (2004) attempted improving content-based music similarity by fine-tuning the parameters of their algorithm. They also tried hidden Markov models in place of GMMs, but saw no improvement. Their results suggested that there exists a "glass ceiling" at 65-70% accuracy for timbral similarity. Berenzweig *et al.* (2003) mapped MFCCs into an anchor space using pattern classifiers and modeled the distributions using GMMs. They compared the models using an approximation of Kullback-Leibler (KL) divergence called Asymptotic Likelihood Approximation as well as Euclidean distance after reducing the distributions to the centroids.

However, the research did not focus solely on MFCCs and timbral features as rhytmic and tonal features could also be extracted from the audio and then combined with the timbral features for better results. Tzanetakis and Cook (2002) extracted several timbral, rhythmic, and pitch features to determine similarity and classify songs into genres. Li and Ogihara (2004) combined various timbral features with Daubechies wavelet filter histograms to determine similarity and also to detect emotion. Pampalk *et al.* (2005) proposed combining fluctuation patterns with the spectral descriptors, such as MFCCs. Ellis (2007) combined timbral features with beat-synchronized chroma features, which represent the harmonic and melodic content. The features were used to identify artists through classification. Some researchers, such as Gomez (2006b), extracted high-level tonal features such as chords and the key of the song.

More recent research on feature extraction has used neural networks to automatically learn features from music audio. Hamel and Eck (2010) used a *deep belief network* (DBN) to learn features, which they used as inputs for a non-linear *support vector machine* SVM. The learned features outperformed MFCCs in genre classification and in an autotagging task. Schmidt and Kim (2011) used a DBN to learn emotion-based features from audio content. The emotions were modeled in the arousal-valence representation of human emotions, where valence indicates positive vs. negative emotions and arousal indicates emotional intensity. Henaff *et al.* (2011) used a sparse coding method called Predictive Sparse Decomposition to learn sparse features from audio data. The features are used as inputs for a linear SVM that is used to predict genres for songs. Van den Oord *et al.* (2013) used deep convolutional neural networks to predict latent factors from music audio for use in music recommendation. Wang and Wang (2014) use a DBN to learn audio features from audio content and combine the learned features with collaborative filtering in a hybrid recommender.

The use of additional features improved the classification accuracy but to further improve the accuracy, researchers have used other classifiers instead of the GMMs that were popular in the early 2000s. Mandel and Ellis (2005) and Xu *et al.* (2003) used SVMs for classifying songs instead of GMMs. Mandel and Ellis used Mahalanobis distance and KL divergence to measure similarity. Xu *et al.* classified individual frames of songs and let the frames vote for the class of the entire song. Mandel and Ellis also discovered the "album effect" in which the classifiers perform significantly better when the songs from the same albums are used to train and test the classifier. The effect is due to timbral similarity recognizing songs from the same album easily.

Some researchers have turned to machine learning to automatically learn a similarity metric for audio content. Slaney *et al.* (2008) used principal component analysis whitening, linear discriminant analysis, relevant component analysis, neighborhood component analysis, and large-margin nearest neighbor on web page co-occurrence data to learn distance metrics, which they tested using a K-nearest nearest neighbor classifier. McFee *et al.* (2012) used metric learning to rank algorithm to learn a content-based similarity measure from collaborative filter data.

Research has also been made on context-based similarity, which uses contextual information to infer similarity between artists and songs. Pachet *et al.* (2001) mined the web and used co-occurrence data to determine similarity among songs. Whitman and Lawrence (2002) queried web search engines for pages related to artists and used the unstructured text to extract a feature space, which they used to predict a list of similar artist based on term overlap and the TF-IDF score. Baumann and Hummel (2003) also used unstructured text data fetched from the web to extract feature spaces and similarity matrices. The difference to the approach of Whitman and Lawrence was the use of filtering to filter out unrelated pages. Schedl *et al.* (2005b) also used co-occurrences to determine similarity but used specific queries for search engines to address the problem of finding unrelated results. Pohle *et al.* (2007) used the TF-IDF approach to analyze the top 100 web pages for each artist and then decomposed the data into base "concepts" using non-negative matrix factorization. The artists were classified based on the concepts with K-nearest neighbors and the classes were used to provide artists recommendations.

Context-based similarity has also been combined with content-based similarity to improve classification accuracy. Knees *et al.* (2007; 2009) combined web page based song ranking with content-based similarity to improve the quality of the results in a music search engine. Turnbull *et al.* (2009) combined social tags and web documents with timbre and harmony based automatic song tagging to improve text-based music retrieval.

The research on music recommendation has focused mainly on content-based filtering as an extension for content-based similarity and feature extraction as well as hybrid recommenders. This can be explained by the fact that recommenders using collaborative filtering outperform purely content-based recommenders (Barrington, Oda, & Lanckriet, 2009; Celma & Herrera, 2008; Slaney, 2011). Content-based filtering is also a compelling research topic as it solves some of the issues collaborative filtering has.

However, there has not been much research on purely content-based recommenders. Cano *et al.* (2005) presented a purely content-based recommender system called MusicSurfer. It automatically extracted descriptors for instrumentation, rhythm and harmony from audio signals. Logan (2004) proposed four solutions for recommending music from song sets, which consisted of songs representative of the sound the user is seeking. The goal was to improve recommendation accuracy by including more audio data from multiple songs. However, the song sets Logan used were taken from the same album and track from the same album was used as an objective criterion for the evaluation, which meant that the real performance was overestimated due to the ”album effect” discovered by Mandel and Ellis (2005).

Recommenders using only collaborative filtering have not been a very popular research topic in the music domain. Most research on collaborative filtering in the music domain has often been related to other recommendation techniques and research that focused only on collaborative filtering applied to recommendation systems for all kinds of media. In recent years, however, research on recommenders using collaborative filtering has gained a more popularity in the music domain.

The first music recommender system using collaborative filtering was Ringo (Shardanand & Maes, 1995), which had a 7-point rating scale and the value 4 was fixed as a neutral value r_z. It used a constrained Pearson correlation for calculating similarity, which correlates absolute like/dislike rather than the relative deviation. This was made possible by the absolute reference value r_z. The recommendations were based on the gathered rating data. In contrast, Cohen and Fan (2000) crawled user logs associated with a large repository of digital music and conducted web searches for lists of someone's favorite artists. They used collaborative filtering methods on the data to form recommendations. Chen and Chen (2005) used content-based and collaborative filtering approaches separately to recommend music based on music and user groups. The music groups contained songs the user was recently interested in and user groups combined users with similar interests. They achieved higher accuracy with the content-based approach, but the collaborative filtering approach provided more surprising recommendations. Sánchez-Moreno *et al.* (2016) proposed a collaborative filtering method that used listening coefficients as a way to address the gray sheep issue of collaborative filtering. To identify the gray sheep users, the listening coefficients and users' behavior regarding artists they listen to were used to characterize the users based on the uncommonness of their preferences. The proposed method significantly outperformed more traditional collaborative filtering methods.

The approach by Chen and Chen (2005) was close to a hybrid recommender as they used two recommendation techniques although they did not combine the two. In contrast, the approach of Stenzel and Kamps (2005) was essentially a hybrid recommender as it combined content-based filtering with collaborative filtering by using support vector machines to predict feature vectors for collaborative filtering from sound features. They combined the vectors with implicit user profile data to build a collaborative model and used item-item collaborative filtering to provide recommendations. Yoshii *et al.* (2006) presented another hybrid recommender, in which they combined collaborative and content-based filtering. The motivation was to solve the problems with both techniques. They associated rating data and the MFCCs of songs with latent variables that describe unobservable user preferences and used a Bayesian network called three-way aspect model to provide recommendations from them.

In contrast, more recent approaches to hybrid recommenders have used contextual information to improve recommendations. Donaldson (2007) used co-occurrence data from user created playlists and several acoustic features in a hybrid recommender. The co-occurrence data was decomposed to eigenvectors that comprised a set of spectral item-to-item graph features. The resulting spectral graph was unified with the acoustic feature vectors and then used for generating recommendations. Chedrawy and Abidi (2009) combined collaborative filtering with ontology-based semantic matching in a web recommender system. The recommendations were based on similarity that was a linearly weighted hybrid of collaborative filtering and item-based semantic similarity. Bu *et al.* (2010) modeled social media information collected from Last.fm and acoustic features extracted from the audio signal as a hypergraph, which allows expressing multiple dimensions of similarity simultaneously. The hypergraph was used to compute a hybrid distance upon which recommendations were based.

In recent years, the research direction for music recommender systems has moved towards user-centric recommenders. The information retrieval community has recognized that accuracy metrics are not enough to evaluate recommenders as they do not measure many aspects of the recommendation process that are important to the end user (McNee, Riedl, & Konstan, 2006; Ge, Delgado-Battenfeld, & Jannach, 2010). These aspects include serendipity (McNee et al., 2006; Ge et al., 2010), novelty (Celma & Herrera, 2008), and coverage (Ge et al., 2010) among others. In addition, Schedl *et al.* (2013) argued that the multifaceted and subjectiveness of music perception has been largely neglected and should be given more attention. They identified that modeling user needs is a key requirement for user-centric music retrieval systems. To improve the evaluation of recommenders, Celma and Herrera (2008) presented item- and user-centric methods for evaluating the quality of novel recommendations. The item-centric method analyzes the item-based recommendation network to detect pathology that hinders novel recommendations in the network topology. The aim of the user-centric method is to measure users' perceived quality of novel recommendations.

One of the earliest studies in more user-focused direction in the music domain was by Hoashi *et al.* (2003). They presented a content-based music retrieval system, which retrieves songs based on the musical preferences of the user. In addition, they proposed a method to generate user profiles from genre preferences with later refinement based on relevance feedback in order to reduce the burden of users to input learning data to the system. Another early study on modeling user preferences was by Grimaldi and Cunningham (2004). They attempted to predict user taste by extending the use of signal approximation and characterization from genre classification to the problem. They only achieved moderate success as the predictors had a mean accuracy of about 4% better than random guessing. In a more recent study, Bogdanov *et al.* (2013) presented a method for modeling users by inferring high-level semantic descriptors for each music track in a set provided by the user as an example of his or her preferences. They compared the recommendation accuracy of their method to two metadata and two content-based baselines. Their high-level semantic description recommender outperformed the content-based baselines as well as one of the metadata baselines, which randomly selected tracks from the same genre. The other metadata baseline, which used recommendations provided by Last.fm based on 20 tracks from the user's preference set, outperformed the semantic recommender. The authors suggested that Last.fm was able to provide better recommendations due to its larger dataset.

As an alternative method, Vignoli and Pauws (2005), Sotiropoulos *et al.* (2007), and Lu and Tseng (2009) all presented metrics adapted to a user's perception of similarity. The metric of Vignoli and Pauws was based on a user-weighted combination of timbre, genre, tempo, year, and mood. In their user test, they found that users preferred the adjustable system to control systems as it allowed more control and was perceived to be more useful. The system of Sotiropoulos *et al.* constructed music similarity perception models of its users by associating different similarity measures based on objective acoustic features to different users. They used relevance feedback and neural network-based incremental learning to determine the subset of the objective features that more accurately approximated a specific user's subjective music similarity perception. Lu and Tseng combined content-based, collaborative filtering, and emotion-based recommendation in a hybrid system that asked users for feedback on the provided recommendations. The initial recommendations were provided after mining users' listening records. The system altered the weights for the recommendation methods based on the user feedback and provided consecutive recommendations.

Some studies looked into improving the overlooked factors of recommenders such as novelty and serendipity. Nakatsuji *et al.* (2010) presented a collaborative filtering method for increasing the novelty of recommendations. Their method first measures user similarity based on rated items and a taxonomy of items and then creates a graph, in which related users are connected with edges that are weighted based on user similarity. The method then extracts related users that do not have high similarity to the active user as they are likely to have novel items for the active user. Their evaluation using multiple datasets showed that their method was able to provide more novel recommendations than other methods. Zhang *et al.* (2012) presented a recommender called *Auralist*, which has a special focus on serendipity. It used *latent dirichlet allocation* (LDA) for computing item features, which they called Artist-based LDA. The increase of novelty, diversity, and serendipity in recommendations was achieved with two algorithms: Listener Diversity and Declustering. The first prioritizes recommending artists with particularly diverse listener communities, which encourages users to explore beyond a given niche. The latter determines clusters of artists that user listens to and recommends artists outside of those clusters. Schedl and Hauger (2015) proposed user features that modeled diversity, mainstreaminess, and novelty of user's music taste in order to include more user-specific characteristics into music recommendation. They evaluated the features using multiple standalone and hybrid recommendation approaches and discovered that grouping users according to the proposed features and performing recommendations within the groups outperforms working on the entire user set.

In addition to the user-centric recommenders, context-aware recommenders have also become a popular research topic. Context-awareness describes the computer's ability to sense and act upon information about its environment. This information includes location, time, temperature, and user identity. Lee and Lee (2007), Cunningham *et al.* (2008), and Herrera *et al.* (2010) all highlighted the usefulness of context-awareness in music recommenders. Cunningham *et al.* hypothesized that contextual factors probably correlated with listener's preferences and Lee and Lee as well as Herrera *et al.* noticed an improvement in recommendation accuracy when they considered contextual factors when making recommendations.

