Automated generation of music playlists: Survey and experiments

GEOFFRAY BONNIN, Technische Universität Dortmund
DIETMAR JANNACH, Technische Universität Dortmund

Most of the time when we listen to music on the radio or on our portable devices, the order in which the tracks are played is governed by so-called playlists. These playlists are basically sequences of tracks which are traditionally designed manually and whose organization is based on some underlying logic or theme. With the digitalization of music and the availability of various types of additional track-related information on the Web, new opportunities emerged of how to automate the playlist creation process. Correspondingly, a number of proposals for automated playlist generation have been made in the literature during the last decade. These approaches vary both with respect to which kind of data they rely on and which types of algorithms they use. In this paper, we review the literature on automated playlist generation and categorize the existing approaches. Furthermore, we discuss the evaluation designs that are used today in research to assess the quality of the generated playlists. Finally, we report the results of a comparative evaluation of typical playlist generation schemes based on historical data. Our results show that track and artist popularity can play a dominant role and that additional measures are required to better characterize and compare the quality of automatically generated playlists.

Categories and Subject Descriptors: H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; H.5.5 [**Information Interfaces and Presentation**]: Sound and Music Computing

General Terms: Playlist generation, Music recommendation

Additional Key Words and Phrases: Music, playlist, algorithm, evaluation

1. INTRODUCTION

Much of the music that we listen to is organized in some form of a playlist. Such playlists are traditionally created manually, e.g., by a radio DJ or music producer, and usually have some internal organization logic, for example that their tracks belong to the same musical genre or are performed by related artists. Beside playlists that are, e.g., broadcast over the radio, the digitalization of music has also made it easier for users to create their own *personal* playlists. People use such personal playlists for a number of purposes, for example to create a collection of tracks used as background music for some event or activity or to share their musical preferences with others. Since the manual creation of playlists can be time-intensive or can require special skills by the playlist creator, a number of proposals has been made in the literature to automate the playlist creation process. Today, popular music platforms such as last.fm or Pandora and personalized Web radio services such as Musicovery rely strongly on automatically created playlists.

In this paper, we review and categorize the various approaches to automated playlist generation. We will look both on the different types of data that are used in the playlist construction process as well as on the various underlying algorithms. Furthermore, we will review and discuss common approaches to evaluate the quality of such automatically generated playlists. Based on a number of experiments we will then discuss possible limitations of existing offline evaluation schemes and current research practice. In addition, we show that simple playlist generation schemes based on popularity and artist information can – depending on the evaluation scheme – represent a comparably hard baseline. Finally, we will discuss perspectives for future research.

Authors' address: TU Dortmund, 44227 Dortmund, Germany

2. THE PLAYLIST GENERATION PROBLEM

In this section, we will first define the concepts "playlist" and "playlist generation", sketch typical challenges in the domain and discuss the relation of playlist generation with the fields of music information retrieval and recommender systems (RS).

2.1. Definitions

In this work, we define the term "playlist" as follows.

Definition 2.1 (*Playlist*). A *playlist* is a sequence of tracks (audio recordings).

This definition corresponds to what is most commonly used in the literature, see, e.g., [Baccigalupo and Plaza 2006; Knees et al. 2006; Baccigalupo 2009; McFee and Lanckriet 2011; Hariri et al. 2012]. Some authors make a distinction between playlists whose tracks are arranged in a certain order and playlists in which the tracks are shuffled. The terminology in the literature is however not always consistent. [Logan 2004], for example, talks about "song sets" when the tracks are shuffled and about "playlists" otherwise. [Cunningham et al. 2006], in contrast, use the term "playlist" for shuffled tracks and "mix" for sequentially ordered tracks.

Our definition is thus very general and covers several types of typical playlists:

— *Broadcasting radio playlists* are made by disc jockeys in radio stations. Such playlists often contain popular tracks in a relatively homogeneous style suited for a certain target audience [Ekelund et al. 2000].
— *Personalized radio playlists* are generated by Web music services such as last.fm[1] or Pandora[2]. These playlists can be very different from broadcasting radio playlists because the listeners directly interact with the service and the playlists can be adapted to the preferences of the listeners. Often, one of the goals of these playlists is "discovery", i.e., to help the listeners to find tracks or artists that are unknown to them.
— *Amateur playlists* are playlists made by non-professional music enthusiasts. Such playlists can have different purposes and people for example prepare them for certain future occasions like a party or a travel [Cunningham et al. 2006], for sharing their music with others or for self-expression.
— *Club playlists* are made by disc jockeys in clubs. Such playlists usually contain "danceable" tracks [Cliff 2006].
— *Album tracklists* are made by artists and labels and are usually a sequence of new tracks of one artist or a group of artists. To which extent labels allow artists to influence the selection and ordering or even recording of the tracks is typically dependent on the popularity of the artists [Baskerville and Baskerville 2010].
— *Compilation tracklists* are also created by artists and labels and, e.g., consist of popular tracks of one artist or are grouped around a theme such as "Christmas songs."

In this work, we focus on techniques that support or automate the process of creating playlists. Such approaches are applicable for the first four types of playlists. In our view, the content and structure of the last two types of playlists can be governed by factors that can be hardly automated, e.g., specific artist preferences or marketing considerations of the producers and labels.

All considered types of playlists have in common that their construction is determined by (a) the set of available tracks and (b) the target characteristics of the playlist. When building a playlist, a human creator will pick and arrange the tracks in a way that they fulfill some characteristics and will rely on some additional information. This

[1] http://last.fm
[2] http://pandora.com

information can for example include the genre, mood or tempo of the tracks but also the general or current popularity of the artist within the target audience.

To automate this task, a playlist generation tool in a first step has to include a mechanism to somehow acquire or assess the intended goals and thus, what is more important, the corresponding desired target characteristics of the playlist to be generated. In addition, a mechanism or heuristic is required that determines the suitability of a given track or track ordering given these target characteristics. The playlist generation problem can thus be defined as follows.

Definition 2.2 (*Playlist generation*). Given (i) a pool of tracks, (ii) a background knowledge database and (iii) some target characteristics of the playlist, create a sequence of tracks fulfilling the target characteristics in the best possible way.

Our definition is again quite broad in order to cover a wide range of application scenarios. Considering, e.g., an interactive scenario as on last.fm, the target characteristics of the playlist can be expressed by the user by providing an artist name or a genre, and then refined implicitly using the "skip" and "ban" buttons. The most recent listening history could also be considered in the target characteristics, i.e., a desired characteristic could be that the continuation is coherent with the most recently listened tracks. The background knowledge last.fm uses can include both genre information about the available tracks but also the popularity of individual tracks. In a one-time generation scenario, the goal could be to create a playlist for a certain activity or context, e.g., doing sports in the gym with a friend [Masahiro et al. 2008]. The required knowledge could be the tempo or mood of the tracks. Obviously, the listening context and the goal "suitability for sports" has first to be translated into target characteristics, e.g., by defining a minimum tempo or by requiring that the tracks are annotated with certain tags before the selection and ordering of the tracks.

2.2. Challenges of playlist generation

A number of aspects make automated playlist generation challenging. First, even if we talk of an automated process, the large number of available tracks that can potentially be included in the playlists – already personal music libraries can be huge [Dias and Fonseca 2010] – and the corresponding enormous amount of additional information that has to be processed can lead to high computational complexity. While some of the computations can be done in an offline model-learning process, the problem exists that many application scenarios require the "on-demand" generation of playlists or the immediate reaction on user feedback.

Another challenge lies in the acquisition of information about the tracks. Extracting characteristics from the audio signal for a large number of tracks can require significant computation times. On the other hand, if we rely on user-provided annotations, obtaining a set of high-quality annotations for each track can represent a major problem. Often, only a part of the tracks is annotated [Celma 2010] and the annotation quality can be low and subjective.

Finally, the taste and context of the listeners are two factors that can have a major influence on the *perceived* suitability or quality of a playlist and two users may actually perceive the same playlist differently even if listened to in the same context. The challenge is therefore not only to find tracks that match the specifics of the current user taste and context but also to identify the user's taste and context correctly in the first place. Questions of how the quality of playlists can be at least partially automatically and objectively assessed will be discussed in Section 5.

2.3. Relation to music information retrieval and recommender systems

Research on automated playlist generation is mainly done in two research fields: Music Information Retrieval (MIR) and Recommender Systems (RS).

2.3.1. Music Information Retrieval

MIR is an interdisciplinary research area that covers a wide range of tasks based on the automatic analysis of music data, among them playlist generation [Downie 2003; Grachten et al. 2009]. The scope of MIR goes thus far beyond the application of information retrieval (IR) to music data, whose scope is often limited to the retrieval of resources matching a user query [Crestani et al. 1998].

In general, playlist generation could be considered as a retrieval task if its definition is limited to the selection of tracks satisfying a user query. In traditional IR scenarios, the optimization goal is usually to find and rank all items that are relevant for a query. The quality of a playlist however cannot be assessed by only looking at its individual items as there might be desirable characteristics that concern the list as a whole, e.g., that it is diverse, contains items that are unknown to the user, or that track transitions are smooth. Furthermore, explicit queries are only one possible user input.

Other major topics in MIR include the automated extraction of features, the categorization of tracks or similarity analysis. These types of automatically identified information can obviously be a basis for building a playlist generation system, see, e.g., [Dopler et al. 2008; Lee et al. 2011].

2.3.2. Recommender Systems

RS are applications that provide personalized recommendations to users [Burke 2002]. Conceptually, the playlist generation problem can be considered as a special case of the recommendation problem, i.e., the recommendation of collections [Hansen and Golbeck 2009]. Furthermore, RS can be used to recommend various musical resources such as albums, artists or concerts and playlist generation can also be considered as a special case of music (track) recommendation. Again, however, the particularity of playlist generation is that the properties of the list itself and track orderings can be relevant.

Unlike in many RS research scenarios, the *immediate consumption* of the recommendations is common in playlist generation scenarios [Lamere 2012]. A playlist generator should thus be able to dynamically adapt its recommendations to the user's current situation, e.g., by incorporating implicit feedback like "skip" actions or other types of information from the user [Reddy and Mascia 2006].

Finally, differently from typical e-commerce scenarios evaluated in RS research [Jannach et al. 2012], the repeated consumption of items is very common in the music domain and listening logs like the ones available from last.fm show that many users listen to their favorite tracks over and over.

3. BACKGROUND KNOWLEDGE AND TARGET CHARACTERISTICS

According to our definition, three inputs are required to generate playlists: a pool of tracks, a background knowledge database and some target characteristics. Next, we will review existing playlist generation approaches with respect to the latter two inputs, i.e., (a) the used background knowledge (b) in which form the target characteristics and user inputs are obtained.

3.1. Background knowledge

The background knowledge corresponds to any information that can be used to determine whether a playlist and its tracks satisfy the target characteristics. We classify this information in the following categories: the audio signal, meta-data, Social Web data and usage data.

3.1.1. Musical features from the audio signal

An obvious type of data that can be used in the playlist generation process is the audio content. The goal of many MIR researchers is to automatically extract musical *features* from the audio signal [Pohle et al. 2005] including, e.g., pitch, loudness [Blum et al. 1999], rhythm, chord changes [Tzanetakis 2002], or simply compact representations of the frequency spectrum, in particular spectral clusters [Logan and Salomon 2001] and Gaussian mixture models [Aucouturier and Pachet 2002b]. Feature information can also be obtained through automated classification algorithms [Wold et al. 1996].