Recent research in context-awareness has focused mainly on either temporal context or the location of the user. Cebrián *et al.* (2010) presented a recommender using collaborative filtering that took into account the temporal context of the user. The system built temporally-constrained micro-profiles that were used to generate recommendations most appropriate for the considered time period. Dias and Fonseca (2013) included temporal user listening patterns into a session-based collaborative filtering recommender. They compared two techniques, one explicit and one implicit, to capture the listening patterns and discovered that inclusion of the temporal information significantly improved the accuracy of the recommendations regardless whether the information was captured explicitly or implicitly. Baltrunas *et al.* (2011) presented a context-aware recommender called *InCarMusic* for recommending music when traveling in a car. The system allowed the user to configure contextual factors, such as mood, traffic conditions, and weather. These factors were then used to generate recommendations specific to the current context. Kaminskas *et al.* (2013) also presented a location-aware hybrid music recommender that used places of interest (POIs), automatically tagged music tracks, and knowledge of the semantic relations between the POIs and music tracks to form recommendations. They evaluated the recommender by comparing it to a system using only the POIs and tags and to a system using only the semantic knowledge. They conducted a web-based user study that had the users pick which recommender produced better recommendations that suit a specific POI. They discovered that the hybrid recommender performed better than the other recommenders.

Besides the temporal and location-aware systems, there have also been recent studies about more general context-awareness. Wang *et al.* (2012) presented a probabilistic model that integrates contextual information collected from mobile devices with audio content analysis in a recommender to satisfy users' short-term music playing needs. Users were able to manually choose the activity they were currently doing or have the phone automatically infer the activity based on the sensor data of the phone, such as acceleration and ambient noise. The system was able to provide good recommendations even without pre-existing ratings or annotations, which they attributed to the system's context-awareness. Vigliensoni and Fujinaga (2016) introduced a dataset containing listening histories of users as well as their demographic information and features that characterize aspects of their listening behavior. They discovered that by including demographic features and a profiling feature termed exploratoryness, the accuracy increased 12 percent compared to only using the listening history data.

4 Music recommendation methods

There are four recommendation methods that are used in music recommender systems: collaborative filtering, context-based filtering, content-based filtering, and hybrid methods, which combine the other filtering methods and minimize the issues a single method can have. This chapter describes how the different recommendation methods work and what kind of limitations the methods have.

4.1 Collaborative filtering

Collaborative filtering (CF) predicts user preferences for items based on the ratings or behavior of others users in the system. It is based on the assumption that other users' opinions can be selected and aggregated in a way that makes it possible to provide a reasonable prediction of the active user's preference. The user gives implicit or explicit feedback to the recommender system and the system recommends new items by inferring similarities between items and comparing them in terms of the people who use them. (Celma, 2010, p. 23; Ekstrand, Riedl, & Konstan, 2011, p. 88; McFee et al., 2012)

The collaborative filtering term was coined by the *Tapestry* project at Xerox PARC and it was the first system to implement the method (Goldberg, Nichols, Oki, & Terry, 1992). Collaborative filtering has become the most successful method for a wide variety of recommendation tasks including music, books, and movies. In fact, several studies have shown that CF systems consistently outperform alternative methods for recommending music (Barrington et al., 2009; Slaney, 2011).

Celma (2010, p. 70) mentions that the early research on CF methods in the music domain was based on explicit feedback, which was based on the ratings about songs or artists. However, this has shifted to implicit feedback as tracking users' listening habits has become the most common feedback collection method. The use of implicit feedback has the drawback that interaction between user and items is usually described by the songs they listen to or the total playcounts instead of a value in a predefined range (e.g., [1, 5] or *like it/hate it*). Jawaheer *et al.* (2010) state that explicit feedback can be positive or negative while implicit feedback is only positive. However, explicit feedback tends to concentrate on either extreme of the rating scale as users are more likely to express their preference if they feel strongly about an item. Additionally, Jawaheer *et al.* note that implicit feedback can be mapped to infer more detailed degree of preference, e.g., a user who has listened to track A 10 times and track B 100 times has a higher preference for track B than A.

As explained by Celma (2010, p. 23), CF methods work by building a matrix M, with n items and m users, that contains the interaction (e.g., rating, play) of the users with the items. Each row represents a user and the columns represent items. Ekstrand *et al.* (2011, p. 101) mention that this type of representation has extremely high dimensions as an item is a m-dimensional vector and a user is a n-dimensional vector. Additionally, there is redundancy in the dimensions as users and items can both be divided into groups with similar preferences. Celma (2010, p. 26) and Ekstrand *et al.* (2011, p. 102) both note that matrix factorization techniques such as singular value decomposition, non-negative matrix factorization, or principal component analysis can be used to reduce the matrix M to k dimensions or latent factors.

Predicting a rating value for an item i can be done in two different ways. The first method, called *item-item collaborative filtering*, uses similarities between the rating patterns of items. Items are considered similar if they tend to have the similar users like and dislike them as users are expected to have similar preferences for similar items. (Ekstrand et al., 2011, p. 96)

Item-item CF generates predictions by computing the similarity between a target item i and a set of items that a user u has rated (Celma, 2010, p. 24). The similarity between two items i and j can be calculated using cosine similarity, Pearson correlation, or by computing the conditional probability, $P\left(j|i\right)$ (Ekstrand et al., 2011, p. 99). Sarwar *et al.* (2001) note that cosine similarity does not account for the differences in rating scale between different users. To offset the drawback, they present the adjusted cosine similarity (Eq. 1), which subtracts the average rating of each user from each co-rated pair:

$$sim(i,j) = \frac{\sum_{u \in U}\left(r_{u,i} - \bar{r}_u\right)\left(r_{u,j} - \bar{r}_u\right)}{\sqrt{\sum_{u \in U}\left(r_{u,i} - \bar{r}_u\right)^2}\sqrt{\sum_{u \in U}\left(r_{u,j} - \bar{r}_u\right)^2}}, \tag{1}$$

where $\bar{r}_u$ is the average rating of the u-th user.

The similarity scores can be used to calculate the predicted rating, $\hat{r}_{u,i}$, using a weighted average. Common way to calculate it is

$$\hat{r}_{u,i} = \frac{\sum_{j \in S^k(i;u)} sim(i,j) r_{u,j}}{\sum_{j \in S^k(i;u)} sim(i,j)}, \tag{2}$$

where $S^k(i;u)$ is the set of k neighbors of item i that the user u has rated (Sarwar et al., 2001). The predicted value is based on the weighted sum of the user's ratings for all items in $S^k(i;u)$, and it captures how the user rates items that are similar to i. (Celma, 2010, p. 25)

The second method, which is called *user-user collaborative filtering* or *k-NN collaborative filtering*, computes the predicted rating by searching for other users who are similar to user u and using their ratings on other items for computing the predictions (Ekstrand et al., 2011, p. 91). The predicted rating, $\hat{r}_{u,i}$, is typically calculated as the weighted average of the neighboring users' ratings i using similarity as the weights:

$$\hat{r}_{u,i} = \bar{r}_u + \frac{\sum_{v \in S^k(u)} sim(u,v)\,(r_{v,i} - \bar{r}_v)}{\sum_{v \in S^k(u)} sim(u,v)}, \tag{3}$$

where $S^k(u)$ is the set of k neighbors for user u, and $\bar{r}_u$ is the average rating for user u (Celma, 2010, p. 25; Ekstrand et al., 2011, p. 91). The user similarity can be computed using several different similarity functions such as Pearson correlation, Spearman rank correlation, and cosine similarity (Ekstrand et al., 2011, pp. 93-94).

Collaborative filtering can be further divided into three categories:

- **Memory-based CF** generates a prediction by using the entire user-item database. Every user is in a group of people with similar interests and by identifying the neighbors of a user, a prediction of preferences on new items for the user can be computed as an aggregate of the neighbors' ratings. Item-based and user-based CF are in this category. (Adomavicius & Tuzhilin, 2005; Su & Khoshgoftaar, 2009; Shi, Larson, & Hanjalic, 2014)

- **Model-based CF** uses machine learning and data mining algorithms to train and model the users' preferences based on the collection of ratings. The model is then used to make predictions for test and real-world data. (Adomavicius & Tuzhilin, 2005; Su & Khoshgoftaar, 2009)

- **Hybrid CF** makes predictions by combining memory-based and model-based CF algorithms, or by combining CF with other recommendation techniques. (Su & Khoshgoftaar, 2009)

Despite its popularity, collaborative filtering has several drawbacks, which affect the quality of recommendations:

- **Cold start problem**. An item with no ratings cannot be recommended until it has been rated by users. Early recommendations of the item will often also be inaccurate due to there being few ratings on which to base the recommendations. The problem also applies to new users that enter the system as they only have few ratings and are thus more difficult to categorize and will likely receive poor recommendations. This problem is also known as the *early rater problem*. (Celma, 2010, p. 27; Claypool et al., 1999)

- **Gray sheep**. Users with atypical tastes would rarely, if ever, receive accurate recommendations, even after the initial start up phase for the user and system. This is due to their opinions not consistently agreeing or disagreeing with any group of people. (Claypool et al., 1999)

- **Sparsity problem**. Due to the large number of items, the matrices containing the users' ratings for the items are very sparse. It is common to have a sparse user-item matrix with coverage of 1% or less. The sparsity can make finding reliable neighbors difficult. (Celma, 2010, p. 26; Claypool et al., 1999)

- **Popularity bias**. Popular items in the dataset are similar to many items and it increases the probability that the system recommends the popular items instead of less popular items that could be more interesting and novel for the users. (Celma, 2010, p. 27)

- **Feedback loop**. Previous social interaction, e.g., ratings, with the system affects the user behavior that feedbacks into the system, which creates a loop. For example, listening to a certain genre will lead to recommendations from that same genre and listening to the recommendations will in turn lead to more recommendations from the genre. (Celma, 2010, p. 27; Cosley, Lam, Albert, Konstan, & Riedl, 2003)

- **Shilling attacks**. In situations where anyone can provide ratings, i.e., explicit feedback, people are able to give lots of positive ratings for their own creations and negative ratings for their competitors. The manipulated ratings can affect users to rate towards the manipulated predictions. (Lam & Riedl, 2004)

4.2 Context-based filtering

Context-based filtering uses cultural information to describe and characterize items, which is then used to compute artist or song similarity (Celma, 2010, p. 30). The cultural information can include metadata such as genre, emotions, semantic features, similarity, labels, and tags (Celma, 2010, p. 56). The filtering is based on data gathered with web mining techniques or data mined from collaborative tagging (Celma, 2010, p. 73).

Web mining techniques aim to discover interesting and useful information by analyzing web content and its usage. Kosala and Blockeel (2000) identify three web mining categories. *Web content mining* includes text, hypertext, semi-structured data, and multimedia mining. *Web structure mining* analyzes the link structures of the Web. It is based on the topology of the hyperlinks. *Web usage mining* analyzes session logs and user interactions. Celma (2010, p. 30) notes that information gained through Web content mining can be used to derive item similarity. Information gained through usage mining can be used derive user habits and preferences as well as item similarity based on co-occurrences in the session logs.

A recommender system combines all three mining techniques to derive similarity among items, e.g., items that co-occur in the same pages, and to model the users based on their interaction with the content. If the information about the content is in textual form, classic information retrieval measures can be used to characterize the items. (Celma, 2010, p. 30)

In music information retrieval a popular method to gather artist related terms is to query a general search engine with the name of artist and extract unigrams, bigrams and noun phrases from the retrieved pages (Celma, 2010, p. 57). Alternative method is to analyze public playlists on the web and compute song or artist co-occurrence from the data, which can then be used to compute artist or song similarity (Celma, 2010, p. 58).

Collaborative tagging (also known as social tagging) is the annotation of web content using tags, which are freely chosen keywords. A bottom-up classification emerges when all the annotations from the users are grouped. Recommender systems can use the collaborative tagging data to derive item or user similarity. (Celma, 2010, p. 31)

In collaborative tagging the users' tags for items can be represented as tuples of $\langle user, item, tag \rangle$, which conform to a 3-dimensional matrix or a *tensor*, a multi-dimensional matrix. The two main approaches for using the collaborative tagging information to compute item and user similarity are unfolding the 3-dimensional tensor in three bi-dimensional matrices (user-tag, item-tag, and user-item), and directly using the 3-dimensional tensor. (Celma, 2010, p. 32)

In the unfolding approach, the user-tag matrix U contains the number of times user i has applied the tag j, $U_{i,j}$. A recommender system can derive a user profile, e.g., a user-specific tag cloud, from the U matrix, which can be used to compute user similarity. The item-tag matrix I contains the number of times an item i has been tagged with tag j, $I_{i,j}$. The matrix I contains the contextual descriptions of the items based on the tags that have been applied to the items by the users. The item-tag matrix I can be used to compute item as well as user similarity with some additional information such as the top-N artists in Last.fm, a website based on collaborative tagging that also tracks listening habits. The user-item matrix R denotes whether a user i has tagged the item j, $R_{i,j}$. Collaborative filtering techniques can be used to compute item or user similarity from the matrix R. (Celma, 2010, p. 32)

One of the main drawbacks of the web mining techniques is that the queries for artist names such as "Hammock", "Low", or "Can" will give results completely unrelated to the artists (Schedl et al., 2005b). Schedl *et al.* (2005a) partially solved the problem by pursuing the $TF \times IDF$ (term frequency $\times$ inverse document frequency) approach and penalizing terms that appear in many documents, i.e., the terms with high document frequency.

Another drawback for the web mining techniques is the high dimensionality of the datasets (Celma, 2010, p. 58). This problem has been avoided by Pohle *et al.* (2007) by using non-negative matrix factorization to reduce the dimensionality of the artist-term matrix and by using a predefined vocabulary of music terms.

According to Celma (2010, p. 34), one of the main limitations for collaborative tagging is coverage despite popular items having been tagged by several users while lesser known items often do not have enough tags to characterize them. Another drawback stems from the lack of a constrained vocabulary for the tags. This leads to problems with polysemy ("I *love* this song" versus the song being about *love*), synonymy (*shoegaze*, *shoegazer*, and *shoegazing*), and personal tags, such as *seen live*, that are not very useful for deriving similarity. Tag sparsity is also present as some tags are widely used in the music domain (e.g., *blues* and *electronic*) and some tags describing the music are rarely applied, e.g., *dulcimer*. This leads to a biased distribution of the terms. Another big drawback with collaborative tagging is that it can be vandalized by deliberately mistagging some items that negatively affect the quality of recommendations of a system using the tag information.

4.3 Content-based filtering

Content-based filtering collects information describing the item content and recommend items that are similar to items the user likes. The items are usually represented by n-dimensional feature vectors and the features they contain can be collected automatically, e.g., by extracting features from the audio signal, or use manual annotations by domain experts, i.e., musicians. (Celma, 2010, p. 28; Kaminskas & Ricci, 2012; Song, Dixon, & Pearce, 2012)

Content-based recommendation relies heavily on audio content analysis and thus content-based music recommenders exploit traditional music information techniques such as automatic genre detection (Kaminskas & Ricci, 2012). Through these techniques, acoustic and musical features are extracted from the audio signal. The features can be divided into three categories: timbral, temporal, and tonal (Schedl, Knees, McFee, Bogdanov, & Kaminskas, 2015, p. 458). The acoustic features can also be further used to infer or predict semantic annotations by using machine learning techniques. The extraction of some of these features is detailed in Chapter 5.

Studies have often combined timbral features with other acoustic features to improve recommendation accuracy. Pampalk *et al.* (2005) combined timbral features with fluctuation patterns that describe loudness fluctuations in 20 frequency bands. The characteristics described by the fluctuation patterns are not described by the timbral features. Cano *et al.* (2005) used temporal and tonal features (tempo, meter, rhythm patterns, tonal strength, key, mode) in addition to timbral features in their music recommender. Maillet *et al.* (2009) combined timbral features with two temporal song-level features: danceability and long-term loudness level. The features were used to learn a similarity model and automatically generate tag clouds to steer a recommendation engine to generate personalized playlists. McFee and Lanckriet (2009) combined timbral features with chroma features and automatically inferred semantic tags (genre, mood) to reproduce human-derived measurements of subjective similarity between artists. Bogdanov *et al.* (2013) inferred semantic descriptors from the low-level audio features and compared recommenders using various types of features. Subjective evaluations they obtained through a user study showed that approaches based on only timbre information have only below average or average user satisfaction. The semantic descriptors outperformed the purely timbre information based approaches, which suggested that pure timbre-based approaches are insufficient compared to approaches that combine a large set of timbral, temporal, and tonal features. Thus, combining multiple features is beneficial but requires additional computation and also increases the size of the feature vectors.

Content-based recommender systems must first compute the similarity among songs before they can recommend music to the user (Celma & Serra, 2008, p. 75). As stated by Bogdanov *et al.* (2011), there exists a wide variety of approaches for measuring the similarity between songs. The approaches comprise both the selection of audio descriptors and an appropriate distance function. The common approaches can include a variety of perceptually relevant descriptors related to different musical aspects. However, such descriptors are generally low-level and not directly related to semantic explanations that users would easily understand (Celma & Serra, 2008). Schedl *et al.* (2015) note that timbral similarity is the most common similarity that can be used. It compares the spectral shapes of the tracks and is thus very basic. Probability distributions of frame-wise Mel-frequency cepstral coefficients (see Subsection 5.1.1) can be used to represent timbre information, which can be then compared using various similarity measures, some of which are detailed in Chapter 6.

In the music domain, content-based filtering ranks songs based on how similar they are to a seed song according to some similarity measure, which focuses on an objective distance between items and does not include any subjective factors. This makes it possible to recommend new items that do not have any user ratings associated with them. Additionally, there is no popularity bias as all items are considered to be of equal importance because user-generated data is not used to measure similarity. (Celma, 2010, p. 29, 75; Kaminskas & Ricci, 2012)

Content-based filtering solves some of the drawbacks of collaborative filtering, but it has several of its own:

- **Cold start problem**. Recommender systems that use user preferences are susceptible to the cold start problem as the system needs time to adapt to new users' preferences. However, the cold start problem does not exist for new items. (Celma, 2010, p. 29; Kaminskas & Ricci, 2012)
- **Gray sheep**. Content-based recommendation relies heavily on the size of the collection and what kind of items are in the collection. The collection may be biased towards a specific genre, i.e., overfitted. Thus, it is possible that users with atypical tastes will not receive relevant recommendations. (Celma, 2010, p. 29)
- **Novelty**. Users can receive recommendations that are too similar when the similarity function is accurate. Content-based recommender systems should promote eclecticness of items by using other factors. (Celma, 2010, p. 29; Kaminskas & Ricci, 2012)
- **Feature limitation.** Content-based recommender system are limited by the features that can be extracted from the content. In other words, the recommender system is limited by the descriptive data that is available. (Celma, 2010, p. 29; Kaminskas & Ricci, 2012)
- **Modeling user preferences**. Content similarity cannot fully capture the user's preferences, which results in a semantic gap between the user's perception of music and the music representation of the system. (Kaminskas & Ricci, 2012)

4.4 Hybrid methods

Hybrid methods combine other filtering techniques to achieve better recommendations. Most commonly, collaborative filtering is combined with the other techniques. By combining different filtering methods, the system minimizes the issues an individual method can have. (Celma, 2010, p. 34, 78)

Burke (2002) defines some combination methods that have been used by researchers to combine filtering techniques:

- **Weighted** hybrid recommender combines the results of all available recommendation techniques to compute a score for a recommended item. The benefits of the weighted hybrid are that all capabilities of the system are relevant to the recommendation process in a straightforward way and performing post-hoc credit assignment and adjusting the hybrid is easy.

- **Switching** hybrid recommender uses some criterion to switch between recommendation techniques. For example, a system using both collaborative and content-based filtering could attempt collaborative filtering if content-based filtering provided recommendations with insufficient confidence. The benefit of a switching hybrid is that it can be sensitive to the strengths and weaknesses of individual recommenders, but it comes with additional complexity in the recommendation process as the switching criteria has to be determined.

- **Mixed** hybrid recommender presents recommendations from multiple techniques simultaneously. In some systems, the recommendations can be combined together in the final suggested item. The benefit of a mixed hybrid is that it avoids the cold start problem for new items as a content-based filtering can be relied on items that have not been rated by anyone.

- **Feature combination** is a way to merge content-based and collaborative filtering techniques. The collaborative information is treated as an additional feature data and content-based techniques are used over the augmented data set. This method lets the system consider collaborative data without exclusively relying on it, which reduces the system's sensitivity to the number of users who have rated an item.

- **Cascade** hybrid recommender involves a staged process, in which one recommendation technique is employed first to produce a coarse ranking of candidates and a second technique is then used to refine the recommendation from the candidates. This method allows the system to avoid employing the lower-priority technique on items that are well-differentiated by the first technique or that will never recommended to sufficiently poor rating.

- **Feature augmentation** method first employs one technique to produce a rating or classification for an item, which is then incorporated into the processing of the next recommendation technique. It differs from the cascade method by using the output of the first technique in the features used by the second technique.

- **Meta-level** hybrid uses the model generated by one technique as the input for another. The difference to feature augmentation is that meta-level hybrid uses the entire model as an input instead of simply generating features for the second technique based on a learned model.

5 Audio content features

There are multiple acoustic features that can be automatically extracted from the audio signal. Most of these features are calculated by first segmenting the audio signal into overlapping frames, with the duration ranging from 10 to 100 ms and a 50% overlap. Each frame is windowed using a window function, e.g., Hann window, and then transformed using *fast Fourier transform* (FFT), which computes the *discrete Fourier transform* (DFT). This process is also known as *short-time Fourier transform* (STFT). Next, a feature vector is calculated for each frame and the features are then summarized by their means and variances across all frames.

This chapter details the computation of several low-level features. For rhythmic and tonal features, the low-level features that are used for computing the higher-level features are detailed.

5.1 Low-level features

Low-level features have little meaning to users but they are the basis for high-level analyses as they are easily exploited by computer systems. They are usually related to loudness and timbre, which has been found to be related to three main music signal properties. The properties are the spectral envelope shape, temporal evolution of energy, and time variation of the spectrum. (Schedl, Gómez, & Urbano, 2014, p. 150)

Low-level features are often the basis for representing timbre in higher-level features such as rhythmic features (Schedl et al., 2014, p. 153). As such, the low-level features are often called timbral features (Bogdanov et al., 2011).

5.1.1 Mel frequency cepstral coefficients

Mel frequency cepstral coefficients (MFCCs) were originally developed for automatic speech recognition and were later found to be useful for music information retrieval (Pampalk, 2006, p. 17). MFCCs are a representation of the timbre of the audio. The original implementation of the MFCC, known as MFCC FB-20, was introduced by Davis and Mermelstein (1980).

The mel frequency in MFCCs is a value on the mel scale (Figure 3), which approximates the frequency resolution of the auditory system by a mapping between

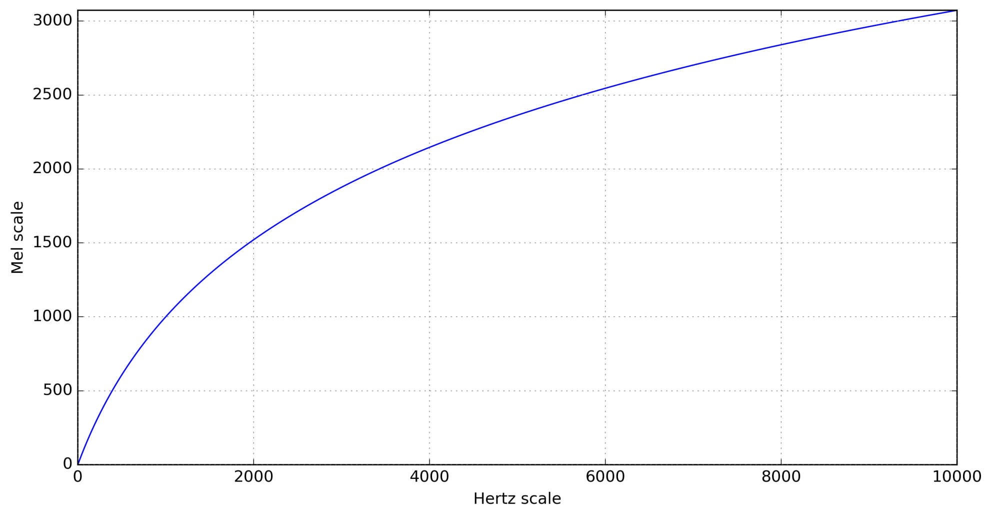


Figure 3. Mel scale

the actual frequency and the perceived pitch (Stevens, Volkmann, & Newman, 1937). The scale is approximately linear for low frequencies (< 500 Hz) and logarithmic for higher frequencies (Pampalk, 2006, p. 18). Stevens *et al.* (1937) defined the reference point to the linear frequency scale as a 1000 Hz tone which is defined as 1000 Mel. A tone that is twice as high is 2000 Mel and a tone that is half as high is 500 Mel. According to Ganchev *et al.* (2005), the widely used formulae in the various implementations of the MFCCs are

$$\hat{f}_{mel} = 2595 \log_{10}(1 + f_{\text{Hz}}/700) \tag{4}$$

and

$$\hat{f}_{mel} = 1127 \ln(1 + f_{\text{Hz}}/700). \tag{5}$$

The mel scale formulae are used to convert the frequency in Hz to mels. The inverse formulae are

$$\hat{f}_{mel}^{-1} = 700 \left(10^{\hat{f}_{mel}/2595} - 1\right) \tag{6}$$

for Equation 4, and

$$\hat{f}_{mel}^{-1} = 700 \left[\exp\left(\frac{\hat{f}_{mel}}{1127} \right) - 1 \right] \tag{7}$$

for Equation 5.