The advantage of relying on the audio signal over, e.g., expert annotations, is that the process is objective and can be automated to a large extent. Extracting the features can however be computationally costly [Schmädecke and Blume 2013]. Another limitation is that there might be features like the release date, the "freshness" or popularity of a track which can be relevant in the playlist generation process but which cannot be extracted from the audio signal [Celma and Serra 2008].

3.1.2. Meta-data and expert annotations

Meta-data shall be defined here as any information describing the tracks which is not derived from the audio signal, e.g., track attributes such as the year of release [van Gulik and Vignoli 2005], the label [Pachet et al. 2000], artist and genre information [Aizenberg et al. 2012; Bogdanov and Herrera 2011], lyrics [Coelho et al. 2013], etc. This information is often the result of a manual annotation process; in some cases, the information can be obtained from online music platforms or music databases like the Million Song Dataset; see the appendix for a list of public sources.

When used in an automated process, data completeness and consistency are crucial. Some parts of the meta-data like artist and track names can be easy to obtain [McFee and Lanckriet 2011] whereas, e.g., the year of release might be more difficult to obtain if it cannot be taken from a track database. Another potential problem is that not all types of meta-data are objective and annotations regarding, e.g., the mood or the genre of a track can be imprecise or inconsistent [Celma 2010; Lee and Cho 2011].

Generally, meta-data from expert annotations can be combined with automatically derived or cultural features [McKay 2010; Coelho et al. 2013]. Deriving the tempo of a track automatically can for example be challenging and inaccurate [Chen et al. 2009; Peeters and Flocon-Cholet 2012]. Therefore, in case of a low confidence, more reliable expert annotations could be used, which however induces higher costs. Relying only on expert annotations may however be too costly for the often large pools of tracks.

3.1.3. Social Web data

With the spread of the World Wide Web and particularly the Social Web, new sources of information about musical tracks have become available.

Tags. Community-provided annotations (tags) and comments made by users are common on many social platforms and can be obtained at comparably low cost through a crawling process or with pre-defined APIs. While such annotations can be rich and diverse, the perception of music is again subjective and can even be influenced by the perception of other people [McDermott 2012]. Furthermore, the tag quality can be low and less reliable than expert annotations, thus requiring data improvement or noise removal protocols. Finally, a sufficient number of tags might only be available for the most popular tracks but not for tracks from the long tail [Celma 2010].

Ratings. Explicit ratings are also typical on the Social Web and include 1-to-5 rating scales like on iTunes, 'like' or 'dislike' statements as on last.fm, as well as other forms of expressing an evaluation for a track, e.g., through a wall posting on Facebook [Germain and Chakareski 2013] or a tweet on Twitter [Hauger et al. 2013]. Ratings play a major

role in recommender systems research and are also particularly valuable for playlist generation as they can be used to create playlists that match the taste of the individual user. When relying on rating information, some challenges however exist, including the problem of data sparsity, especially for the tracks from the long tail, or a positivity bias, i.e., the phenomenon that most of the ratings are highly positive and negative feedback is rare [Celma 2010].

The Social Graph. On Social Web platforms, people are connected to each other through "friend" relationships or similar concepts, forming a social graph. When we assume that people who are connected on such platforms share similar tastes, we can exploit such relations and, e.g., create playlists containing tracks that are liked by the friends of a given user. A special type of relationships in that context are artist networks and the connections between individual artists as available for instance on MySpace or Facebook [Germain and Chakareski 2013]. As we will see later on, artists can play an important role in the playlist generation process. Knowing which artists are connected and related in some form can thus represent a valuable information.

3.1.4. Usage data

Web music platforms often provide information about the listening behavior of their users including listening logs for individual tracks or playlists or about the general popularity of individual tracks. Such information about patterns and regularities can serve as a valuable input to the playlist generation process [Zheleva et al. 2010].

Track popularity. The popularity of a track can be quite easily obtained based, e.g., on chart positions, playcounts or occurrences in user playlists. However, relying only on popularity to generate playlists may not be sufficient. In particular, playlists that only contain popular tracks can be of limited value when the goal is the discovery of new tracks. Still, when limited information is available about a new user, starting with relatively popular tracks can represent a reasonable initial strategy. Moreover, depending on the available data and the chosen evaluation measure, it has been shown that very simple popularity-based algorithms can outperform sophisticated algorithms in more general music recommendation scenarios [McFee et al. 2012].

Listening logs. Listening logs, which are available from some music platforms, can tell us much about the taste and the preferences of a user and thus represent a valuable information source for personalized playlist generation. Not only do we know which tracks are listened how often; if in addition "skip" actions are recorded, further inferences about the user taste or the suitability of a track within a playlist or listening context can be made [Hu and Ogihara 2011]. Listening logs have however to be interpreted with care as users may play some tracks in their full length simply because they were not listening to them [Bosteels et al. 2009].

Manually created playlists. As playlists shared by users are designed in a manual process for a certain purpose, we can to some extent assume that many of these lists obey certain quality criteria, e.g., in terms of homogeneity, with respect to their overall mood or at least the genre. Such hand-crafted playlists in addition might be based on relationships between tracks that cannot be captured using the audio signal or metadata [Maillet et al. 2009; McFee and Lanckriet 2011]. Once detected, these patterns and relationships can be used to guide the automated generation of playlists.

3.2. Specifying target characteristics for playlists

Automating the playlist generation process requires that the desired target characteristics are specified in a machine-interpretable form. These characteristics correspond to the overall purpose or intended goals of the playlist. We can identify three types

of such information that can be fed into the automated creation process, (i) explicit preferences and constraints, (ii) past user preferences and (iii) contextual and sensor information, which we discuss in the next sections.

3.2.1. Explicit preferences and constraints

We identified four typical ways for users to specify their short-term preferences and constraints in the literature: (1) seed tracks, (2) free-form keywords, (3) explicit, predefined constraints and (4) real-time feedback about tracks.

(1) The provision of seed tracks by the user can be done in different ways. The typical options are (a) the selection of a single seed track [Logan 2002], (b) the specification of a start and end track [Flexer et al. 2008], (c) a full list of tracks to be sorted or mixed with other tracks [Logan 2004], or (d) a list of tracks already contained in the playlist or recently played (playlist history) [Baur et al. 2010; Hariri et al. 2012].
(2) Free-form inputs are common on Web music services such as last.fm. The users can enter keywords which are then matched with the tracks. Typical inputs could be artist names, track names or genres. The keywords can however also represent the user's mood or information about the user's activity or environment [Reddy and Mascia 2006; Jensen et al. 2007; Meyers 2007; Coelho et al. 2013].
(3) Explicit pre-defined constraints are typically provided with standard user interface elements such as check boxes, sliders or selection boxes [Pachet et al. 2000; Reddy and Mascia 2006; Sandvold et al. 2006]. The constraints can concern various aspects including the desired genre, tempo, year of release, etc.
(4) Finally, users can express their preferences by immediately rating (like/dislike) the tracks in the created playlist [Pauws and Eggen 2002] or by skipping individual tracks broadcasted by a personalized online radio station [Pampalk et al. 2005].

3.2.2. Past user preferences

The user's past preferences can represent another input guiding the playlist creation process. Such preferences can be explicitly stated (ratings) but also estimated by the system. Typical sources for estimates are the user's personal track library on the device, the listening history or playlists that the user has created in the past.

The *library of tracks* owned by the user can be seen as an indicator for the long-term taste of the user. In case the library is huge, it is however unclear if the user has ever heard or even knows all the tracks [Lamere 2012]. Thus, this sort of information can be unreliable and in addition does not contain "negative" feedback. The user's *listening logs* are therefore often more informative, especially when "skip" actions are recorded. Finally, when the user has created playlists in the past, it is intuitive to assume that the user knows and likes the tracks in the playlists. Again, however, the playlists do not provide us with information about what the user does not like.

3.2.3. Contextual and sensor information

The suitability of a certain track can finally be determined by incorporating short-term contextual information about the environment and the listeners. With the increasing capabilities of mobile devices and the immediate availability of various types of information on the Web, the opportunities to include additional contextual and sensor information continuously increase. In the literature, a number of such approaches have been proposed, including the types of sensors and information shown in Table I.

4. PLAYLIST GENERATION ALGORITHMS

Next, we will review strategies for playlist generation. We organize these strategies in seven categories: similarity-based algorithms, collaborative filtering, frequent pattern mining, statistical models, case-based reasoning, discrete optimization and hybrids.

Table I. Examples for additional information used in the playlist generation process

Information	Sensor	Reference
Physiological state	ECG, Accelerometer	[Oliver and Flores-Mangas 2006; Oliver and Kreger-Stickles 2006] propose to determine the user's physiological state with the help of sensors and use it to enhance their exercise performance. Similarly, [Biehl et al. 2006] select tracks based on the user's heart beats per minutes and pace.
Environment	Cameras, trackers	Cameras and trackers can provide information, e.g., on the number of clients on a dancefloor in the context of an automated club DJ [Cliff 2006].
Sound, noise	Microphones	Microphones were used to obtain information such as the sound of walking [Moens et al. 2010; Jylhä et al. 2012] or surrounding noise [Reddy and Mascia 2006].
Location	GPS	The location of the user can be provided by a GPS [Reddy and Mascia 2006]
Time	Clock	Time information can easily be accessed from the computer or mobile device on which the system is deployed [Liu et al. 2009; Hu and Ogihara 2011].
Other context information	Web	Information such as the temperature of a city, the traffic level, etc. can be found on the Web [Reddy and Mascia 2006].

4.1. Similarity-based algorithms

The coherence of the tracks is a typical quality criterion for playlists [Logan 2004; Knees et al. 2006; Fields et al. 2008]. Therefore, selecting and ordering tracks based on their similarities is an obvious strategy to generate playlists. The core of any similarity-based approach is its *distance function*, which characterizes the closeness of two tracks. How the distance function is actually designed depends on the available data, which could include the raw audio signal along with the features that can be derived from it, but also meta-data, e.g., the artists, the genres, playcounts, or ratings [Slaney and White 2007]. In many cases, a signature or model of each track is determined in a first step on which the distance function is then applied. Typical examples for such functions applied on more abstract models of a track's features are the earth-mover's distance [Logan 2004], the Kullback-Leibler (KL) divergence [Vignoli and Pauws 2005], or the Euclidean distance [Knees et al. 2006].

Once these similarities are computed, different strategies of determining the tracks in the playlist are possible. In [Lehtiniemi and Seppänen 2007], for example, tracks are selected based on their distance to the seed track. Another approach is to repeatedly select a track according to the similarity with the previously selected one [Chen et al. 2012]. Quite differently, one could also simply select those tracks that have the highest similarity with the tracks the user liked [Pampalk et al. 2005; Gartner et al. 2007], which corresponds to a typical content-based recommendation approach. Clustering techniques also rely on similarity functions, see, e.g., [Pauws and Eggen 2002] and [Dopler et al. 2008] and for example exploit similarities based on artist information and other meta-data or audio features. [Ragno et al. 2005], [Pohle et al. 2007] and [Hartono and Yoshitake 2013] finally represent examples of a very specific way of using the track similarities, where the goal is to optimize the playlist in a way that the sum of the similarities is maximized.

One possible drawback of pure similarity-based approaches is that the similarity (homogeneity) is often considered as the main and possibly only quality criterion. Therefore, the danger exists that such playlists are too homogeneous or contain too many tracks from the same artist and are thus not well suited for the discovery of new

tracks. Depending on the application scenario and the specific technique, similarity-based approaches can also be computationally expensive [Budhraja et al. 2012].

One way to reduce the computational complexity is to rely only on artist similarities as done in [Hauver and French 2001], [Dopler et al. 2008] or [Fields et al. 2008]. The similarity of artists can be estimated in different ways, e.g., based on artist ratings, the similarity of artist descriptions, or based on the relationships between artists on music platforms like MySpace. Still, the resulting playlists can in some cases be not very coherent as there are artists who perform music of different genres or styles.

4.2. Collaborative filtering

Collaborative Filtering (CF) is the predominant approach in the field of RS [Jannach et al. 2012] and is solely based on community-provided ratings. These ratings can be either provided explicitly or automatically collected (e.g., by recording track listening logs). In principle, any of the various collaborative filtering techniques can be used for playlist generation, provided that there exists a reasonably large dataset with ratings. Still, CF approaches were not designed for the specific challenges of playlist generation, and aspects like list homogeneity or smooth transitions between the tracks have to be addressed separately. In addition, in contrast to the large rating datasets that exist in general RS research, not many datasets exist in the music domain and research is often based on very small samples [Liu et al. 2009].

Since rating information can be sparse, CF methods are often combined with other techniques. [Chedrawy and Abidi 2009], for example, propose a hybrid system which is based on a semantic similarity function and multi-attribute ratings from users on playlists with respect to lyrics, rhythm, tunes, performance and overall likability. In their approach, they first identify additional existing playlists that are relevant for the user and then construct a new *personalized* playlist using tracks from these playlists.

A different way of applying CF techniques for playlist generation is to consider *playlists as users* and tracks as items in the sense of a binary user-item rating matrix. The idea is that when we are given a playlist which is currently being constructed, we consider the set of already selected tracks as the positive ratings. Then, we can for example apply a nearest-neighbor approach and look for similar playlists that contain some of these tracks. Tracks appearing in similar playlists can then be considered as candidates for inclusion in the new playlist. This last approach was for instance applied and compared to the more elaborate BPR approach (Bayesian Personalized Ranking) from [Rendle et al. 2009] in [Hariri et al. 2012].

One of the main limitations of such approaches is what is called the “new user” problem, i.e., the question what to additionally include in the playlist when not much information about the user is available (or the set of start tracks is small). This problem is even more important when considering playlists as users, as each playlist to generate is equivalent to a new user.

4.3. Frequent pattern mining

The above-mentioned neighborhood-based method for playlist generation relies on the co-occurrence of items in playlists. Frequent pattern mining approaches are based on a similar principle. However, instead of only looking on local neighbors for a given playlist, they try to identify global patterns in the data. There are two types of such patterns: association rules (AR) [Agrawal et al. 1993] and sequential patterns (SP) [Agrawal and Srikant 1995]. AR are often applied for shopping basket analysis problems and have the form A $\Longrightarrow$ B, where both A and B are sets of items and the rule expresses “whenever A was bought, also the items in B were bought with some probability” [Han and Kamber 2006]. Sequential patterns are similar, except that the order

of the elements in A and B is relevant. Whether or not sequential patterns are more accurate and preferable over association rules depends on the data characteristics.

Frequent patterns for playlist generation can for example be mined in samples of hand-crafted playlists provided by the user community. When there are tracks that frequently appear together in such playlists, the assumption can be made that the tracks have something in common and that they both fulfill the target characteristics that the creator of the playlist had in mind. The mined patterns can be used for playlist generation as follows. Given a start track or a recent listening history, search for patterns whose left-hand side elements are contained in the history and consider the elements on the right-hand side as candidates to extend the playlist.

To the best of our knowledge, no "pure" application of frequent patterns for playlist generation has been proposed so far. The most similar approach are n-gram models, i.e., Markov models of order $n-1$ in which the transition probabilities are derived from the track co-occurrences, which can be considered to be similar to contiguous sequential patterns of fixed size n. [Crampes et al. 2007] propose to train a trigram model (n-grams with $n = 3$) on playlists made by DJs. [Chen et al. 2012] include a bigram model smoothed with the Witten-Bell discounting [Jurafsky and Martin 2009] in a comparative evaluation. Mining for frequently co-occurring tracks is however only one option and one could also mine patterns of frequently co-occurring artists or genres in playlists. [Hariri et al. 2012], for example, propose to mine sequential patterns of latent topics based on the tags attached to the tracks.

One of the main advantages of frequent patterns is that they allow us to implicitly reproduce the observed characteristics of manually-defined playlists without looking at the "content" of the tracks. A possible drawback is that the quality of the generated playlists depends on the number and quality of the playlists used for rule extraction.

4.4. Statistical Models

Playlists can also be generated using statistical models[3]. One of the simplest approaches is to assume that each track selection is only dependent on the previous track, which corresponds to Markov models. The transition probabilities for such models can be estimated based on various strategies, using, e.g., track co-occurrences in playlists (thus leading to the aforementioned bigram model) or the similarity of the audio signal or the meta-data.

A comparison of different basic strategies can be found in [McFee and Lanckriet 2011], who compare Markov models based on a uniform distribution, artist popularity, tags and the audio signal. The experiments show that artist popularity leads to the best results in terms of the average log-likelihood measure (cf. Section 5) and that an optimized mixture of all the models further enhances the results. In [Chen et al. 2012], a more sophisticated Markov model is proposed, which is called Latent Markov Embedding (LME). The authors represent tracks as points in the Euclidean space and derive the transition probabilities based on the Euclidean distance. The coordinates of the track are learned using a likelihood maximization heuristic. In [Moore et al. 2012], the same authors extend their model by including tag information. Another statistical approach was presented in [McFee and Lanckriet 2012] which is based on random walks on a hypergraph, i.e., an undirected graph whose edges can connect any

[3]Notice that frequent pattern mining approaches can also be considered as statistical estimation methods. In this paper however, we dedicated a whole section to them because they represent one of the most straightforward solutions to playlist generation, a number of algorithms for efficient pattern mining exist and our own recent work shows that they work comparably well [Bonnin and Jannach 2013a]. Interestingly, however, these methods were so far not very frequently used in the literature. Similarly, CF can use statistical models. However, as it is not the only possible strategy and since CF is the predominant approach of building RS, we dedicate an own section to this approach.

number of vertices. Based on the edge weights, their playlist generator first selects an edge containing the previously selected track and then chooses one of the remaining tracks of that edge in a uniform manner. The weights of the edges are learned using the L-BFGS-B algorithm [Byrd et al. 1995].

A major drawback of the some of these approaches is that assuming the Markov Property may actually be too strong for the domain. Researchers have therefore proposed models in which more complex dependencies can be expressed using graphical models. In [Zheleva et al. 2010], two models called the "taste model" and the "session model" are proposed. Both models select tracks based on distributions over tracks using latent clusters that were obtained by applying Latent Dirichlet Allocation (LDA) [Blei et al. 2003] on the usage data. The difference between the two models is that the first one only uses one latent cluster per listening session and can thus be considered as reflecting the user's taste. The second model can use a different cluster for each track which the authors consider as reflecting the user's current "mood" (contextual taste). Later on, a similar approach was proposed by [Hariri et al. 2013]. In their approach, the latent clusters can also incorporate tag information, which allows the use of queries for specifying preferences.

In general, one main advantage of using statistical models is that many algorithms exist to optimize the values of the parameters, as illustrated by [Chen et al. 2012] and [McFee and Lanckriet 2012]. Moreover, different statistical models can be easily combined through linear interpolations, which can help to further enhance the results [McFee and Lanckriet 2011]. One possible drawback however is that the learning process can be highly time consuming, as will be discussed in Section 6.

4.5. Case-Based Reasoning

The general idea of Case-Based Reasoning (CBR) techniques is to exploit information about problem settings (cases) encountered in the past to solve new problems [Althoff 2001]. CBR approaches therefore first store a set of representative cases together with their solutions. Given a new problem instance, the case repository is searched for past similar cases whose solution is then adapted according to the specifics of the new case.

A CBR-based technique has for example been proposed for playlist generation in [Baccigalupo and Plaza 2006]. The case base in their scenario consists of a set of playlists created by a user community. Within these playlists, frequent sub-sequences (patterns) are identified. In contrast to typical CBR-approaches, the goal of their work is however not to find the most similar playlists given some seed track, but those that are considered to be the most "useful", e.g., in terms of diversity. The elements of the retrieved playlists are then combined to generate a new playlist for a given seed track.

Later on, the same authors proposed to apply CBR techniques in the specific context of a broadcasting radio where the listeners can give feedback on the selected tracks [Baccigalupo and Plaza 2007]. In this setting, the system uses a case representation of the user preferences to adapt the selection of tracks for a given group of listeners.

One possible advantage of typical CBR-techniques is that their computational complexity can be comparably low when only a limited number of cases is used. The specific techniques proposed by Baccigalupo et al., however, may be limited in their scalability as they scan the entire database upon new playlist generation requests.

4.6. Discrete optimization

A quite different approach is to consider playlist generation as a discrete optimization problem. Given a set of tracks, their characteristics and a set of explicitly specified constraints capturing the desired characteristics, the goal is to create one arbitrary or an optimal sequence of tracks that satisfies the constraints. Examples for constraints

could be a given start or end track, that the tracks in the playlist are taken from at least n different genres, or constraints related to the transitions between the tracks.

Technically, the problem can be modeled for example as an Integer Linear Program as done in [Alghoniemy and Tewfik 2000; 2001]. In an alternative formulation, [Pachet et al. 2000] proposed to model the task as a Constraint Satisfaction Problem (CSP) and apply the existing search algorithms to select sequences of tracks that satisfy various constraints, in that case from a comparably small catalogue.

When we adopt a standard CSP formulation, no solution can be found in case the problem is overconstrained. In such situations, however, "soft constraints" approaches can be applied. In [Aucouturier and Pachet 2002c; 2002a], the authors propose to build playlists by iteratively enhancing a randomly chosen playlist through a local search procedure. The quality and suitability of the resulting playlist can be assessed by counting the number of satisfied constraints. From a technical perspective, different local search procedures are possible including, e.g., Simulated Annealing [Pauws et al. 2006; 2008] or genetic algorithms [Hsu and Chung 2011].

The advantage of these approaches is that if the used background knowledge is accurate, only playlists will be generated that satisfy all or most of the target characteristics. A potential drawback is the high computational complexity which makes the problem intractable for larger music collections [Pauws et al. 2008].

4.7. Hybrid techniques

In the field of recommender systems, hybridization is often used to combine the advantages of different techniques and at the same time avoid the drawbacks of individual techniques. [Burke 2002] proposes a classification in seven categories for hybridization of recommender systems, which is reproduced in Table II.

Table II. Hybridization methods.

Hybridization method	Description
Weighted	The scores from several recommendation techniques are combined.
Switching	The system switches between recommendation techniques depending on the current situation.
Mixed	Recommendations from several different recommenders are presented at the same time.
Feature combination	Features from different recommendation data sources are thrown together into a single recommendation algorithm.
Feature augmentation	The output of one technique is used as an input feature to another one.
Cascade	One recommender refines the recommendations given by another.
Meta-level	The model learned by one recommender is used as input to another.

Hybrid approaches can also be found in the literature for playlist generation. [McFee and Lanckriet 2011] for example uses a weighted hybridization approach – a mixture model – that combines algorithms that rely on track similarity, track familiarity and the uniform distribution. The same hybridization method is used by [Chedrawy and Abidi 2009], who combine a multi-criteria kNN-based collaborative filtering algorithm with an ontology-based item similarity approach. [Hariri et al. 2012] also use a weighting strategy and combine a frequent pattern mining approach with the kNN-approach.