The first step in the MFCC computation is to transform the magnitude spectrum obtained through DFT to the mel scale using a filter bank consisting of triangular filters (Figure 4). Each triangular filter defines the response of one frequency band and is normalized so that the sum of the weights for each triangle is the same. (Pampalk, 2006)

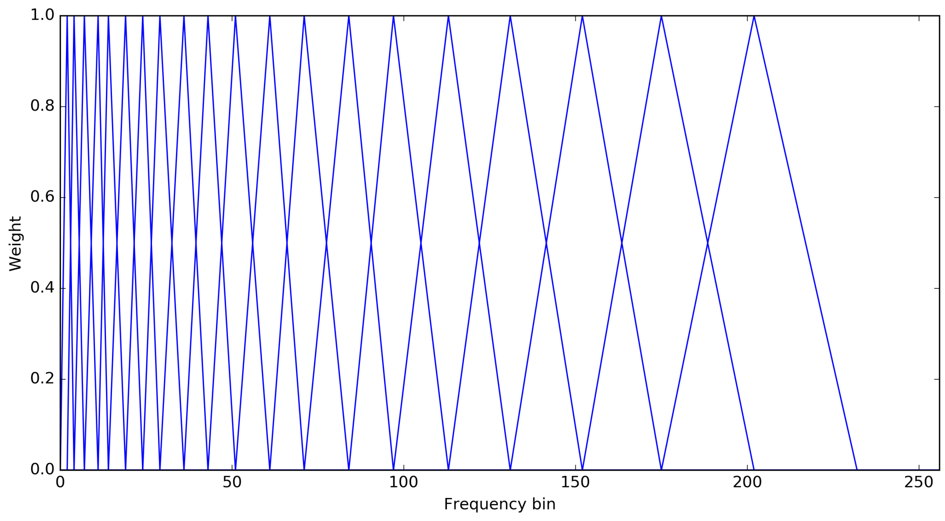


Figure 4. 20 triangular filters with 50% overlap

In the MFCC FB-20 implementation, a filter bank $H_i(k)$ is consists of M equal height triangular filters, each of which is defined as

$$H_i(k) = \begin{cases} 0 & \text{for } k < f_{b_{i-1}} \\ \dfrac{k - f_{b_{i-1}}}{f_{b_i} - f_{b_{i-1}}} & \text{for } f_{b_{i-1}} \le k \le f_{b_i} \\ \dfrac{f_{b_{i+1}} - k}{f_{b_{i+1}} - f_{b_i}} & \text{for } f_{b_i} \le k \le f_{b_{i+1}} \\ 0 & \text{for } k > f_{b_{i+1}} \end{cases}, \quad i = 1, 2, \ldots, M, \tag{8}$$

where i stands for the i-th filter, f_{b_i} are the boundary points of the filter, and corresponds to the k-th coefficient of the N-point DFT. The positions for boundary points f_{b_i} depend on the sampling frequency F_s and the number of points N in the DFT as follows

$$f_{b_i} = \left(\frac{N}{F_s}\right) \hat{f}_{mel}^{-1} \left(\hat{f}_{mel}(f_{low}) + i\frac{\hat{f}_{mel}(f_{high}) - \hat{f}_{mel}(f_{low})}{M+1}\right), \tag{9}$$

where f_{low} and f_{high} are the low and high boundary frequencies in Hz for the whole filter bank, and M is the number of filters. (Ganchev et al., 2005)

In Davis and Mermelstein's (1980) MFCC implementation, the MFCC parameters are computed as

$$C_j = \sum_{i=1}^{M} X_i \cos\left[j\left(i - \frac{1}{2}\right)\frac{\pi}{M}\right], \text{ with } j = 1, 2, \ldots, J, \tag{10}$$

where M is the number of filters in the filter bank, J is the number of computed cepstral coefficients (usually $J < M$), and X_i is the "log-energy output of the i-th filter", which is

$$X_i = \log_{10}\left(\sum_{k=1}^{N} |X(k)| H_i(k)\right), \quad i = 1, 2, \ldots, M, \tag{11}$$

where $X(k)$ is the magnitude spectrum of the Fourier transform.

5.1.2 Spectral features

Spectral features are related to the spectral shape of the audio signal. They are computed in the spectral domain.

Spectral centroid measures the average frequency of the spectrum weighted by the magnitude. It is the center of gravity of the magnitude spectrum of the STFT. It is a measure of the spectral shape and higher values correspond to "brighter" textures with higher frequencies. (Celma, 2010, p. 64; Tzanetakis & Cook, 2002)

Tzanetakis and Cook (2002) give the formula as

$$\text{Centroid}(X) = \frac{\sum_{n=1}^{N} nX(n)}{\sum_{n=1}^{N} X(n)}, \tag{12}$$

where $X(n)$ is the magnitude of the Fourier transform at frequency bin n, X is a DFT frame, and N is the number of frequency bins, i.e., half the number of samples in a DFT frame.

Spectral flatness is the ratio between the geometrical mean and the arithmetical mean of the spectrum magnitude (Celma, 2010, p. 64). It relates to the distinction between more noise-like and more tone-like sound (Allamanche et al., 2001). The formal definition is

$$\text{Flatness}(X) = \frac{\sqrt[N]{\prod_{n=1}^{N} M_t(n)}}{\frac{1}{N}\sum_{n=1}^{N} M_t(n)}. \tag{13}$$

Spectral skewness is the third order central moment and it gives indication about the shape of the spectrum by describing the degree of asymmetry of the distribution. When the skewness is equal to 0, it indicates that the distribution is symmetric. Values less than 0 and values greater than 0 indicate more energy on the right and on the left of the distribution respectively. (Celma, 2010, p. 64; Peeters, 2004)

The formula for spectral skewness is

$$\text{Skewness}(X) = \frac{\frac{1}{N}\sum_{n=1}^{N}(X(n)-\mu_t)^3}{\left[\frac{1}{N-1}\sum_{n=1}^{N}(X(n)-\mu_t)^2\right]^{3/2}}, \tag{14}$$

where μ_t is the arithmetic mean for frame t.

Spectral kurtosis is the fourth order central moment and it indicates whether the distribution is peaked or flat relative to a normal distribution. When the kurtosis is equal to 3, the distribution is a normal distribution. Values less than 3 indicate a flatter distribution and values greater than 3 indicate a peaker distribution. (Celma, 2010, p. 64; Peeters, 2004)

The formula for spectral kurtosis is

$$\text{Kurtosis}(X) = \frac{\frac{1}{N}\sum_{n=1}^{N}(X(n)-\mu_t)^4}{\left(\frac{1}{N}\sum_{n=1}^{N}(X(n)-\mu_t)^2\right)^2} - 3. \tag{15}$$

5.2 Rhythmic features

As described by Gouyon (2005, p. 15), the main goal of automatic rhythm description is turning acoustic events that occur in time into more abstract notions of tempo, timing, and metrical structure. Tempo is the pace of a musical piece, i.e., how fast or slow it is. In a metrical structure, tempo is the rate of beats at a given metrical level. Timing describes when the acoustic events occur. Metrical structure describes the regular temporal structure underlying musical event occurrences. More accurately, the metrical structure deals with durationless points in time, i.e., the beats.

The computation of rhythmic features is based on measuring periodicity of events, which are represented by onsets or low-level features such as energy or spectral features (Schedl et al., 2014, p. 165). Specifically, onsets are used to used estimate beat positions as well as beat loudness when combined with spectral energy.

There are multiple algorithms for onset detection. One of the earliest was presented by Masri and Bateman (1996) and it was based on *high frequency content* (HFC), which characterizes the high-frequency content in the signal. It is computed as

$$HFC(X) = \sum_{n=1}^{N} n|X(n)|^2. \tag{16}$$

The actual detection was done as

$$\frac{HFC(X_i)}{HFC(X_{i-1})} \cdot \frac{HFC(X_i)}{E(X_i)} > T_D, \tag{17}$$

where subscript i denotes the current frame, T_D is the detection threshold, and E_i is the spectral energy, which is computed as

$$E(X) = \sum_{n=1}^{N} |X(n)|^2. \tag{18}$$

Bello *et al.* (2004) presented an onset detection function called complex spectral difference, which combines energy-based and phase-based onset detection and detects the onset by computing the spectral difference between successive frames in the complex domain. The two detection methods are complementary as energy-based approaches favor strong percussive onsets and phase-based approaches emphasize soft, tonal onsets.

Following the simpler description of the function of Bello *et al.* given by Davies and Plumbley (2007), the function makes two assumptions. Its first assumption is that the magnitude spectrum remains approximately constant during steady-state regions of the signal, i.e., not at a note onset. This allows making a prediction of the magnitude spectrum $\hat{M}_i(n)$ at frame i given the magnitude of the previous frame

$$\hat{M}_i(n) = M_{i-1}(n) = |X_{i-1}(n)|. \tag{19}$$

The second assumption is that phase velocity of the nth STFT bin should ideally be constant during steady-state regions of the signal, that is,

$$\tilde{\varphi}_i(n) - \tilde{\varphi}_{i-1}(n) \approx \tilde{\varphi}_{i-1}(n) - \tilde{\varphi}_{i-2}(n), \tag{20}$$

where $\tilde{\varphi}_i(n)$ is the unwrapped phase of the Fourier transform at frame i and frequency bin n.

By adopting a short-hand notation for the left-hand side of Equation 20

$$\Delta\tilde{\varphi}_i(n) = \tilde{\varphi}_i(n) - \tilde{\varphi}_{i-1}(n) \tag{21}$$

and substituting the short-hand into Equation 20 and rearranging the terms, a prediction $\hat{\varphi}_i(n)$ can be made about the phase of the nth bin for frame i given the observations of the previous two frames as

$$\hat{\varphi}_i(n) = \text{princarg}\left[\tilde{\varphi}_{i-1}(n) + \Delta\tilde{\varphi}_{i-1}(n)\right], \tag{22}$$

where the function princarg unwraps the phase value and maps it into the range $[-\pi, \pi]$.

The predictions of the magnitude spectrum, $\hat{M}_i(n)$, and the phase spectrum, $\hat{\varphi}_i(n)$, are then converted into polar form to give a spectral prediction, $\hat{S}_i(n)$, in the complex domain

$$\hat{S}_i(n) = \hat{M}_i(n) \exp\left(j\hat{\varphi}_i(n)\right), \tag{23}$$

which is compared to the observed complex spectrum

$$S_i(n) = M_i(n) \exp\left(j\varphi_i(n)\right) \tag{24}$$

using Euclidean distance. The complex spectral difference, Γ_i, at frame i is then computed as the sum of the distances between the predicted and observed spectra for all n bins

$$\Gamma_i = \sum_{n=1}^{N} |S_i(n) - \hat{S}_i(n)|^2. \tag{25}$$

The complex spectral difference onset detection function has been relatively popular and has been used by many researchers (e.g., Davies and Plumbley (2007) and Degara *et al.* (2012)) in beat tracking systems.

5.3 Tonal features

Tonality is one of the main aspects of Western music and as such, it is important to have methods for extracting the tonal content of a song. In Western tonal music, *key* is a system of relationships between a series of pitches having a central pitch, or a *tonic*, as its most important element. In addition to the tonic, a key has two other important pitches: the *dominant* degree, which is the fifth degree of the scale; and the *subdominant* degree, which is the fourth degree of the scale. The subdominant degree lies below the tonic and dominant lies above it. (Gómez, 2006b)

The two basic key *modes*, major and minor, have different musical characteristics regarding the position of tones and semitones within their respective scales. A *scale* is composed of a sequence of notes and each two notes form an interval that is defined by a ratio between two note frequencies f_1 and f_2. A semitone is defined by a frequency ratio of $f_2/f_2 = 2^{1/12}$ for an equal-tempered scale. An interval of n semitones is defined by a frequency ratio of $f_2/f_1 = 2^{n/12}$. Thus, the amount of semitones n in an interval between two frequencies f_1 and f_2 is defined by $n = 12\log_2(f_2/f_1)$. (Gómez, 2006b)

When considering an equal-tempered scale and enharmonic equivalence, there exists 24 keys in total as each tonic manages both a major and a minor mode. This means that there are 12 equally distributed semitones within an octave. The 12 semitones play a major role when computing tonal features. (Gómez, 2006b)

Gómez (2006b) presented the *harmonic pitch class profile* (HPCP), which is a 12-dimensional vector representing the intensities of the 12 semitone pitch classes of an equally-tempered chromatic scale. It is based on the pitch class profile proposed by Fujishima (1999). The differences to PCP are the use of only those spectral peaks whose frequency is within the interval [100, 5000] Hz, the introduction of a weight into the feature computation, and the use of a higher resolution in frequency bin, which is achieved by decreasing the quantization level to less than a semitone. The pitch class intensities are sometimes called chroma features and the HPCP is then called the chroma feature vector (Schedl et al., 2014, pp. 159-160).

As described by Gómez (2006b), the first step in the HPCP computation is the STFT, which calculated using a frame size of 4096 samples, hop size of 512 samples, and a Blackman-Harris 62 dB window (Harris, 1978, pp. 64-65). Next, the spectral peaks, i.e., the local maxima of the magnitude spectrum, are determined. Finally, the HPCP values are then computed as

$$\text{HPCP}(n) = \sum_{i=1}^{nPeaks} w(n, f_i) a_1^2, \tag{26}$$

where n is the HPCP bin, a_i is the linear magnitude of the i-th peak, f_i is the frequency value of the i-th peak, $nPeaks$ is the number of considered spectral peaks, and $w(n, f_i)$ is the weight of the frequency f_i when considering the HPCP bin n.

The weight depends on the frequency distance between f_i and the center frequency of the bin n, f_n, measured in semitones. The center frequency of the n-th bin is

$$f_n = f_{ref} 2^{n/size}, \tag{27}$$

where $size$ is the number of bins in the HPCP vector (either 12, 24, or 36), and f_{ref}

is the reference frequency, which is defined as

$$f_{ref} = 440 \cdot 2^{df/12}, \tag{28}$$

where *df* is the detuning factor, which can be estimated by analyzing the deviation of the spectral peaks with respect to the standard reference frequency of 440 Hz (Gómez, 2006a, p. 74). The distance in semitones between the peak frequency f_i and the center frequency f_n is defined as

$$d = 12 \log_2 \left(\frac{f_i}{f_n} \right) + 12m, \tag{29}$$

where m is the integer that minimizes the magnitude of the distance $|d|$. Finally, the weight is computed as

$$w(n, f_i) = \begin{cases} \cos^2 \left(\frac{\pi}{2} \frac{d}{0.5l} \right) & \text{if } |d| \leq 0.5l \\ 0 & \text{if } |d| > 0.5l \end{cases}, \tag{30}$$

where l is the length of the weighting window, which is a parameter of the algorithm and has been empirically set to 4/3 semitones.

After the HPCP values have been computed, they are normalized for each analysis frame. This is done with respect to the maximum value of the entire vector to store the relative relevance of each HPCP bin. Thus, the normalization is computed as

$$\text{HPCP}_{normalized}(n) = \frac{\text{HPCP}(n)}{max(\text{HPCP})}. \tag{31}$$

To estimate the key, the correlation of the HPCP vector with a matrix of HPCP profiles corresponding to different keys K is computed. K consists of two matrices, one for each key mode, of size $12 \cdot size$. The correlation value $R(i, j)$ is computed for each key note as

$$R(i, j) = r(\text{HPCP}, K(i, j)) = \frac{E[(\text{HPCP} - \mu_{\text{HPCP}}) \cdot (K(i, j) - \mu_{K(i,j)})]}{\sigma_{\text{HPCP}} \cdot \sigma_{K(i,j)}}, \tag{32}$$

where $K(i, j)$ is the key profile, $i = 1, 2$ where 1 represents the major profile and 2 the minor profile, $j = 1, \ldots, 12$ for the 12 possible key notes, and E is expected value operator. For details on the construction of the key profile matrix based on the model proposed by Krumhansl and Schmuckler (1990) see (Gómez, 2006b). The maximum correlation value $R(i_{\max}, j_{\max}) = \max_{i,j}(R(i, j))$ corresponds to the estimated key note and mode, and it is used as a measure of the tonalness, the degree of tonality or key strength.

The HPCP can also be used as an input for other algorithms to estimate the chords and chord progression of a piece of music.

6 Audio-based similarity measures

Various similarity measures have been used to compare audio features extracted from the audio signal. This chapter introduces some of the typical measures for timbral similarity.

6.1 K-means clustering with Earth Mover's Distance

K-means clustering with Earth Mover's Distance (EMD) was originally proposed by Logan and Salomon (2001). The first step is to cluster the target song's feature vector using K-means clustering. The set of clusters is characterized by the mean, covariance, and weight of each cluster, respectively. Logan and Salomon denote the set of clusters as the signature of song.

The obtained song signatures are then compared using EMD (Rubner, Tomasi, & Guibas, 2000), which calculates the minimum amount of work needed to transform one distribution into the other. Let $P = \{(\mu_{p_1}, \Sigma_{p_1}, w_{p_1}), \ldots, (\mu_{p_m}, \Sigma_{p_m}, w_{p_m})\}$ be a song signature with m clusters where μ_{p_i}, Σ_{p_i}, and w_{p_i} are the mean, covariance, and weight of cluster p_i. Let $Q = \{(\mu_{q_1}, \Sigma_{q_1}, w_{q_1}), \ldots, (\mu_{q_n}, \Sigma_{q_n}, w_{q_n})\}$ be another song signature. Let $d_{p_i q_j}$ be the distance between clusters p_i and q_j, which is computed using a symmetric form of Kullback-Leibler divergence. For clusters p_i and q_j this takes the form

$$d_{p_i q_j} = \frac{\Sigma_{p_i}}{\Sigma_{q_j}} + \frac{\Sigma_{q_j}}{\Sigma_{p_i}} + \left(\mu_{p_i} - \mu_{q_j}\right)^2 \cdot \left(\frac{1}{\Sigma_{p_i}} + \frac{1}{\Sigma_{q_j}}\right). \tag{33}$$

Define $f_{p_i q_j}$ as the flow between p_i and q_j. It reflects the cost of moving probability mass from one cluster to the other. Next, all $f_{p_i q_j}$'s are solved so that the overall cost W is minimized. W is defined as

$$W = \sum_{i=1}^{m} \sum_{j=1}^{n} d_{p_i q_j} f_{p_i q_j} \tag{34}$$

and is subject to a series of constraints. These constraints define the problem of seeking the cheapest way to transform signature P to signature Q, which can be formulated as a linear programming task and solved efficiently. Once all $f_{p_i q_j}$ have been solved, the EMD is calculated as

$$EMD(P,Q) = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} d_{p_i q_j} f_{p_i q_j}}{\sum_{i=1}^{m} \sum_{j=1}^{n} f_{p_i q_j}}. \tag{35}$$

Magno and Sable (2008) present a slightly different version of this measure. In their version, each cluster is fit with a Gaussian component to form a *Gaussian mixture model* (GMM). Magno and Sable, as well as many other authors (e.g., Celma (2010, p. 76) and Mandel and Ellis (2005)), falsely say that Logan and Salomon used K-means to learn the GMM parameters when they in fact used the K-means clusters to model the songs and did not use GMMs at all.

6.2 Gaussian Mixture Models with Monte Carlo sampling

Aucouturier and Pachet (2002) introduced the use of GMMs to model songs. Their approach consists of two steps, of which the first is to cluster the MFCC feature vectors using a GMM. A GMM estimates probability density as the weighted sum of M simpler Gaussian densities, which are often called components. Aucouturier and Pachet give the formula for a GMM as

$$p(F_t) = \sum_{m=1}^{M} C_m \mathcal{N}\left(F_t, \mu_m, \Sigma_m\right), \tag{36}$$

where F_t is the feature vector observed at time t, $\mathcal{N}$ is a multivariate Gaussian probability density function with mean vector μ_m, covariance matrix Σ_m, and C_m is a mixture coefficient. Pampalk (2006) gives the formula for $\mathcal{N}$ as

$$\mathcal{N}\left(x, \mu, \Sigma\right) = \left(\frac{1}{2\pi}\right)^{n/2} |\Sigma|^{-1/2} \exp\left(-\frac{1}{2}\left(x-\mu\right)^T \Sigma^{-1}\left(x-\mu\right)\right), \tag{37}$$

where n is the dimension of the feature vector.

Aucouturier and Pachet used three Gaussian distributions per mixture, but in a later evaluation (2004) discovered that using more Gaussian components provided better results. They obtained the best results using 20 MFCCs and 50 Gaussian components. The implementation of this method that won the ISMIR 2004 genre classification contest used 30 Gaussian components (Pampalk, 2004).

As described by Pampalk (2006), the parameters of a GMM, $\Theta = \{\mu_m, \Sigma_m, C_m | m = 1..M\}$, have to be estimated per GMM, i.e., per song. The optimal estimate for the parameters maximizes the likelihood that the feature vector $F_t = \{x_1, \ldots, x_n\}$ was generated by the GMM. Log-likelihood is the commonly used measure to compute the likelihood and it is computed as

$$L(F_t|\Theta) = \log \prod_n p(x_n|\Theta) = \sum_n \log p(x_n|\Theta). \tag{38}$$

Good estimates for the parameters can be found using the Expectation-Maximization (EM) algorithm although the initial estimates can be random, or computed using other clustering algorithms. Aucouturier and Pachet used K-means to compute the initial parameters.

The EM algorithm has two steps. The first step is to compute the expectation, which is the probability that an observation x_n was generated by the m-th component. The formula for the expectation step is

$$P(m|x_n, \Theta) = \frac{p(x_n|m, \Theta)C_m}{p(x_n)} = \frac{\mathcal{N}(x_n, \mu_m, \Sigma_m)\, C_m}{\sum_{m'=1}^{M} \mathcal{N}(x_n, \mu_{m'}, \Sigma_{m'})\, C_{m'}}. \tag{39}$$

The second step of EM is the maximization of the expectation, in which the parameters are recomputed based on the expectations. The formula for the maximization step is

$$\begin{aligned} \mu_m^* &= \frac{\sum_n P(m|x_n, \Theta)x_n}{\sum_{n'} P(m|x_{n'}, \Theta)} \\ \Sigma_m^* &= \frac{\sum_n P(m|x_n, \Theta)(x_n - \mu_m)(x_n - \mu_m)^T}{\sum_{n'} P(m|x_{n'}, \Theta)} \\ C_m^* &= \frac{1}{N} \sum_n P(m|x_n, \Theta). \end{aligned} \tag{40}$$

After the GMMs have been formed, the second step of the approach is to compute the similarity between the cluster models, i.e., songs. This is done by drawing a sample from GMMs representing songs A and B. Let these samples be X^A and X^B. Aucouturier and Patchet mention that the sampling process roughly corresponds to recreating a song from its timbre model. The sampling process is known as Monte Carlo Sampling. The log-likelihood $L(X|\Theta)$ (Equation 38) is computed for each song/sample combination and the results are used to compute the distance. This is computed as

$$d_{AB} = L(X^A|\Theta^A) + L(X^B|\Theta^B) - L(X^A|\Theta^B) - L(X^B|\Theta^A). \tag{41}$$

Pampalk (2006) notes that $L(X^A|\Theta^B)$ and $L(X^B|\Theta^A)$ are generally different values. Both are used because a symmetric similarity measure is desirable for most applications. Furthermore, the self-similarity is added to normalize the results and in most cases the inequalities $L(X^A|\Theta^A) > L(X^A|\Theta^B)$ and $L(X^A|\Theta^A) > L(X^B|\Theta^A)$ hold true.

6.3 Average feature vectors with Euclidean distance

Tzanetakis and Cook (2002) presented a simple method in which the means and variances of spectral centroid, spectral rolloff, spectral flux, time domain zero crossings, low energy, and the first five MFCC coefficients were combined into a single 19-dimensional feature vector. In addition to the timbral features, they used six rhythmic content features and five pitch content features resulting in a 30-dimensional feature vector.

Tzanetakis and Cook trained a GMM for a genre classification tasks using the feature vectors. However, Magno and Sable (2008) used Euclidean distance to determine the similarity of the feature vectors as the vectors can be compared with standard distance measures. In another work, Tzanetakis (2002, p. 58) used Mahalanobis distance to compare the vectors citing different dynamic ranges of the features and their probable correlation as the reason for using it over Euclidean distance. In fact, Tzanetakis mentions that a naïve implementation of the similarity retrieval would use Euclidean distance.

6.4 Single Gaussian with Kullback-Leibler divergence

Mandel and Ellis (2005) presented a method using a single Gaussian with full covariance matrix to model a song. For comparing two Gaussian models, they used Kullback-Leibler (KL) divergence. For two distributions, $p(x)$ and $q(x)$, it is defined as

$$KL\left(p\|q\right) = \int p(x)\log\frac{p(x)}{q(x)}dx. \tag{42}$$

There exists a closed form of the KL divergence for single Gaussians and it is

$$KL\left(p\|q\right) = \frac{1}{2}\left[\log\left(\frac{|\Sigma_q|}{|\Sigma_p|}\right) + Tr(\Sigma_q^{-1}\Sigma_p) + (\mu_p - \mu_q)^T\Sigma_q^{-1}(\mu_p - \mu_q) - d\right], \tag{43}$$

where d is the dimensionality of x, and $Tr(M)$ is the trace of the matrix M.

As Mandel and Ellis used the KL divergence as a support vector machine kernel, it had to be modified to satisfy the Mercer conditions. The conditions are not fulfilled because KL divergence is neither symmetric nor positive definite. To symmetrize KL divergence, Mandel and Ellis summed the two divergences together

$$D_{KL}(p,q) = KL\left(p\|q\right) + KL\left(q\|p\right). \tag{44}$$

Fulfilling the other Mercer condition can be done by exponentiating the elements of this matrix as it will create a positive definite matrix. The final gram matrix has elements

$$K(X_i, X_j) = e^{-\gamma D_{KL}(X_i,X_j)}, \tag{45}$$

where γ is a parameter that can be tuned to maximize classification accuracy.