This last work additionally uses a feature augmentation strategy as they first extract latent topics from tags and then extract frequent patterns from them. [Meyers 2007] is an example for feature combination. Features extracted from the audio file and lyrics are combined to generate playlists matching a certain mood. Generally, also approaches in which features are inferred from the audio signal and then combined with manually annotated meta-data can be considered as a form of feature combination, see, e.g., [Aucouturier and Pachet 2002c; Vignoli and Pauws 2005]. For more information about hybrid recommendations outside of the specific scope of playlist generation see, e.g., [Burke 2002; Lampropoulos et al. 2012; Hornung et al. 2013].

5. EVALUATING PLAYLIST GENERATION SYSTEMS

In general, *user satisfaction* could be considered as the ultimate criterion to assess the quality or usefulness of the generated playlists. Whether a user is satisfied with a playlist can however depend on various criteria. The satisfaction can for example be influenced by the extent to which the list matches its intended purpose, fulfills the desired target characteristics, or is in line with the user's expectations including aspects of taste, context or mood [Fields 2011]. In the following, we will first discuss general quality criteria for playlists from the literature and then review approaches to evaluate the quality of playlist generation systems.

5.1. Quality criteria for playlists

One first way to determine the general principles for a good playlist design is to analyze the characteristics of playlists that are created and shared by users. Such an analysis can be found for example in [Slaney and White 2006], where the role of the *diversity* of playlists was in the focus. The chosen research approach was to use an objective measure that captures the diversity of playlists and then to analyze if users tend to create rather diverse or more homogeneous playlists. More precisely, tracks were represented as points in a genre-space and the diversity was defined based on the volume of the enclosing ellipsoid of the tracks of the playlist. This metric was applied to 887 user playlists containing about 30,000 tracks and the output shows that diversity actually matters for playlists[4]. [Sarroff and Casey 2012] investigated which track features are the most important for track transitions. Their results highlighted the importance of fade durations and the mean timbre of track endings and beginnings.

A different approach to determine the guiding rules of playlist creators was chosen by [Cunningham et al. 2006], who analyzed the messages posted on the Art of the Mix forum. Their analysis revealed that the most typical principles for track selection mentioned in these messages are the artist (e.g., the best tracks of a given artist), the genre, the style, the event or activity, the semantics (e.g., "romance") and the mood. Additional criteria mentioned in these messages are the maximum number of consecutive tracks of the same artist or genre or the smoothness of the transitions. Some playlist creators also organize their playlists in a way that the most popular or most "important" tracks appear at the end of the playlist as a sort of "grand finale".

Another possible approach is to perform user studies aimed at determining quality criteria for playlists. An example for such a study was presented in [Hansen and Golbeck 2009]. Their study involved 52 participants and one of the main results was that the choice of the first and the last tracks was considered to be particularly important. In a smaller study involving 24 participants, [Andric and Haus 2005] showed that for playlists that contained only tracks the users preferred in general, the order of the tracks and even the selection of tracks has a low impact on the perceived quality.

[4]To which extent the chosen objective diversity measure corresponds to the diversity *perceived* by the users was however not been analyzed in this work.

In [Reynolds et al. 2008], the results of an online music survey with 750 participants are presented, which showed that the user's context or environment (location, activity, temperature, lighting or weather) can have a strong influence both on the mood of the listeners and the track selection. Another smaller study involving eight subjects was done in [Lee et al. 2011]. The identified key criteria included the variety of the list as well as meta-data features including lyrics, theme and time. Furthermore, having a mix of familiar tracks and new music was considered to be a factor for satisfaction. Finally, [Kamalzadeh et al. 2012] presented the results of a larger study involving 222 participants which again showed the importance of criteria related to the mood, the genres, and the artists of the tracks.

Overall, such analyses give us important insights about what users find important and which criteria can be used to judge the quality of playlists. However, in our view, more research in that respect is required and one possible direction for future work could be a systematic approach for quality criteria identification as was done in [Wang and Strong 1996] for the Information Quality field.

5.2. Evaluation approaches for automatically generated playlists

Various approaches have been proposed in the literature to evaluate the quality of playlists. [McFee and Lanckriet 2011], for example, listed three categories of evaluation strategies: human evaluation, semantic cohesion and sequence prediction. In this work, we propose to organize the possible evaluation approaches in four more general categories, which we will discuss in the following subsections: (1) User studies, (2) Log analysis, (3) Objective measures and (4) Comparison with hand-crafted playlists.

5.2.1. User studies

User studies are a form of evaluation that helps us to determine the *perceived* quality of playlists with good confidence. One of the major drawbacks of such studies however is that they are time consuming and expensive. The particular problem in the domain is that in many experimental designs the participants are asked to listen to all the tracks of the playlist. This can take a long time, even if only excerpts are played. In settings where the participants evaluate the playlists only based on meta-information or evaluate playlists consisting of tracks they already know, the obtained results might be biased and depend, e.g., on the general popularity of the individual tracks.

A typical problem in the context of user studies reported in the literature is their size. Many studies are based only on 10 to 20 participants who evaluate playlists with a questionnaire or give feedback using an application that was developed for the study. The later form of evaluation often scales better. [Barrington et al. 2009], for example, present the results of a study involving 185 subjects. Unfortunately, the reproducibility of such designs is limited as the experiment is based on a specific software application. Finally, the studies in the literature are often based on quite different collections of tracks and the generalizability of the findings is not always clear.

5.2.2. Log analysis

Instead of asking the users about their subjective evaluations, one can look at how the generated playlists are accepted by the users by analyzing their behavior using listening and interaction logs. Such logs for example contain information about how often each user listened to each playlist. In addition, explicit "like" or "dislike" statements or user actions such as skipping to the next track can be recorded.

Log analysis is something that would usually be done in A/B tests performed by music platform providers. In such a setting, the listeners are split into different groups and each group would receive playlists that are generated with different techniques. After the test has ended, e.g., after 4 weeks, the logs could be analyzed to determine

which playlist generator worked best, e.g., based on the like statements or play counts. Obviously, researchers usually cannot perform A/B tests with real users. Still, there are at least platforms from which information about the listening behavior can be obtained. In particular, last.fm gives access to the listening logs of its users through their public API. So far, however, only limited research has been based on such logs, the work of [Bosteels et al. 2009] being one of the few examples, who however used a very specific evaluation protocol. They retrieved sequences of tracks of length 22 such that the last two elements were not both accepted or rejected (one was accepted and the other rejected). Different playlist generation heuristics were then compared by counting how often they could rank the accepted track before the rejected track.

5.2.3. Objective measures

With the term objective measures, we refer to measures that can be automatically computed for a given playlist and which try to *approximate* a user-perceived *subjective* quality [Cremonesi et al. 2011]. A typical example for such a subjective measure is the diversity or homogeneity of a playlist. One basic way of assessing the diversity of a given playlist could be to simply count the number of different artists or genres appearing in the playlist. Several works base their evaluations on variants of such a strategy and for example determine the genre distribution only based on meta-data [Pohle et al. 2005; Knees et al. 2006; Flexer et al. 2008; Dopler et al. 2008], or on a combination of meta-data and the audio signal [Cai et al. 2007]. Some authors simply consider tracks to be similar if they are from the same artist [Slaney and White 2007] or the same genre [Balkema and van der Heijden 2010]. This might however be too simplistic, given that there are artists who cover a broader spectrum of genres and that the tracks of some genres such as jazz can be quite heterogeneous.

Considering a single quality criterion might however not be sufficient to assess the quality of a playlist. Determining, e.g., the homogeneity alone might be misleading and some works indicate that diversity can be as important as homogeneity [Slaney and White 2006; 2007; Lee et al. 2011; Kamalzadeh et al. 2012]. Therefore, more comprehensive evaluation approaches are required that use multiple measures at the same time and in addition analyze typical trade-offs such as homogeneity vs. diversity. Beside diversity and homogeneity, also other quality criteria can be approximated through a numeric measure, e.g., the novelty, the freshness of the tracks, their familiarity with respect to a certain user, the consistency of the mood and even the smoothness of the transitions, e.g., based on the tempo and instrumentation of the tracks.

A limitation of such approaches is that it is not always clear if the objective measure truly approximates the subjective quality experience. For instance, a playlist covering several genres may actually not be perceived to be diverse, e.g., because the genre annotations of tracks can be subjective and some genres might be similar to each other.

5.2.4. Comparison with hand-crafted playlists

The final category of evaluation approaches discussed here is based on the estimation of the ability of the algorithms to reproduce hand-crafted playlists, i.e., playlists created by music enthusiasts. The assumption is that such playlists respect various of the described quality criteria. An automatically created list should therefore in some form be similar to manually created ones and a playlist generation algorithm correspondingly should rely on patterns that are similar to those of real users. In the literature, two measures and protocols can be found that are based on such a strategy.

Hit rates. The hit rate is a measure commonly used in the domain of information retrieval (IR). The principle is to count how many of the actually relevant items were

contained in a top-N result list for a given query[5]. The measure is also applied in the field of recommender systems [Jannach et al. 2012]. Here, the input is however rather a user profile – typically information about items that the user liked in the past – than a query. In both cases, to apply this measure, the "ground truth", i.e., whether or not an item is considered to be relevant, has to be known or approximated.

A similar principle can be applied to evaluate the ability of the algorithms to generate good playlists. The idea is to take a hand-crafted playlist and "hide" some of the tracks. The task of the playlist generation system is then to guess the hidden items. Each correctly guessed hidden item corresponds to a hit and the more often a playlist generator "agrees" with the hand-crafted list, the better.

One limitation of such a strategy is that it is based on the assumption that the hidden tracks are the only relevant ones. Clearly, however, many other tracks may also be considered relevant. If the number of these tracks could be known, then recommendation lists of exactly this size could be computed. Such lists should then at least contain the test track. Although it is impossible to determine the exact number of relevant tracks, the length of the recommendation list can be considered as an assumption about the average number of truly relevant tracks. Therefore, it makes sense to look at how the results change when the size of the recommendation lists varies, and to use values well beyond what users might conceivably browse through. Notice that this strategy should not be considered as a means to estimate the number of true positives, but mainly as a way to observe the results under different assumptions.

In general, any subset of the playlist elements could be hidden. When we however remove more than one element, the playlist generation algorithm might not be able to take the role of the transitions between the tracks fully into account. One option is therefore to hide only one track at a time and consider the preceding tracks in the playlist as an input. Such a protocol was for example used in [Hariri et al. 2012], who hide the last element of each tested playlist[6].

Technically, to measure the hit rate, we split the available set of playlists into a training set $Train$ and a test set $Test$. Then each playlist in $Test$ is split into the current set of given tracks and the remaining set of tracks to be recommended. If we limit ourselves to the recommendation of the last track t given the history of the i first tracks $h = (t_1, ..., t_i)$, the hit rate can be measured as follows:

$$HitRate(Train, Test) = \frac{1}{\|Test\|} \sum_{(h,t) \in Test} 1_{R(h)}(t) \tag{1}$$

with $R(h)$ is a recommendation list of a defined length computed by an algorithm based on $Train$, h the playlist beginning and $1_{R(h)}(t)$ an indicator function and indicates the membership of t in $R(h)$[7].

Average log-likelihood. The average log-likelihood $\mathcal{ALL}$ is a measure that is commonly used in the domain of natural language processing to assess the quality of the modeling techniques [Rosenfeld 2000]. The measure can be used to assess how likely a system is to generate the tracks of a given set of assumedly good playlists and has first been used in [McFee and Lanckriet 2011]. Conceptually, the generation process is

[5]Alternative terms for hit rate are true positive rate, recall or sensitivity.

[6]Given that the very last element can play a special role for playlist creators, this strategy might not be optimal. Still, in our evaluations in Section 6, we will rely on this scheme to make our work comparable with existing research.

[7]As an alternative to the use of hit rates to evaluate playlists, researchers sometimes use measures such as the "half-life of user interest" [Platt et al. 2001; Xiao et al. 2009]

assumed to be based on a weighted random process, where the generation algorithm assigns a probability of being selected to each of the tracks. Thus, the application of this measure requires that the algorithms assign probabilities to the candidate tracks.

Technically, given a test set of playlists, the average log-likelihood $\mathcal{ALL}$ is given by:

$$\mathcal{ALL}(Train, Test) = \frac{1}{\|Test\|} \sum_{(h,t) \in Test} \log \left(P_{Train}(t \mid h) \right) \tag{2}$$

where $P_{Train}(t \mid h)$ corresponds to the probability of observing t given h according to a model learned based on $Train$. Research works that use this measure for playlists include [McFee and Lanckriet 2011], [Chen et al. 2012] and [Moore et al. 2012].

In contrast to the hit rate, the $\mathcal{ALL}$ measure is not interpretable on an absolute scale: the possible values range from $-\infty$ (at least one test track has a probability of 0) and 0 (all probabilities for all test tracks are 1). This measure therefore only allows us to compare the results of different approaches without knowing if the best one is actually good. It can thus be considered as a complementary measure to the hit rate.

Note that when an algorithm assigns a probability of zero to one single test track, we would obtain an $\mathcal{ALL}$ value of $-\infty$. Such zero probabilities must therefore be avoided. This can be achieved through a smoothing step, e.g., by combining the probabilities of a given model, which were learned on a training dataset, with the probabilities from the uniform distribution using a mixture model (see section 4). The corresponding weights can then be learned on yet another (optimization) dataset using some optimization algorithm [McFee and Lanckriet 2011]. However, due to the long-tailed distribution of the tracks with respect to their popularity, a large part of the tracks of the optimization dataset do not appear in the training dataset. The larger the number of these tracks, the larger the weights assigned by the optimization algorithm to the uniform distribution. The weights from the uniform distribution might thus be too large and bias the algorithms' estimates too much towards the niche items.

5.3. Other evaluation criteria for playlist generation systems

So far, we have focused on measures that are related to the quality of the playlists themselves. There are, however, additional criteria that can be considered when comparing playlist generation algorithms and the corresponding applications. We will only discuss two of them briefly here: scalability and user effort. Other criteria can be found in [Vatolkin et al. 2011].

5.3.1. Scalability and computational complexity

In many application domains, being able to dynamically compute contextualized playlists or playlist continuations is a crucial requirement. Online services such as last.fm should more or less be able to immediately create a playlist for a given set of keywords and a personalized radio station should be able to react to the skip actions of a user in a reasonable time frame.

In contrast to the domain of recommender systems where this is a niche topic, scalability and computational complexity are comparably often discussed in the context of playlist generation, see, e.g., [Pohle et al. 2005], [Knees et al. 2006] and [Cai et al. 2007]. One possible explanation is that one of the most prominent approaches for playlist generation is discrete optimization, which involves the exploration of large search spaces or the maximization of the number of satisfied constraints. Finally, while *learning a statistical model* can be done offline, the required running times can also be problematic in particular when the pool of available tracks is huge.

5.3.2. User effort

Whether or not users are satisfied with a system can largely depend on its user interface. Design, ergonomics and in particular how much effort is required by the user to specify the target characteristics of a playlist can play a major role. An example for research in that direction is [Hilliges et al. 2006], where the goal is to make the playlist creation process more convenient for users by using an audio similarity map with colors for mood constrains. Another example is the Rush system [Baur et al. 2010], in which users receive repeated recommendations and can thereby construct the playlist in an incremental manner. In the experiments in [Pauws and van de Wijdeven 2005], finally, the authors include measures such as the time spent for creating a playlist or the number of required user actions and relate them to the resulting playlist quality.

Many popular systems including last.fm or the iTunes Genius provide relatively simple mechanisms for the user to specify the desired playlist characteristics. These mechanisms typically include the selection of a seed track or free text input that can correspond to the preferred genre or artist. On last.fm, users can furthermore decide to listen to a number of pre-defined "radios" whose playlists are determined by the system depending on the listening history of the current user or his or her friends. The user can however not directly influence what is played on these radios.

Generally, in our view more research is required with respect to what can be expected from the user. One question here is how much effort users are willing to spend on the input specification and playlist refinement process. Another question is how to enable the users to specify their preferences in an intuitive and convenient way.

6. A COMPARATIVE EVALUATION OF PLAYLIST GENERATION TECHNIQUES

In the following section, we report the results of a comparative evaluation of typical playlist generation approaches[8]. Although user studies are a reliable means to determine the quality of playlists, we argue that interesting new insights can also be obtained based on historical data. We thus adopt an offline experimental design and base our experiments on large pools of hand-crafted playlists shared by music enthusiasts as described in Section 5.2.4.

As evaluation measures of playlist quality, we use hit rates and the average log likelihood $\mathcal{ALL}$. Although these measures do not explicitly address aspects like diversity, homogeneity, track freshness and others, as mentioned in Section 5.2.4, they implicitly take them all into account and reflect the general user preferences. Moreover, one of the goals of our experiments is to identify possible limitations of these measures.

Through our experiments, we aim to analyze the following research questions:

RQ1: Are algorithms that take the order of the tracks in their models into account favorable over those that only look at track co-occurrences?
RQ2: How important are artist-related information and popularity aspects when creating playlists?
RQ3: Do the hit rate and the $\mathcal{ALL}$ measure agree with respect to the ranking of the algorithms and are there differences when different datasets are used?
RQ4: What is the computational complexity of the different techniques?

6.1. Experimental procedure

Our experiments are based on pools of hand-crafted playlists which we randomly split into training (90%) and test set (10%) in a 10-fold cross-validation procedure. The task of the playlist generation algorithms is to compute scores (which can be probabilities) for the possible next tracks given the first tracks of each playlist in the test set.

[8]See also our previous works [Bonnin and Jannach 2013a] and [Bonnin and Jannach 2013b].

6.1.1. Algorithms

We used six algorithms with different characteristics in order to answer our research questions. The specific details about the individual techniques are given in Table III.

(1) kNN: This algorithm takes the playlist beginning h and looks for other playlists in the training data that contain the same tracks. The main assumption is that if there are additional tracks in a very similar playlist, chances are good that these tracks suit the playlist beginning h. Recent works have shown that this technique represents a strong baseline [Hariri et al. 2012][9].
(2) Association Rules (AR): This technique (see Section 4) generates association rules from the training set. Given a history h, the playlist generation algorithm looks for rules whose left-hand side matches some tracks in h. The confidence of the tracks on the right-hand side is then used to compute the scores of the tracks.
(3) Sequential Patterns (SP): The way we implemented this algorithm is equivalent to the previous one with the only difference that the order of the elements in the rules is taken into account[10].
(4) Same Artist – Greatest Hits (SAGH): This method is based on a proposal from [McFee et al. 2012] and the idea is simply to only consider tracks of artists that already appear in h. The score of the tracks is then based on their popularity. In our case, popularity is measured in terms of how often a track appears in the playlists in the training set. We include this method in our evaluation to assess the importance of artist information and popularity.
(5) Collocated Artists – Greatest Hits (CAGH): We propose this method as an extension to the previous one. Instead of only considering artists that appear in h we also include the greatest hits of "similar" artists. As a measure of artist similarity, we simply count how often two artists appear together in the playlists of the training set, i.e., how often they are collocated.
(6) PopRank: This algorithm simply recommends the tracks by decreasing order of their overall occurrence in the playlists.

6.1.2. Datasets

We used three sets of playlists which we obtained from different music platforms.

(1) *last.fm*: We retrieved this dataset, which comprises 50,000 playlists, from last.fm through their public API.
(2) *Art of the Mix (AotM)*: The dataset was made available by McFee and Lanckriet and the subset we finally used contained 50,000 playlists.
(3) *8tracks*: This dataset was provided to us directly by 8tracks. The dataset used in the experiments comprises about 100,000 playlists.

The last.fm dataset contains playlists that were manually created and shared by the users of the platform. One difference when compared to the other datasets is that creating and sharing playlists is not the main feature of last.fm but merely an additional feature. AotM is a comparably old website (1997) for creating and sharing playlists. Although it does not allow users to listen to the created playlists, it is extensively used by a very active community of music enthusiasts who exchange ideas and try to maximize the quality of their playlists. 8tracks is a more recent website (2006) dedicated to sharing playlists and has the advantage that its users can listen to the tracks of the

[9]In fact the more sophisticated hybrid approach from [Hariri et al. 2012] only slightly outperforms the kNN method for low values of k. They could also show that the kNN method is better than the recent learning-to-rank technique BPR [Rendle et al. 2009] and a content-based approach for their dataset.
[10]For a detailed description of similar versions of these approaches see [Nakagawa and Mobasher 2003].

Table III. Details of the algorithms used in the evaluation.

Algorithm	**Details**
kNN	As for every kNN method, a similarity function is required to determine the neighborhood. We calculate this using a binary cosine similarity measure between the playlist history h and each hand-crafted playlist p: $$\text{sim}(h,p) = \frac{\|h \cap p\|}{\sqrt{\|h\|\|p\|}}$$ Given N_h nearest neighbors for a playlist history h, the score of a track t is defined as: $$\text{score}_{k\text{NN}}(t,h) = \sum_{n \in N_h} \text{sim}_\text{p}(h,n) \cdot 1_n(t) \quad (3)$$ with $1_n(t) = 1$ if n contains t and 0 otherwise.
Association Rules (AR) and Sequential Patterns (SP)	We use the Apriori algorithm for the extraction of association rules and a simple variant thereof for the sequential patterns. For both techniques we can vary the minimum values for support and confidence as well as the maximum size of the patterns and the maximum size of the sliding window. Given a set of extracted patterns, the score of a track t given a playlist history h is computed as follows: $$\text{score}_{pattern}(t,h) = \sum_{\omega \in \Omega} \text{confidence}(\omega \rightarrow t) \quad (4)$$ with Ω is the set of all possible rule antecedents that can be considered in h and confidence$(\omega \rightarrow t)$ the confidence of the rule $\omega \rightarrow t$, i.e., the conditional probability of t given ω.
Same Artists – Greatest Hits (SAGH)	Given a playlist history h, the algorithm determines the set of artists appearing in h and then assigns scores to their tracks depending on how often these tracks appeared in playlists in the training set. Given a history h, the score for a track t with artist a is computed as follows: $$\text{score}_{\text{SAGH}}(t,h) = \text{counts}(t) \cdot 1_h(a) \quad (5)$$ with $1_h(a) = 1$ if a is also the artist of one of the track in h and 0 otherwise, and counts(t) is the number of occurrences of t in the dataset, which corresponds to the greatest hits of the dataset.
Collocated Artists – Greatest Hits (CAGH)	We propose to select the greatest hits of artists that already appear in h or are similar to the artists of h. We compute the similarity between two artists a and b as follows: $$\text{sim}_\text{a}(a,b) = \frac{\sum_p (1_p(a) \cdot 1_p(b))}{\sqrt{\sum_p 1_p(a) \cdot \sum_p 1_p(b)}}$$ with $1_p(a) = 1$ if playlist p contains a and 0 otherwise. The similarity thus depends on the collocations of artists within playlists, which can be computed offline. Our proposed formula for the computation of the score of a next track t with artist a given a playlist beginning h is as follows: $$\text{score}_{\text{CAGH}}(t,h) = \sum_{b \in A_h} \text{sim}_\text{a}(a,b) \cdot \text{counts}(t) \quad (6)$$ where A_h is the set of artist names of the tracks in h and counts(t) is the number of occurrences of t in the dataset. Similar to the frequent patterns, we can vary the minimum values of artist co-occurrences.

playlists online. However, this feature comes at the cost of some limitations, e.g., that only a certain number of tracks of one artist can be contained in one single playlist.