No closed form solution exists for KL divergence between two GMMs and it must be approximated instead. Bogdanov *et al.* (2009) compare songs X and Y modeled using GMMs containing a single Gaussian component with a closed form symmetric approximation of KL divergence as

$$\begin{aligned} d(X,Y) = {} & Tr(\Sigma_X^{-1}\Sigma_Y) + Tr(\Sigma_Y^{-1}\Sigma_X) + \\ & Tr((\Sigma_X^{-1} + \Sigma_Y^{-1})(\mu_X - \mu_Y)(\mu_X - \mu_Y)^T) - 2N_{MFCC}, \end{aligned} \tag{46}$$

where μ_X and μ_Y are MFCC means, Σ_X and Σ_Y are MFCC covariance matrices, and N_{MFCC} is the number of used MFCCs.

6.5 Euclidean distance based on Principal Component Analysis

Cano *et al.* (2005) used the Euclidean metric to compare feature vectors. Bogdanov *et al.* (2009) consulted Cano on the methodology used as the cited reference lacked specific details. Cano *et al.* normalized the feature data to interval [0, 1] and then used principal component analysis to reduce the dimension of the feature vectors to 25 variables. The comparisons were then done in the reduced feature space.

This similarity measure does not use MFCCs unlike all the other described measures. It uses an extensive set of features representing the timbral, rhythmic, and tonal aspects of the song. These include various spectral features such as spectral centroid, spread, and kurtosis. Rhythmic aspects are represented by beats loudness and the tonal aspects by untransposed harmonic pitch class profiles and key strength. (Bogdanov et al., 2011)

7 Content-based music recommender

This chapter describes a query-by-example content-based music recommender, which allows the user to input a song for analysis and recommends n most similar songs to the user. The recommender does not incorporate any user profiling and is based purely on objective similarity.

7.1 Dataset

The dataset that was used to train and evaluate the recommender was based on the 10,000-song subset of the *Million Song Dataset* (MSD) (Bertin-Mahieux, Ellis, Whitman, & Lamere, 2011). The songs in the subset have been selected randomly from the full dataset. The MSD dataset consists of pre-computed song features and metadata information related to the songs. The dataset does not contain any actual audio content as sharing songs would infringe on the artists' copyright, but pre-computed song features do not. However, as the method for computing the features is not well documented and we needed more control over the features as well as more features in general, we needed to get access to the audio content.

The recommended method for obtaining the audio is to fetch song previews from 7digital[1] and the dataset contains several 7digital identifier numbers related to each song, which makes it relatively easy to do so. By using the 7digital identifiers from the dataset as well as identifiers fetched by searching the store for the songs, we were able to fetch previews for 7143 songs. The length of each preview was either 30 or 60 seconds although few songs had previews whose duration was less than 30 seconds. The sampling rate for most songs was 44.1 kHz but some songs had a sampling rate of 22.05 kHz. Many songs were no longer available with the dataset's identifier as different versions, e.g., remastered, of the albums had replaced the original albums. Some albums or artists had been completely removed from the store and were not available.

To compute the audio features, an open-source C++ library called Essentia (Bogdanov, Wack, et al., 2013) was used. The library includes bindings for Python, which was the programming language of choice for implementing the feature extractor. The computed audio features are listed in Table 1. Some of the computed features were not used in similarity comparisons due to their inaccuracy, but were

[1]An online music store. https://www.7digital.com/

still used for computing the high-level features.

Instead of storing all values for the audio frames, several statistics were computed over the frames. These statistics included the minimum and maximum values, median, mean, variance, and the mean and variance of the first derivative and second derivative. For some features the values were simply summarized. The means of the coefficients, the covariance matrix and its inverse were computed for MFCCs and GFCCs.

The high-level features were computed using pre-trained classifier models from the lower-level representations of the songs. These models were provided separately by the authors of Essentia[2]. We used only the models with the highest accuracy.

Table 1: Computed audio features.

Feature group	Feature name
Low-level	Average loudness, dynamic complexity
	Spectral centroid, spread, kurtosis, rolloff, decrease, skewness
	Spectral crest, flatness, RMS, flux
	Spectral energy, energy band ratio, strong peak, complexity
	Spectral dissonance, entropy, contrast
	High frequency content
	Pitch salience
	Silence rate at 20, 30, and 60 dB*
	Zero crossing rate
	Mel bands, ERB bands, Bark bands
	MFCCs*, GFCCs*
	Kurtosis, skewness, spread, crest, and flatness for Mel, ERB and Bark bands
Rhythmic	Beat positions*
	Beat loudness, beat loudness band ratio
	BPM, BPM histogram*
	Onset rate
	Danceability
Tonal	Transposed and untransposed HPCP, key strength*
	Key*, scale*
	HPCP crest and entropy
	Tuning frequency*, diatonic strength*
	Chords key*, scale*, strength*
	Chord change rate*, chords histogram*
	Equal-tempered deviation*, non-tempered energy ratio*
High-level	Danceability (danceable/not danceable)
	Singer gender (male/female)
	Vocal/instrumental
	Mood (acoustic/not acoustic and aggressive/not aggressive)
	Timbre (bright/dark)
	Tonality (tonal/atonal)

*Not used in similarity comparisons.

[2]http://essentia.upf.edu/documentation/svm_models/

7.2 Implementation

The implemented recommender attempts to improve the recommendations by using more features than Cano *et al.* (2005) while using a similar similarity measure, which is described in Section 6.5. One noteworthy addition to the used features are the high-level features that Bogdanov *et al.* (2013) used to achieve better recommendations than methods based on only MFCCs. The high-level features were not included in the PCA and were appended to the reduced feature vectors consisting of 25 PCA components. With the high-level features, the feature vectors contained 40 elements per song.

The recommender uses *K-nearest neighbors* (KNN) as the classifier to easily obtain the top-N recommendations for a given query song. In this case, we settled for $N = 10$. We used Euclidean distance as the similarity metric to stay as close to the method used by Cano *et al.*

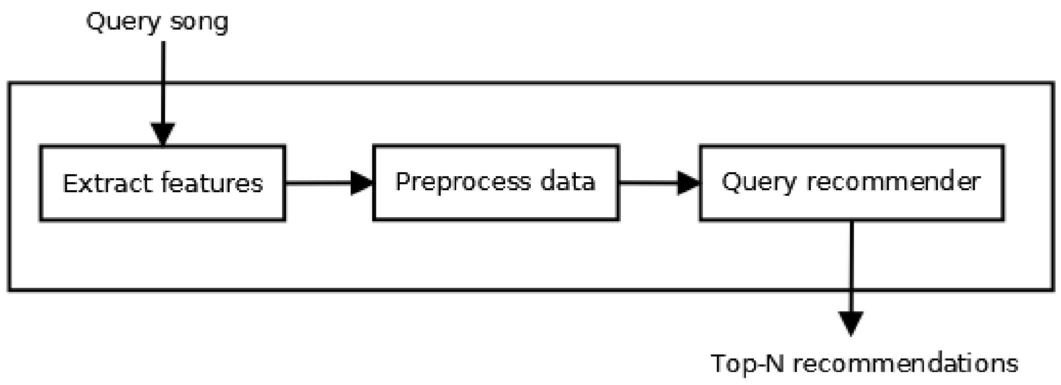


Figure 5. The pipeline for obtaining recommendations for a query song.

The pipeline for obtaining recommendations for a query song is detailed as a flowchart in Figure 5. In the feature extraction step, all features listed in Table 1 are extracted. After the extraction, the data is first normalized to range $[0, 1]$ and then the dimensions of the data are reduced using PCA. The final feature vector is then used as an input for the trained KNN model that outputs the top-N recommendations.

7.3 Evaluation

We evaluated the recommender using two different methods. The first evaluation method was more objective as it was based on the tags obtained from Last.fm. The second method was entirely subjective and based on the author's judgment on the similarity of the songs. In the first method, our recommender (REC-PCA+HI) was compared to randomly chosen recommendations as well as a recommender using the single Gaussian similarity measure described in Section 6.4, that is, a recommender using only MFCCs (REC-MFCC). In the second method, our recommender was compared to the MFCC-based recommender. We also evaluated the effectiveness of the high-level features by comparing our recommender to a recommender trained without the high-level features (REC-PCA).

7.3.1 Objective evaluation

The objective evaluation method split the dataset into a test set containing 100 randomly selected songs and a training set containing the rest of the songs. The recommender was trained using the training set and then queried using every song in the test set as the input.

The actual evaluation of the recommendation quality was based on the similar tags the input song and the recommended songs had. The top tags (genre, personal tags) were fetched from Last.fm for the input song and for each of the recommendations provided by the recommender. At most 10 tags were used per song. The tags were then compared and used to form a ratio of how many of the tags of the recommended song were similar. After the similar tag ratios had been discovered for all recommendations, the ratio of good recommendations was calculated. We considered every recommendation that had a similar tag ratio of 0.3, i.e., three similar tags in most cases, to be a good recommendation in the context of this evaluation. Once every song in the test set had been used as the query song, the mean of the good recommendation ratios was computed as the overall accuracy of the recommender.

However, this approach had some issues as some songs did not have any top tags or the particular song was not found in the Last.fm database. In these cases, the top tags for the artist were fetched. However, not all of the artists were found in the database either. In most cases these issues were due to the song having been performed by multiple artists. Due to inconsistent artist naming in the MSD, parsing only the relevant artist name would have required relatively complex rules. Additionally, certain song names consisted of multiple parts, e.g., "Concerto for Orchestra (Zoroastrian Riddles) (1996)/3. Adagio non troppo", and determining the name possibly used in Last.fm's database would have been non-trivial. Thus, when tags could not be found for a song, the song was simply skipped.

The evaluation described above was repeated 50 times after which the total accuracy for each recommender was computed. The results of the evaluation are presented in Table 2. The table lists the accuracy as a percentage for every evaluation as well as the total accuracy computed over all the evaluations.

The obtained results correspond to the results obtained by Bogdanov *et al.* (2011) in their similarity measure comparison, in which the single Gaussian recommender performed better than a recommender using the PCA method. In some evaluations, both REC-PCA and REC-PCA+HI provide better recommendations than REC-MFCC but overall they provide less accurate recommendations. All methods outperform the random selection.

The addition of the high-level features improves the quality of recommendations in most cases, which suggests that the high-level features are beneficial for music recommendation.

7.3.2 Subjective evaluation

The subjective evaluation method was based on listening to the song previews and deciding whether the songs sounded similar. As discussed previously in Section 2.2, the similarity of the songs is completely subjective and properly evaluating the recommender would require multiple subjects. This was not done due to time constraints. The accuracy of randomly selected recommendations was not evaluated subjectively as the objective evaluation showed that all recommenders outperformed the random baseline.

For the evaluation, 25 songs were randomly selected from the dataset. The songs were then used as queries for the recommenders and the recommended songs were compared to query song. The songs were given a similarity rating using the following ratings:

- **Not similar** songs did not sound like the song used as a query.
- **Somewhat similar** songs had some similarities to the query song, e.g., similar

Table 2: Results of objective evaluation.

Method	REC-PCA	REC-PCA+HI	REC-MFCC	Random
	6.733218	7.755456	6.570354	2.561873
	3.113426	3.206019	4.155093	2.292926
	5.824742	6.062428	9.650630	2.752556
	6.819967	7.742690	10.140351	2.266527
	4.783626	4.577277	5.298246	2.048767
	6.563239	6.315012	6.870567	2.336528
	6.296296	5.791246	7.732884	2.327734
	5.055556	4.792398	6.970760	2.115976
	4.905754	6.262401	6.795222	2.164036
	5.161565	5.609410	4.196226	2.810787
	5.598846	6.007295	6.105499	1.923309
	6.746032	6.907814	7.716728	2.143820
	5.314505	5.984043	7.044917	2.733811
	6.607757	7.364998	6.338570	2.600848
	5.107184	5.203322	5.693013	1.870962
	5.950000	6.390079	6.717460	2.646911
	6.895255	8.683449	7.743056	2.247353
	3.673469	4.481293	4.529478	1.753077
	7.972509	8.737113	8.582474	3.072995
	5.461988	6.192982	7.039265	2.537376
	4.736690	4.864005	4.884259	2.482440
	8.040828	7.644003	8.924480	2.369983
	7.701754	7.017544	7.915205	2.294046
	6.923977	6.461988	6.988304	2.151584
	9.207702	8.598485	8.860480	2.465699
	7.187135	7.267335	8.008772	2.449631
	7.047101	7.304607	8.230676	2.491446
	5.815789	5.359649	7.644528	2.266437
	3.451968	3.177083	4.415509	2.041062
	5.856307	5.646199	8.170426	2.224903
	7.884039	8.793210	10.435185	2.538937
	3.535880	3.399884	3.927745	2.040127
	5.685764	5.607639	8.046875	1.999770
	5.400585	5.701754	7.833333	1.854815
	5.269274	5.918367	6.493764	2.021732
	5.203322	5.005727	5.057274	2.406994
	5.525687	5.740741	5.430108	1.877152
	6.140046	6.513310	6.178902	2.323281
	6.529240	6.921053	8.801170	2.310103
	6.486111	6.288889	8.127778	2.390718
	3.315217	3.817719	5.469289	1.819739
	5.202295	5.763889	8.454106	2.364619
	5.238095	6.383220	7.593537	1.974333
	5.816151	6.592211	8.611111	2.089455
	3.176714	3.540189	2.612293	2.212621
	6.629439	6.620848	8.101375	2.301098
	8.073671	8.354469	8.546843	2.485782
	6.187642	6.502268	6.363379	2.225862
	6.022459	6.081560	5.789007	2.465544
	3.315603	4.045508	4.373522	2.419606
Average of total accuracy	5.823828	6.100002	6.923601	2.291354

feel or style.