For the last.fm and Art of the Mix data sets, we removed playlists of size 1 and random playlists that contain tracks appearing only once (i.e., only in the considered playlist) until reaching a size of 50,000 playlists. To enhance the quality of the data, we used the API of last.fm to correct artist and track misspellings and used the existing MusicBrainz IDs of the resulting tracks as a basis to identify tracks.

More details about the characteristics of the datasets are given in Table IV. The datasets are partially quite different, e.g., with respect to the average length of the playlists or how many artists appear on average in a playlist. The row "artist reuse rate" in the table is the percentage of cases when the artist of the last track of a playlist already appeared in the same playlist before. The "usage count" statistics express how often a track or artist on average appeared in the playlists.

Using three datasets with quite different characteristics should allow us to analyze how the different algorithms perform in different situations. At least two characteristics are of major importance: the size of the datasets and the sparsity of track usage. Regarding these characteristics, our datasets have complementary properties: one dataset has a limited size and sparsity (last.fm), one dataset has a limited size and a higher sparsity (AotM) and one dataset has a larger size and a smaller sparsity. Overall, generating playlists using the AotM data is expected to be the most difficult task. Other factors may however play a major role as well, in particular that 8tracks users are only allowed to add two tracks of the same artist in one playlist.

Table IV. Properties of the datasets.

	last.fm	AotM	8tracks
Playlists	50,000	50,000	99,542
Users	47,603	11,655	51,743
Tracks	69,022	453,270	179,779
Avg. tracks/playlist	7.6	19.4	9.7
Avg. track usage count	5.5	2.1	5.3
Artists	11,788	119,878	29,352
Avg. artists/playlist	4.5	16.9	8.9
Avg. artist usage count	32.2	8.1	32.7
Artist reuse rate	31.1%	21.3%	13.8%

6.2. Evaluation results

RQ1: Relevance of track order in models

Our first set of experiments is concerned with the question: "Does the order matter?" To that purpose, we made a comparison between the Association Rules (AR) and the Sequential Patterns (SP) method. In this experiment, we used hit rates as the only measure to assess the playlist quality and manually optimized the support and confidence threshold values for each dataset[11]. In order to make our results comparable, we applied the experimental procedure that was used in [Hariri et al. 2012].

Figure 1 shows the hit rates for the three datasets. When comparing the absolute values for the different datasets, we can observe that they can be quite different across

[11]We do not present the results in terms of the $\mathcal{ALL}$ because of the limitations of this measure, as will be presented in RQ3.

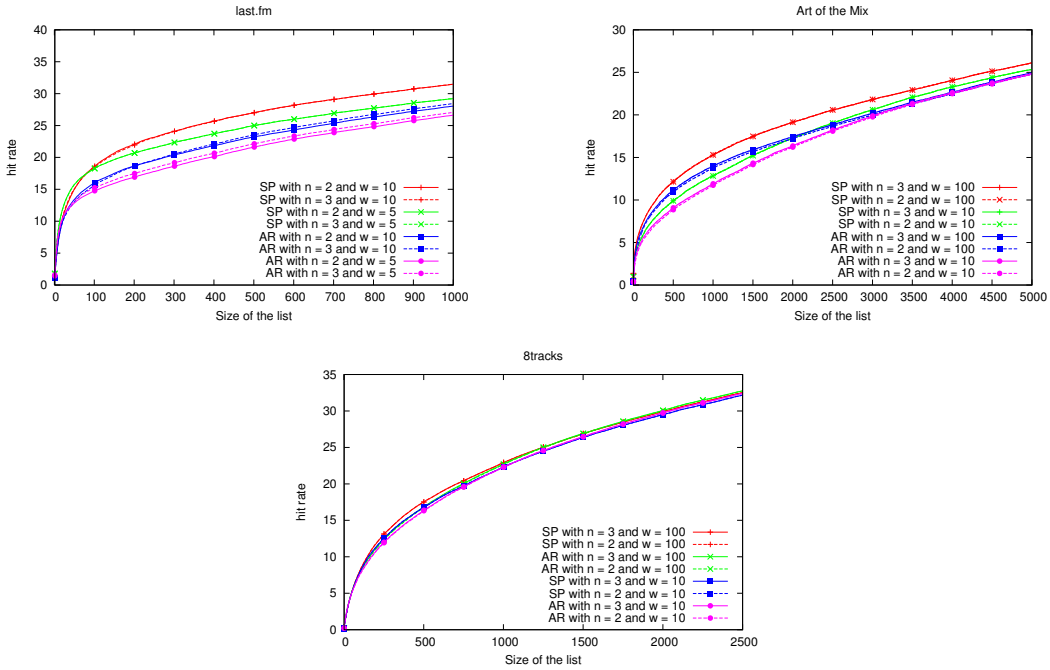

Fig. 1. Hit rates of the different frequent pattern mining techniques for the three datasets. AR = Association Rules, SP = Sequential Patterns, n = max. size of patterns, w = window size.

the datasets. This is however not surprising given the different characteristics of the datasets. As expected, the highest values are observed for the last.fm dataset, which has a low sparsity, and the lowest values were obtained for AotM.

Regarding research question RQ1 and the role of the track orderings, both for the AotM and the last.fm datasets, the Sequential Patterns lead to statistically significant higher hit rates than the Association Rules according to a one-way ANOVA analysis and a Tukey post-hoc test (the largest p-value equals 0.03 on last.fm and 10^{-5} on AotM, and the smallest Cohen's d values equals 1.44 on last.fm and 1.21 on AotM). This indicates that integrating information about track orderings can actually help to increase the quality of the generated playlists. The effect however seems to depend on the dataset and no significant difference could be found for the 8tracks dataset.

A side observation in this context is that larger window sizes seem to be preferable for the Sequential Patterns approach. This means that it is better not to look only on the very recent playlist history. In the subsequent experiments, we will therefore use the SP approach with larger window sizes.

RQ2: The role of artist-related information and track popularity

To determine to which extent artist and popularity information are relevant for the playlist generation process, we compared the hit rates of the artist-agnostic techniques kNN, Sequential Patterns[12] and PopRank to the SAGH and CAGH algorithms. We ran the kNN method with three comparably large neighborhood sizes to see how it influences the results. We also manually optimized a minimum threshold regarding

[12]While Sequential Patterns do not exploit information about artists, the popularity of tracks is implicitly taken into account as the support and confidence of the rules are based on the occurrences of their elements in the playlists.

artist co-occurrences for the computation of the artist similarities. Again, we used hit rates as the measure to assess the playlist quality and hid only the last track.

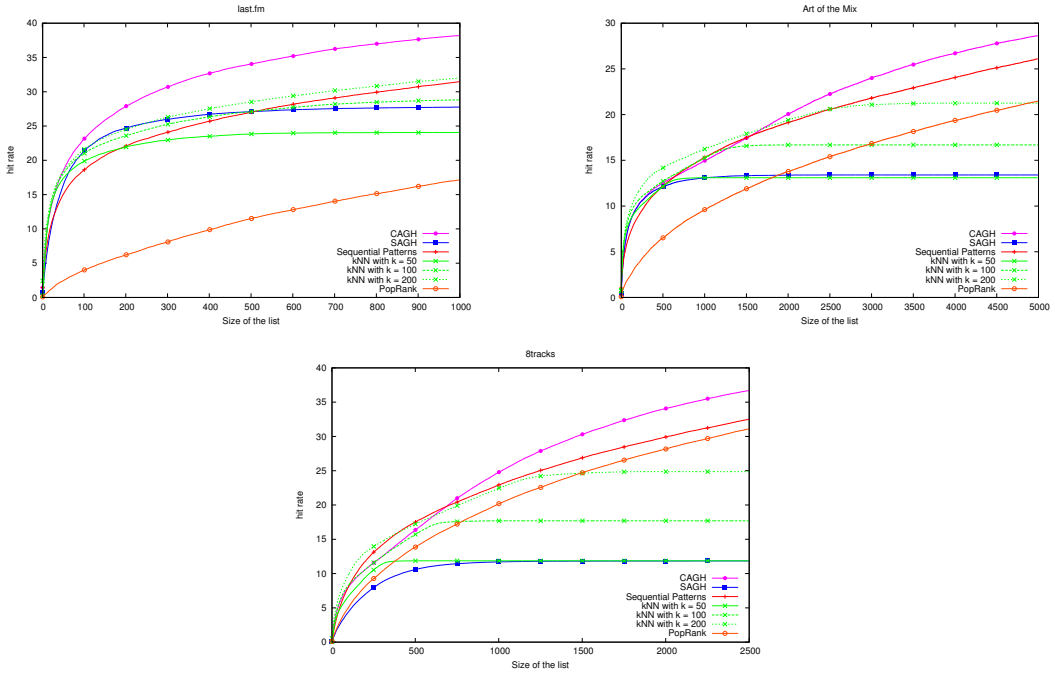

Fig. 2. Hit rates of the different algorithms.

Figure 2 shows the hit rates obtained by the different approaches for the three datasets. Overall, we can see that the CAGH approach outperforms the other approaches. These differences were again statistically significant for the last.fm and the AotM datasets (the largest p-value equals 10^{-4} on last.fm and 10^{-8} on AotM, and the smallest Cohen's d value equals 2.73 on last.fm and 5.11 on AotM). With respect to research question RQ2 this means that including tracks from similar artists can be a good strategy to improve the quality of the generated playlists. In fact, even a very simple strategy like CAGH seems to be able to capture interesting characteristics of what users consider to be a good playlist.

It is worth noting that playlist creators on 8tracks are not allowed to have the same artist more than twice in one playlist. Still, the CAGH algorithm performs well and would perhaps be even better if this constraint would not be there. Another interesting particularity of the results on this dataset is the surprisingly high hit rates of the popularity-based approach for lists longer than 1500 tracks. In other words, playlist creators on 8tracks include one of the 1,500 most popular tracks (out of about 180,000) as the last track of the playlist about 25% of the time, which means that the head of the long-tailed distribution of track usage is particularly large. The Sequential Patterns approach generally works well on all datasets. Considering the track ordering and to some extent the popularity of the tracks thus appears to be a good strategy in general.

These results also give an insight on how the sparsity of track usage affects the performance of the different algorithms. As for the previous results, we can see that the lowest values were obtained for the AotM dataset, which has the highest sparsity. However, this is only true for some of the algorithms. In particular, the SAGH

algorithm has higher hit rates on AotM than on 8tracks, which could be caused by the aforementioned constraint for the playlists of 8tracks. The same phenomenon can however be observed for the kNN algorithm with 50 neighbors, on which the constraint of the 8tracks dataset should not have such strong effects. This can in turn be explained by the aforementioned large size of the head of long-tailed distribution on this dataset (the higher the number of neighbors, the more popular tracks can be taken into account).

Beside the experiments whose results are shown in Figure 2, we also experimented with models based on the Markov property, among them the simple bigram model and the Latent Markov Embedding (LME) model of [Chen et al. 2012]. Despite the long time that can be required to train these models – e.g., several weeks for the LME model – these methods led to particularly low hit rates values which were consistently below 10% on the three datasets. We therefore omit these results in this paper. In general, given these differences, assuming the Markov property might be too strong for this problem setting.

Overall, given the results of this and the previous measurements, we can conclude that the choice of the "best" playlist generation strategy can to some extent depend on the dataset characteristics. At the same time, our measurements suggest that considering different information sources including artist information or general track popularity can help to improve the results of individual techniques. In our view, future research could thus (a) aim to develop appropriate hybridization designs and (b) explore the use of algorithms that take the particularities of the datasets into account[13].

RQ3: Comparing hit rate and $\mathcal{ALL}$ results

In a next step, we evaluated the different algorithms using the $\mathcal{ALL}$ measure to compare it with our findings regarding the hit rates. The results are shown in Figure 3.

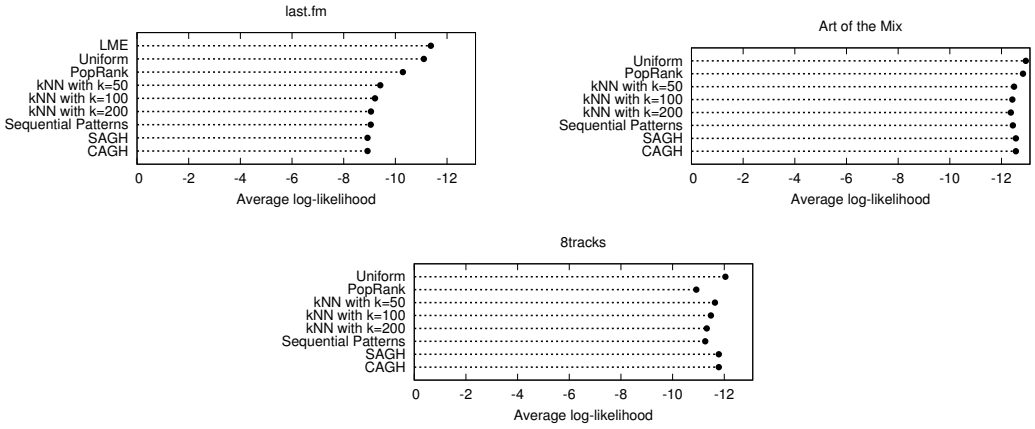

Fig. 3. Average log-likelihood of the evaluated approaches on the different datasets

As described in Section 5, this evaluation approach requires a smoothing step to avoid zero probabilities. To make the measurement, we therefore developed variants of the described algorithms in which the returned scores correspond to probability values

[13]In the context of frequent pattern mining, the usage of techniques that are particularly suited for long-tail distributions and that are capable of handling "rare" itemsets [Weiss 2004], could for example be explored.

and combined them with a uniform distribution in a mixture model. These values were obtained using a linear normalization of the scores of the tracks:

$$p_M(t|h) = \frac{score_M(t,h)}{\sum_i score_M(i,h)}$$

Notice that using a linear normalization is of course not the only possible strategy. Each training set of the cross-validation process was additionally split into a learning set to obtain the model probabilities and a validation set to compute optimized weights using the Expectation-Maximization algorithm. We also provide the results of the aforementioned LME model with standard parameters for the last.fm dataset. On both other datasets, the training process lasted more than a month.

The figure shows that the $\mathcal{ALL}$ results are different from the hit rate results and there is, for example, no consistent "winner" across the datasets. The CAGH method only outperforms the others very slightly on the last.fm dataset. On the AotM dataset, generally no strong differences between the algorithms were observed. On the 8tracks dataset, finally, the PopRank method outperformed the others. Generally, the differences between the methods are very small. The explanation for this phenomenon lies in the quite extreme long-tail distributions regarding the popularity of the tracks. When the probabilities of the playlist generation methods are combined with the uniform distribution, too much weight is put on the long-tail tracks. As a result, the differences between the techniques are dampened.

In order to verify our explanation about the similar results, we also evaluated all the mixture models of this section in terms of the hit rate and obtained very similar outputs for all algorithms, which we do not include for space reasons. In particular, the hit rate of the CAGH model is lowered to an extent that both approaches CAGH and SAGH lead to similar low hit rates.

Regarding research question RQ3 we conclude that the ranking of the algorithms can depend on the used measure and that $\mathcal{ALL}$ measurements do not necessarily follow the trend of the hit rate measure. At the same time, using the $\mathcal{ALL}$ measure exhibits major limitations. First, it can only be used to compare algorithms and it does not tell us if the generated playlists are actually good. Second, the measure requires model smoothing in order to avoid zero probabilities. This in turn can result in generally lower values of the estimated quality which can finally lead to very small differences between different algorithms, at least when a linear normalization is used.

RQ4: Computational complexity

Finally, we draw our attention to questions of computational complexity and the running times (in seconds) required to generate playlists. Figure 4 shows the running times that the different algorithms required to compute 1,000 predictions on the three datasets. For each dataset, we averaged the running time of each algorithm on each test set of the 10-fold cross-validation. In order to minimize the effect of possible perturbations, each of these running times corresponds the minimum of 10 runs. The experiments were run on a Linux computer with an AMD Opteron Processor 6272 CPU and required around 500 MB of RAM in the worst case.

As expected, the running time of the kNN algorithm is much higher than that of the other algorithms. The results of this algorithm are strongly dependent on the number and size of playlists that have to be scanned to determine the neighbors for each prediction. Therefore, the running times on the 8tracks dataset are much higher than on the last.fm dataset, and much lower than the results on the AotM dataset. Notice that in the case of playlist generation, neighborhoods cannot be pre-computed as the playlist history to be continued is not known in advance. The measurements also show

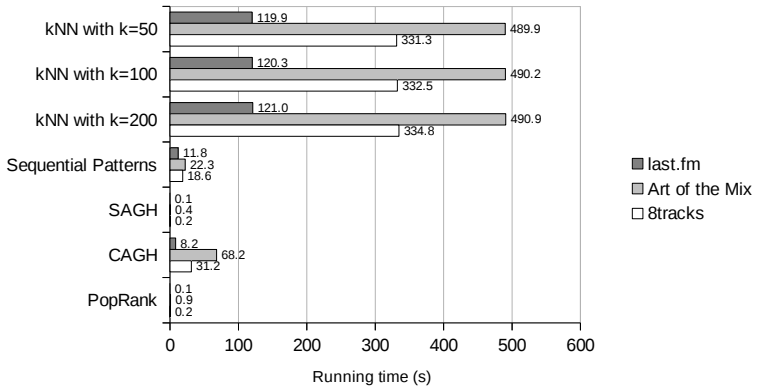

Fig. 4. Time required for 1,000 predictions by each algorithm.

that the value of k only had a small influence on the running times. Another expected result is the very short running time of the SAGH and PopRank algorithms. Finally, the results show that when using the CAGH algorithm or the sequential patterns, the recommendations can be generated quite fast.

Since the observed running times depend on a number of factors including the effectiveness of the implementation, we made a theoretical analysis of the properties of the different techniques compared in this paper. The observed running times reported here correspond to the insights of the theoretical analysis. The details of this analysis can be found in the appendix.

7. RESEARCH PERSPECTIVES AND CONCLUSIONS

In this work, we have reviewed the recent literature on automated playlist generation and categorized the different proposals with respect to the types of background data they use, the types of inputs they can process, and the algorithms on which they base the generation process. In the course of the literature review, we collected a number of statistics about which kinds of approaches are popular in the research community. Based on these statistics and our observations from the empirical evaluation reported in Section 6, we derived a number of limitations of current research and a subjective list of potential directions for future research.

7.1. Data perspective

We first draw our attention to the different ways of specifying the target characteristics, as discussed in section 3.2. Figure 5 shows a statistic on how often each method was used in the literature reviewed for this paper.

In the research literature, approaches that rely on a given seed track or on explicit constraints, e.g., on genre or mood, are the most common ones. Quite interestingly, even though most online music services allow their user to enter free-text input (e.g., in the form of keywords), this type of user interaction is barely addressed in the research community. Interpreting user inputs based on individual keywords can however be challenging. Typing in, for example, the term “eclipse” on last.fm’s music search field results in dozens if not hundreds of artists with the keyword appearing in the band name, hundreds of track and album titles, and even two dozen different labels [Lamere 2012]. Another type of input that is not in the focus of the research community is the usage of long-term user preferences model which can be for example based on past user interactions, listening logs or explicit preference statements in the user profile.

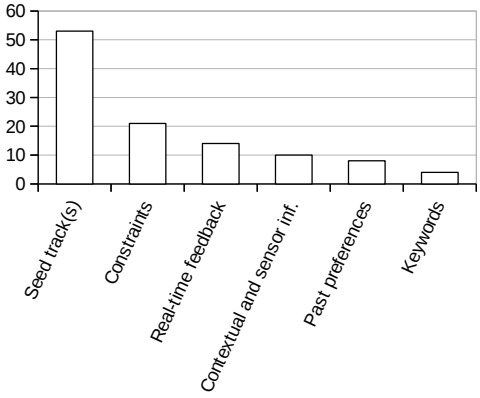

Fig. 5. Methods used in the literature to specify target characteristics.

Future research could therefore aim to explore the challenges and opportunities when different types of user inputs should be supported. The interpretation of individual keywords could for example be based on information from the current user's profile. At the same time, such profile information could also be the basis for personalizing the generated playlists. Another interesting perspective is that of contextual playlisting, as only a few works focus on it, e.g., [Reddy and Mascia 2006; Chiarandini et al. 2011]. Some inspiration could be taken from the more general domain of music recommendation, where this is an emerging topic [Kaminskas and Ricci 2011].

Figure 6 gives an overview on which types of background knowledge is used in the literature and at the same time gives statistics about the corresponding algorithmic approaches[14]. The figure shows that most existing research is based on features extracted from the audio signal or some additionally available information, e.g., about genres or artists. This observation is in general not particularly surprising as many of the reviewed papers appeared in publication outlets of the MIR community. However, the figure clearly highlights that potentially valuable sources of information that can be obtained from Social Web sources or from listening logs are not yet fully explored and thus represent a promising area for future research. Although the number of papers we reviewed is not very large, it seems that more and more papers deal with such type of background knowledge. However, the majority of recent research we reviewed still uses the audio signal and meta-data.

7.2. Algorithms perspective

Figure 6 also reveals that similarity-based algorithms are the most common approach to generate playlists, which is an obvious approach when the goal is to maximize the homogeneity of the playlists. Correspondingly, however, the diversity and serendipity of the tracks can be low. Interestingly, collaborative filtering techniques, which dominate the RS research field, are not very frequently used in the literature even though platforms like last.fm at least partially base their radios on such techniques[15].

Looking at the dates of publication, we could see that most of the research based on discrete optimization date back to around the year 2000. Since then, only a few

[14]The numbers were obtained by classifying only the papers that were reviewed in the context of this work. Therefore, they should be interpreted as being rough indicators.
[15]http://www.last.fm/help/faq

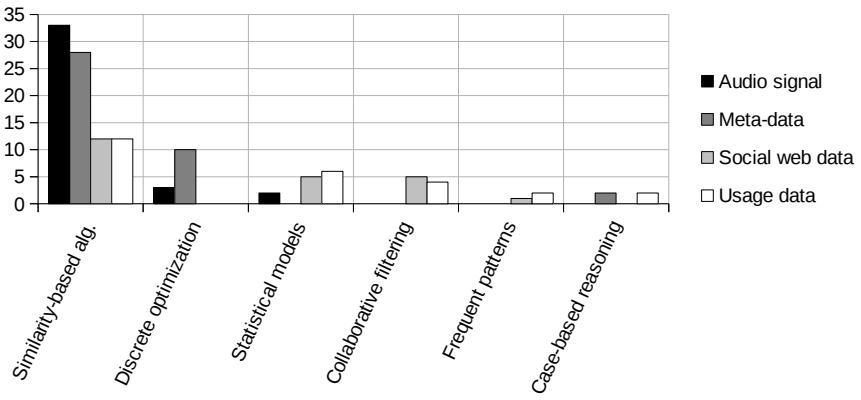

Fig. 6. Types of background knowledge per type of algorithm in the literature.

papers deal with this type of algorithms and approaches based on similarity analysis represent the majority even in the most recent research.