- **Similar** songs sounded similar to the query song. Generally these songs had many similarities but might not have been good recommendations.

- **Very similar** songs sounded very similar to the query song and would have been excellent recommendations as a result unless they were from the same artist.

The results of the evaluation are presented in Table 3. REC-PCA+HI had the most *not similar* ratings as well as the most *similar* and *very similar* ratings, which indicates that while the recommendations were more often complete misses, the hits were of better quality. REC-PCA and REC-MFCC had nearly identical ratios of ratings.

All recommenders generally recommended songs that were similar to each other but sometimes not even remotely similar to query song. The recommenders also provided more accurate recommendations for query songs belonging to certain genres. Rap and metal songs generally received many recommendations that were at least *somewhat similar* to the query song. REC-PCA+HI worked especially well for metal songs.

Some genres were problematic for the other similarity but not for the other. REC-MFCC often recommended completely different genre for electronic music while REC-PCA and REC-PCA+HI provided relatively accurate recommendations. REC-PCA and REC-PCA+HI in turn had the same issue with Latin music, e.g., salsa, while REC-MFCC was able to provide some similar recommendations.

The album effect discovered by Mandel and Ellis (2005) was also noticeable as songs from the same album were usually in the top three recommendations. These obvious recommendations are not great as the user generally wants to find similar songs by other artists.

Table 3: Results of subjective evaluation.

	Not similar	Somewhat similar	Similar	Very similar
REC-PCA	0.765	0.163	0.056	0.016
REC-PCA+HI	0.772	0.136	0.068	0.024
REC-MFCC	0.764	0.164	0.056	0.016

8 Conclusions

Both objective and subjective evaluation show that music recommendation based solely on the audio content does not give accurate recommendations. This goes in line with the other research on content-based recommendation. Had the recommenders been compared to a collaborative filtering recommender, their ineffectiveness would have been even more apparent.

It is possible that better results would have been obtained with complete songs and larger dataset. The small size of the dataset naturally means that there are fewer songs that can be recommended. This affects the quality of the recommendations as there are fewer similar songs, which leads to the recommender recommending less similar songs.

The use of previews also has an effect on the recommendations as the preview might not be very representative of the entire song. For example, a preview of a rock song might only contain a softer or a quieter part of the song, which would lead to recommendations that completely lack the heavy sections that might be present in the full query song.

The selection of the extracted features for determining the similarity among songs seems to be very important as using more features does not seem to improve the recommendation quality. The recommender using only the MFCCs outperformed the recommenders using a wide range of features. It seems to be more important to select a subset of features that better represent the audio content and use a better similarity metric than a simple metric with many features. The use of high-level features in addition to the lower level features had a very minor but positive effect by improving recommendations slightly.

Nonetheless, researchers have discovered that content-based recommenders are not the solution to the recommendation problem on their own and they work better as a complementary recommendation technique when combined with other techniques. Content-based recommendation is especially useful cold-start situations when no collaborative filtering data is available as it makes it possible to provide at least somewhat accurate recommendations to the user.

Content-based recommendation can still improve in the future if certain features such as the key and chord progression of a song can be more accurately computed. Additionally, high-level semantic features inferred from the low-level features have been shown to improve the quality of recommendations (Bogdanov, Haro, et al., 2013).

References

Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. Knowl. Data Eng.*, *17*(6), 734–749.

Allamanche, E., Herre, J., Hellmuth, O., Fröba, B., Kastner, T., & Cremer, M. (2001). Content-based identification of audio material using MPEG-7 low level description. In *Proceedings of the 2nd International Symposium on Music Information Retrieval.*

Aucouturier, J., & Pachet, F. (2002). Music similarity measures: What's the use? In *Proceedings of the 3rd International Conference on Music Information Retrieval* (pp. 13–17).

Baltrunas, L., Kaminskas, M., Ludwig, B., Moling, O., Ricci, F., Aydin, A., ... Schwaiger, R. (2011). InCarMusic: Context-aware music recommendations in a car. In *Proceedings of the 12th International Conference on E-Commerce and Web Technologies* (pp. 89–100). Springer Berlin Heidelberg.

Barrington, L., Oda, R., & Lanckriet, G. R. G. (2009). Smarter than Genius? Human evaluation of music recommender systems. In *Proceedings of the 10th International Society for Music Information Retrieval Conference* (pp. 357–362).

Baumann, S., & Hummel, O. (2003). Using cultural metadata for artist recommendations. In *Proceedings of the 3rd International Conference on WEB Delivering of Music* (pp. 138–141).

Bello, J. P., Duxbury, C., Davies, M. E., & Sandler, M. B. (2004). On the use of phase and energy for musical onset detection in the complex domain. *IEEE Signal Processing Letters*, *11*(6), 553–556.

Berenzweig, A., Ellis, D. P. W., & Lawrence, S. (2003). Anchor space for classification and similarity measurement of music. In *Proceedings of the 2003 IEEE International Conference on Multimedia and Expo* (Vol. 1, pp. 29–32).

Berenzweig, A., Logan, B., Ellis, D. P. W., & Whitman, B. (2004). A large-scale evaluation of acoustic and subjective music-similarity measures. *Computer Music Journal*, *28*(2), 63–76.

Bertin-Mahieux, T., Ellis, D. P., Whitman, B., & Lamere, P. (2011). The Million Song Dataset. In *Proceedings of the 12th International Conference on Music*

Information Retrieval.

Blum, T., Keislar, D., Wheaton, J., & Wold, E. (1999). *Method and article of manufacture for content-based analysis, storage, retrieval, and segmentation of audio information.* (US Patent 5,918,223)

Bogdanov, D., Haro, M., Fuhrmann, F., Xambó, A., Gómez, E., & Herrera, P. (2013). Semantic audio content-based music recommendation and visualization based on user preference examples. *Information Processing & Management*, *49*(1), 13–33.

Bogdanov, D., Serrà, J., Wack, N., & Herrera, P. (2009). From low-level to high-level: Comparative study of music similarity measures. In *Proceedings of the 11th IEEE International Symposium on Multimedia* (pp. 453–458). IEEE Computer Society.

Bogdanov, D., Serrà, J., Wack, N., Herrera, P., & Serra, X. (2011). Unifying low-level and high-level music similarity measures. *IEEE Trans. Multimedia*, *13*(4), 687–701.

Bogdanov, D., Wack, N., Gómez, E., Gulati, S., Herrera, P., Mayor, O., … Serra, X. (2013). ESSENTIA: An audio analysis library for music information retrieval. In *Proceedings of the 14th International Society for Music Information Retrieval Conference* (pp. 493–498).

Bu, J., Tan, S., Chen, C., Wang, C., Wu, H., Zhang, L., & He, X. (2010). Music recommendation by unified hypergraph: Combining social media information and music content. In *Proceedings of the 18th ACM International Conference on Multimedia* (pp. 391–400). ACM.

Burke, R. D. (2002). Hybrid recommender systems: Survey and experiments. *User Model. User-Adapt. Interact.*, *12*(4), 331–370.

Cano, P., Koppenberger, M., & Wack, N. (2005). Content-based music audio recommendation. In *Proceedings of the 13th Annual ACM International Conference on Multimedia* (pp. 211–212). ACM.

Cebrián, T., Planagumà, M., Villegas, P., & Amatriain, X. (2010). Music recommendations with temporal context awareness. In *Proceedings of the 4th ACM Conference on Recommender Systems* (pp. 349–352). ACM.

Celma, Ò. (2010). *Music Recommendation and Discovery - The Long Tail, Long Fail, and Long Play in the Digital Music Space.* Springer.

Celma, Ò., & Herrera, P. (2008). A new approach to evaluating novel recommendations. In *Proceedings of the 2008 ACM Conference on Recommender Systems* (pp. 179–186). ACM.

Celma, Ò., & Serra, X. (2008). FOAFing the music: Bridging the semantic gap in music recommendation. *J. Web Sem.*, *6*(4), 250–256.

Chedrawy, Z., & Abidi, S. S. R. (2009). A web recommender system for recommending, predicting and personalizing music playlists. In *Proceedings of the 10th International Conference on Web Information Systems Engineering* (pp. 335–342). Springer-Verlag.

Chen, H., & Chen, A. L. P. (2005). A music recommendation system based on music and user grouping. *J. Intell. Inf. Syst.*, *24*(2-3), 113–132.

Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D., & Sartin, M. (1999). Combining content-based and collaborative filters in an online newspaper. In *Proceedings of ACM SIGIR Workshop on Recommender Systems.*

Cohen, W. W., & Fan, W. (2000). Web-collaborative filtering: Recommending music by crawling the web. *Computer Networks*, *33*(1-6), 685–698.

Cosley, D., Lam, S. K., Albert, I., Konstan, J. A., & Riedl, J. (2003). Is seeing believing?: How recommender system interfaces affect users' opinions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 585–592). ACM.

Cunningham, S., Caulder, S., & Grout, V. (2008). Saturday night or fever? Context-aware music playlists. In *Proceedings of the Audio Mostly Conference.*

Davies, M. E. P., & Plumbley, M. D. (2007). Context-dependent beat tracking of musical audio. *IEEE Trans. Audio, Speech & Lang. Proc.*, *15*(3), 1009–1020.

Davis, S., & Mermelstein, P. (1980). Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, *28*(4), 357–366.

Degara, N., Argones-Rúa, E., Pena, A., Torres-Guijarro, S., Davies, M. E. P., & Plumbley, M. D. (2012). Reliability-informed beat tracking of musical signals. *IEEE Trans. Audio, Speech & Lang. Proc.*, *20*(1), 290–301.

Dias, R., & Fonseca, M. J. (2013). Improving music recommendation in session-based collaborative filtering by using temporal context. In *Proceedings of the 2013 IEEE 25th International Conference on Tools with Artificial Intelligence* (pp. 783–788). IEEE Computer Society.

Donaldson, J. (2007). A hybrid social-acoustic recommendation system for popular music. In *Proceedings of the 2007 ACM Conference on Recommender Systems* (pp. 187–190). ACM.

Ekstrand, M. D., Riedl, J., & Konstan, J. A. (2011). Collaborative filtering recommender systems. *Foundations and Trends in Human-Computer Interaction*, *4*(2), 175–243.

Ellis, D. P. W. (2007). Classifying music audio with timbral and chroma features. In *Proceedings of the 8th International Conference on Music Information Retrieval* (pp. 339–340).

Foote, J. T. (1997). Content-based retrieval of music and audio. In *Proceedings of SPIE 3229, Multimedia Storage and Archiving Systems II* (pp. 138–147).

Fujishima, T. (1999). Realtime chord recognition of musical sound: A system using Common Lisp Music. In *Proceedings of the 1999 International Computer Music Conference.*

Ganchev, T., Fakotakis, N., & Kokkinakis, G. (2005). Comparative evaluation of various MFCC implementations on the speaker verification task. In *Proceedings of the SPECOM* (Vol. 1, pp. 191–194).

Ge, M., Delgado-Battenfeld, C., & Jannach, D. (2010). Beyond accuracy: Evaluating recommender systems by coverage and serendipity. In *Proceedings of the 4th ACM Conference on Recommender Systems* (pp. 257–260). ACM.

Goldberg, D., Nichols, D. A., Oki, B. M., & Terry, D. B. (1992). Using collaborative filtering to weave an information tapestry. *Comm. of the ACM*, *35*(12), 61–70.

Gómez, E. (2006a). *Tonal description of music audio signals* (PhD thesis). Universitat Pompeu Fabra, Barcelona, Spain.

Gómez, E. (2006b). Tonal description of polyphonic audio for music content processing. *INFORMS Journal on Computing*, *18*(3), 294–304.

Gouyon, F. (2005). *A computational approach to rhythm description — Audio features for the computation of rhythm periodicity functions and their use in tempo induction and music content processing* (PhD thesis). Universitat Pompeu Fabra, Barcelona, Spain.

Grimaldi, M., & Cunningham, P. (2004). Experimenting with music taste prediction by user profiling. In *Proceedings of the 6th ACM SIGMM International Workshop on Multimedia Information Retrieval* (pp. 173–180). ACM.

Hamel, P., & Eck, D. (2010). Learning features from music audio with deep belief networks. In *Proceedings of the 11th International Society for Music Informa-*

tion Retrieval Conference (pp. 339–344).

Harris, F. J. (1978). On the use of windows for harmonic analysis with the discrete Fourier transform. *Proceedings of the IEEE*, *66*(1), 51–83.

Henaff, M., Jarrett, K., Kavukcuoglu, K., & LeCun, Y. (2011). Unsupervised learning of sparse features for scalable audio classification. In *Proceedings of the 12th International Society for Music Information Retrieval Conference* (pp. 681–686).

Herlocker, J. L., Konstan, J. A., & Riedl, J. (2000). Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work* (pp. 241–250). ACM.

Herlocker, J. L., Konstan, J. A., Terveen, L. G., & Riedl, J. (2004). Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, *22*(1), 5–53.

Herrera, P., Resa, Z., & Sordo, M. (2010). Rocking around the clock eight days a week: An exploration of temporal patterns of music listening. In *Proceedings of the 1st Workshop on Music Recommendation and Discovery.*

Hoashi, K., Matsumoto, K., & Inoue, N. (2003). Personalization of user profiles for content-based music retrieval based on relevance feedback. In *Proceedings of the 11th ACM International Conference on Multimedia* (pp. 110–119). ACM.

Jawaheer, G., Szomszor, M., & Kostkova, P. (2010). Comparison of implicit and explicit feedback from an online music recommendation service. In *Proceedings of the 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems* (pp. 47–51). ACM.

Jennings, D. (2007). *Net, Blogs and Rock 'n' Roll: How Digital Discovery Works and What it Means for Consumers, Creators and Culture.* Nicholas Brealey Publishing.

Kaminskas, M., & Ricci, F. (2012). Contextual music information retrieval and recommendation: State of the art and challenges. *Computer Science Review*, *6*(2-3), 89–119.

Kaminskas, M., Ricci, F., & Schedl, M. (2013). Location-aware music recommendation using auto-tagging and hybrid matching. In *Proceedings of the 7th ACM Conference on Recommender Systems* (pp. 17–24). ACM.

Knees, P., Pohle, T., Schedl, M., Schnitzer, D., Seyerlehner, K., & Widmer, G. (2009). Augmenting text-based music retrieval with audio similarity: Advantages and limitations. In *Proceedings of the 10th International Society for Music Information Retrieval Conference* (pp. 579–584).

Knees, P., Pohle, T., Schedl, M., & Widmer, G. (2007). A music search engine built upon audio-based and web-based similarity measures. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 447–454). ACM.

Kosala, R., & Blockeel, H. (2000). Web mining research: A survey. *SIGKDD Explorations*, *2*(1), 1–15.

Krumhansl, C. L. (1990). *Cognitive foundations of musical pitch.* Oxford University Press.

Lam, S. K., & Riedl, J. (2004). Shilling recommender systems for fun and profit. In *Proceedings of the 13th International Conference on World Wide Web* (pp. 393–402). ACM.

Lee, J. S., & Lee, J. C. (2007). Context awareness by case-based reasoning in a music recommendation system. In *Proceedings of the 4th International Conference on Ubiquitous Computing Systems* (pp. 45–58). Springer-Verlag.

Lesaffre, M., Leman, M., & Martens, J. (2006). A user-oriented approach to music information retrieval. In *Content-Based Retrieval.*

Li, T., & Ogihara, M. (2004). Content-based music similarity search and emotion detection. In *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing* (Vol. 5, pp. 705–708).

Logan, B. (2004). Music recommendation from song sets. In *Proceedings of the 5th International Conference on Music Information Retrieval.*

Logan, B., & Salomon, A. (2001). A music similarity function based on signal analysis. In *Proceedings of the 2001 IEEE International Conference on Multimedia and Expo* (pp. 745–748).

Lu, C., & Tseng, V. S. (2009). A novel method for personalized music recommendation. *Expert Syst. Appl.*, *36*(6), 10035–10044.

Magno, T., & Sable, C. (2008). A comparison of signal based music recommendation to genre labels, collaborative filtering, musicological analysis, human recommendation and random baseline. In *Proceedings of the 9th International Conference on Music Information Retrieval* (pp. 161–166).

Maillet, F., Eck, D., Desjardins, G., & Lamere, P. (2009). Steerable playlist generation by learning song similarity from radio station playlists. In *Proceedings* *of the 10th International Society for Music Information Retrieval Conference* (pp. 345–350).

Mandel, M. I., & Ellis, D. (2005). Song-level features and support vector machines for music classification. In *Proceedings of the 6th International Conference on Music Information Retrieval* (pp. 594–599).

Masri, P., & Bateman, A. (1996). Improved modeling of attack transients in music analysis-resynthesis. In *Proceedings of the International Computer Music Conference* (pp. 100–103).

McFee, B., Barrington, L., & Lanckriet, G. (2012). Learning content similarity for music recommendation. *IEEE Trans. Audio, Speech, and Lang. Proc.*, *20*(8), 2207–2218.

McFee, B., & Lanckriet, G. R. G. (2009). Heterogeneous embedding for subjective artist similarity. In *Proceedings of the 10th International Society for Music Information Retrieval Conference.*

McNee, S. M., Riedl, J., & Konstan, J. A. (2006). Being accurate is not enough: How accuracy metrics have hurt recommender systems. In *Extended abstracts on human factors in computing systems* (pp. 1097–1101). ACM.

Nakatsuji, M., Fujiwara, Y., Tanaka, A., Uchiyama, T., Fujimura, K., & Ishida, T. (2010). Classical music for rock fans?: Novel recommendations for expanding user interests. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management* (pp. 949–958). ACM.

Pachet, F., & Aucouturier, J.-J. (2004, April). Improving timbre similarity: How high is the sky? *Journal of Negative Results in Speech and Audio Sciences*, 1–13.

Pachet, F., Westermann, G., & Laigre, D. (2001). Musical data mining for electronic music distribution. In *Proceedings of the 1st International Conference on WEB Delivering of Music* (pp. 101–106). IEEE Computer Society.

Pampalk, E. (2004). A Matlab toolbox to compute music similarity from audio. In *Proceedings of the 5th International Conference on Music Information Retrieval.*

Pampalk, E. (2006). *Computational models of music similarity and their application in music information retrieval* (PhD thesis, Vienna University of Technology, Vienna, Austria). Retrieved from `http://www.ofai.at/~elias.pampalk/publications/pampalk06thesis.pdf`

Pampalk, E., Flexer, A., & Widmer, G. (2005). Improvements of audio-based music similarity and genre classificaton. In *Proceedings of the 6th International Conference on Music Information Retrieval* (pp. 628–633).

Peeters, G. (2004). *A large set of audio features for sound description (similarity and classification) in the CUIDADO project* (Tech. Rep.). IRCAM.

Pohle, T., Knees, P., Schedl, M., & Widmer, G. (2007). Building an interactive next-generation artist recommender based on automatically derived high-level concepts. In *Proceedings of the 2007 International Workshop on Content-Based Multimedia Indexing* (pp. 336–343).

Rubner, Y., Tomasi, C., & Guibas, L. J. (2000). The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, *40*(2), 99–121.

Sánchez-Moreno, D., González, A. B. G., Vicente, M. D. M., Batista, V. F. L., & García, M. N. M. (2016). A collaborative filtering method for music recommendation using playing coefficients for artists and users. *Expert Syst. Appl.*, *66*, 234–244.

Sarwar, B. M., Karypis, G., Konstan, J. A., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International World Wide Web Conference* (pp. 285–295). ACM.

Schedl, M., Flexer, A., & Urbano, J. (2013). The neglected user in music information retrieval research. *J. Intell. Inf. Syst.*, *41*(3), 523–539.

Schedl, M., Gómez, E., & Urbano, J. (2014). Music information retrieval: Recent developments and applications. *Foundations and Trends in Information Retrieval*, *8*(2-3), 127–261.

Schedl, M., & Hauger, D. (2015). Tailoring music recommendations to users by considering diversity, mainstreaminess, and novelty. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 947–950). ACM.

Schedl, M., Knees, P., McFee, B., Bogdanov, D., & Kaminskas, M. (2015). Music recommender systems. In F. Ricci, L. Rokach, & B. Shapira (Eds.), *Recommender Systems Handbook* (pp. 453–492). Springer US.

Schedl, M., Knees, P., & Widmer, G. (2005a). Improving prototypical artist detection by penalizing exorbitant popularity. In *Proceedings of the 3rd International Symposium on Computer Music Modeling and Retrieval* (pp. 196–200).

Schedl, M., Knees, P., & Widmer, G. (2005b). A web-based approach to assessing artist similarity using co-occurrences. In *Proceedings of the 4th International Workshop on Content-Based Multimedia Indexing.*

Schmidt, E. M., & Kim, Y. E. (2011). Learning emotion-based acoustic features with deep belief networks. In *Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics* (pp. 65–68).

Shardanand, U., & Maes, P. (1995). Social information filtering: Algorithms for automating "word of mouth". In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 210–217). ACM Press/Addison-Wesley Publishing Co.

Shi, Y., Larson, M., & Hanjalic, A. (2014). Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *ACM Comput. Surv.*, *47*(1), 3:1–3:45.

Slaney, M. (2011). Web-scale multimedia analysis: Does content matter? *IEEE MultiMedia*, *18*(2), 12–15.

Slaney, M., Weinberger, K. Q., & White, W. (2008). Learning a metric for music similarity. In *Proceedings of the 9th International Society for Music Information Retrieval Conference* (pp. 313–318).

Song, Y., Dixon, S., & Pearce, M. (2012). A survey of music recommendation systems and future perspectives. In *Proceedings of the 9th International Symposium on Computer Music Modeling and Retrieval* (pp. 395–410).

Sotiropoulos, D. N., Lampropoulos, A. S., & Tsihrintzis, G. A. (2007). MUSIPER: A system for modeling music similarity perception based on objective feature subset selection. *User Model. User-Adapt. Interact.*, *18*(4), 315–348.

Stenzel, R., & Kamps, T. (2005). Improving content-based similarity measures by training a collaborative model. In *Proceedings of 6th International Conference on Music Information Retrieval* (pp. 264–271).

Stevens, S., Volkmann, J., & Newman, E. (1937). A scale for the measurement of the psychological magnitude pitch. *Journal of the Acoustical Society of America*, *8*(3), 185–190.

Su, X., & Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, *2009*, 421425:1–421425:19.

Turnbull, D., Barrington, L., Lanckriet, G. R. G., & Yazdani, M. (2009). Combining audio content and social context for semantic music discovery. In *Proceedings* *of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 387–394). ACM.

Tzanetakis, G. (2002). *Manipulation, analysis and retrieval systems for audio signals* (PhD thesis). Princeton University, Princeton, NJ, USA.

Tzanetakis, G., & Cook, P. R. (2002). Musical genre classification of audio signals. *IEEE Trans. Speech and Audio Processing*, *10*(5), 293–302.

Uitdenbogerd, A. L., & van Schyndel, R. G. (2002). A review of factors affecting music recommender success. In *Proceedings of the 3rd International Conference on Music Information Retrieval* (pp. 204–208).

van den Oord, A., Dieleman, S., & Schrauwen, B. (2013). Deep content-based music recommendation. In *Advances in Neural Information Processing Systems 26* (pp. 2643–2651). Curran Associates, Inc.

Vigliensoni, G., & Fujinaga, I. (2016). Automatic music recommendation systems: Do demographic, profiling, and contextual features improve their performance? In *Proceedings of the 17th International Society for Music Information Retrieval Conference* (pp. 94–100).

Vignoli, F., & Pauws, S. (2005). A music retrieval system based on user driven similarity and its evaluation. In *Proceedings of the 6th International Conference on Music Information Retrieval* (pp. 272–279).

Wang, X., Rosenblum, D. S., & Wang, Y. (2012). Context-aware mobile music recommendation for daily activities. In *Proceedings of the 20th ACM International Conference on Multimedia* (pp. 99–108). ACM.

Wang, X., & Wang, Y. (2014). Improving content-based and hybrid music recommendation using deep learning. In *Proceedings of the 22nd ACM International Conference on Multimedia* (pp. 627–636). ACM.

Welsh, M., Borishov, N., Hill, J., von Behren, R., & Woo, A. (1999). *Querying large collections of music for similarity* (Tech. Rep.). University of California, Berkeley, CA.

Whitman, B., & Lawrence, S. (2002). Inferring descriptions and similarity for music from community metadata. In *Proceedings of the 2002 International Computer Music Conference.*

Wiggins, G. A. (2009). Semantic gap?? Schemantic schmap!! Methodological considerations in the scientific study of music. In *Proceedings of the 11th IEEE International Symposium on Multimedia* (pp. 477–482). IEEE Computer

Society.

Wold, E., Blum, T., Keislar, D., & Wheaton, J. (1996). Content-based classification, search, and retrieval of audio. *IEEE MultiMedia*, *3*, 27–36.

Xu, C., Maddage, N. C., Shao, X., Cao, F., & Tian, Q. (2003). Musical genre classification using support vector machines. In *Proceedings of the 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing* (pp. 429–432).

Yoshii, K., Goto, M., Komatani, K., Ogata, T., & Okuno, H. G. (2006). Hybrid collaborative and content-based music recommendation using probabilistic model with latent user preferences. In *Proceedings of the 7th International Conference on Music Information Retrieval* (pp. 296–301).

Zhang, Y. C., Séaghdha, D. Ó., Quercia, D., & Jambor, T. (2012). Auralist: Introducing serendipity into music recommendation. In *Proceedings of the 5th ACM International Conference on Web Search and Data Mining* (pp. 13–22). ACM.