In future works, the goal could thus be to better exploit the additionally available information about the tastes of users and the community to build not only more diverse playlists but in particular generate playlists that are *personalized* according to the user preferences. In that context, hybrid systems that combine different types of information sources like [Hariri et al. 2012] appear to be particular promising as the inner consistency of a playlist can be relevant, which perhaps can only be achieved if meta-data about the tracks is considered.

Since smooth transitions of tracks are often mentioned as an important quality criterion in the literature, the value of better incorporating such aspects into playlists generation approaches could be explored in more depth in future research. Finally, no substantial amount of research has been done on automatic playlisting for social groups. This is however an interesting topic as music is often shared and used for social interaction (parties, motivation for team sport, romantic evening, etc.).

7.3. Evaluation perspective

We finally looked at what represents the "common practice" in research to evaluate automatically generated playlists. Given the subjectivity of how playlists are perceived by users, user studies are not surprisingly an important means used by researchers to assess the quality of playlists. Overall, about one third of the papers that we reviewed used some form of user studies to validate their hypotheses. Another third of the papers however also relied on historical data, offline experimental designs and objective quality measures. About 10 percent of the papers (additionally) used individual examples to discuss the quality of the generated playlists. Another 10 percent however did not contain any proper evaluation at all. The remaining papers evaluated other aspects of the generation system as a whole like the computational complexity.

When the evaluation was based on historical data (e.g., on hand-crafted playlists) or other experimental designs not involving users, IR measures like the hit-rate are common. The $\mathcal{ALL}$ measure only plays a minor role. As another "objective" measure, the homogeneity of the generated playlists is very often in the focus of researchers.

Overall, while user studies are probably the best means for evaluation, these studies often have their limitations, in particular with respect to their size and the selection of the participants. We therefore argue that the available (log) data – including listening

logs – should be better exploited in future research as there are a number of aspects that could be assessed in offline experimental designs. In particular, future research could try to assess multiple criteria at the same time and explore existing trade-offs like homogeneity vs. diversity or familiarity vs. novelty [Jannach et al. 2013]. The result of such a research could be a guideline of which playlist generation strategy is particularly suited for a given playlist goal. If, for example, the goal is the discovery of new songs, collaborative filtering methods could be favorable. If the playlist tracks should be on a certain theme, other techniques could be better.

Furthermore, additional objective measures beyond homogeneity could be designed to assess different quality criteria including novelty, serendipity or the smoothness of the track transitions. In any case, however, experimental studies should be made to analyze the correspondence with the user-perceived quality.

Finally, while there exists a number of papers that explicitly address questions of computational complexity and scalability, this topic needs in our view more attention, in particular as our experiments have shown that some advanced playlist generation methods do not scale well and require weeks of training already for medium-sized problem settings.

7.4. Conclusions

In this paper, we have reviewed different approaches to automatically generating music playlists. After having characterized the playlist generation problem and its challenges, we have looked on both what kinds of data are used by such systems and which technical approach they adopt to create the playlists. Our literature review furthermore included a discussion of common approaches of how to evaluate the quality of the generated playlists.

In the second part of the paper, we compared several of the existing techniques from the literature on different datasets using different measures of quality and complexity. Among other aspects, our evaluation showed the importance of taking into account the order of the tracks, artist information and track popularity in the generation process. The results also indicate that some of the measures applied in the literature have their limitations and that the outcomes can strongly depend on the characteristics of the underlying datasets. Last, the results of the experiments on the computational complexity confirmed the importance of this dimension when designing algorithms for playlist generation.

Based on our literature review and the experiments we have finally sketched a number of possible directions for future research. These directions include the incorporation of additional knowledge sources and the use of additional measures that can help us to better characterize the quality of playlists.

REFERENCES

Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. 1993. Mining Association Rules between Sets of Items in Large Databases. In *Proc. SIGMOD*. 207–216.

Rakesh Agrawal and Ramakrishnan Srikant. 1995. Mining Sequential Patterns. In *Proc. ICDE*. 3–14.

Natalie Aizenberg, Yehuda Koren, and Oren Somekh. 2012. Build Your Own Music Recommender by Modeling Internet Radio Streams. In *Proc. WWW*. 1–10.

Masoud Alghoniemy and Ahmed H. Tewfik. 2000. User-defined Music Sequence Retrieval. In *Proc. Multimedia*. 356–358.

Masoud Alghoniemy and Ahmed H Tewfik. 2001. A Network Flow Model for Playlist Generation. In *Proc. ICME*. 445–448.

Klaus-Dieter Althoff. 2001. Case-Based Reasoning. *Handbook on Software Engineering and Knowledge Engineering* 1 (2001), 549–588.

Andreja Andric and Goffredo Haus. 2005. Estimating Quality of Playlists by Sight. In *Proc. AXMEDIS*. 68–74.

Jean-Julien Aucouturier and François Pachet. 2002a. Finding Songs That Sound The Same. In *Proc. MPCA*. 1–8.

Jean-Julien Aucouturier and François Pachet. 2002b. Music Similarity Measures: What's the Use?. In *Proc. ISMIR*. 157–163.

Jean-Julien Aucouturier and François Pachet. 2002c. Scaling Up Music Playlist Generation. In *Proc. ICME*, Vol. 1. 105–108.

Claudio Baccigalupo and Enric Plaza. 2006. Case-based Sequential Ordering of Songs for Playlist Recommendation. In *Proc. ECCBR*. 286–300.

Claudio Baccigalupo and Enric Plaza. 2007. A Case-Based Song Scheduler for Group Customised Radio. In *Case-Based Reasoning Research and Development*. 433–448.

Claudio Giovanni Baccigalupo. 2009. *Poolcasting: An Intelligent Technique to Customise Musical Programmes for Their Audience*. Ph.D. Dissertation. Universitat Autònoma de Barcelona.

Wietse Balkema and Ferdi van der Heijden. 2010. Music Playlist Generation by Assimilating GMMs into SOMs. *Pattern Recognition Letters* 31, 11 (2010), 1396–1402.

Luke Barrington, Reid Oda, and Gert RG Lanckriet. 2009. Smarter than Genius? Human Evaluation of Music Recommender Systems. In *Proc. ISMIR*. 357–362.

David Baskerville and Tim Baskerville. 2010. *Music Business Handbook and Career Guide*. Sage.

Dominikus Baur, Sebastian Boring, and Andreas Butz. 2010. Rush: Repeated Recommendations on Mobile Devices. In *Proc. IUI*. 91–100.

Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. 2011. The Million Song Dataset. In *Proc. ISMIR*. 591–596.

Jacob T. Biehl, Piotr D. Adamczyk, and Brian P. Bailey. 2006. DJogger: A Mobile Dynamic Music Device. In *Proc. CHI*. 556–561.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent Dirichlet Allocation. *the Journal of Machine Learning research* 3 (2003), 993–1022.

Thomas L Blum, Douglas F Keislar, James A Wheaton, and Erling H Wold. 1999. Method and Article of Manufacture for Content-Based Analysis, Storage, Retrieval, and Segmentation of Audio Information. (1999). US Patent 5,918,223.

Dmitry Bogdanov and Perfecto Herrera. 2011. How Much Metadata Do We Need in Music Recommendation? A Subjective Evaluation Using Preference Sets. In *Proc. ISMIR*. 97–102.

Geoffray Bonnin and Dietmar Jannach. 2013a. A Comparison of Playlist Generation Strategies for Music Recommendation and a New Baseline Scheme. In *Proc. ITWP*. 16–23.

Geoffray Bonnin and Dietmar Jannach. 2013b. Evaluating the Quality of Playlists Based on Hand-Crafted Samples. In *Proc. ISMIR*. 263–268.

Klaas Bosteels, Elias Pampalk, and Etienne E Kerre. 2009. Evaluating and Analysing Dynamic Playlist Generation Heuristics Using Radio Logs and Fuzzy Set Theory. In *Proc. ISMIR*. 351–356.

Karan Kumar Budhraja, Ashutosh Singh, Gaurav Dubey, and Arun Khosla. 2012. Probability Based Playlist Generation Based on Music Similarity and User Customization. In *Proc. NCCCS*. 1–5.

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

Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. 1995. A Limited Memory Algorithm for Bound Constrained Optimization. *SIAM Journal on Scientific Computing* 16, 5 (1995), 1190–1208.

Rui Cai, Chao Zhang, Lei Zhang, and Wei-Ying Ma. 2007. Scalable Music Recommendation by Search. In *Proc. Multimedia*. 1065–1074.

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

Òscar Celma and Xavier Serra. 2008. FOAFing the music: Bridging the Semantic Gap in Music Recommendation. *Web Semantics: Science, Services and Agents on the World Wide Web* 6, 4 (2008), 250–256.

Zeina Chedrawy and Syed Sibte Raza Abidi. 2009. A Web Recommender System for Recommending, Predicting and Personalizing Music Playlists. In *Proc. WISE*. 335–342.

Ching-Wei Chen, Kyogu Lee, and Ho-Hsiang Wu. 2009. Towards a Class-Based Representation of Perceptual Tempo for Music Retrieval. In *Proc. ICMLA*. 602–607.

Shuo Chen, Josh L. Moore, Douglas Turnbull, and Thorsten Joachims. 2012. Playlist Prediction via Metric Embedding. In *Proc. KDD*. 714–722.

Luca Chiarandini, Massimiliano Zanoni, and Augusto Sarti. 2011. A System for Dynamic Playlist Generation Driven by Multimodal Control Signals and Descriptors. In *Proc. MMSP*. 1–6.

Dave Cliff. 2006. hpDJ: An automated DJ with floorshow feedback. In *Consuming Music Together*. 241–264.

Filipe Coelho, José Devezas, and Cristina Ribeiro. 2013. Large-Scale Crossmedia Retrieval for Playlist Generation and Song Discovery. In *Proc. OAIR*. 61–64.

Michel Crampes, Jean Villerd, Andrew Emery, and Sylvie Ranwez. 2007. Automatic Playlist Composition in a Dynamic Music Landscape. In *Proc. SADPI*. 15–20.

Paolo Cremonesi, Franca Garzotto, Sara Negro, Alessandro Vittorio Papadopoulos, and Roberto Turrin. 2011. Looking for "Good" Recommendations: A Comparative Evaluation of Recommender Systems. In *Proc. INTERACT*. 152–168.

F. Crestani, M. Lalmas, and C.J. van Rijsbergen. 1998. *Information Retrieval: Uncertainty and Logics : Advanced Models for the Representation and Retrieval of Information*. Springer US.

S. Cunningham, D. Bainbridge, and A. Falconer. 2006. 'More of an Art than a Science': Supporting the Creation of Playlists and Mixes. In *Proc. ISMIR*. 240–245.

Ricardo Dias and Manuel J. Fonseca. 2010. MuVis: an Application for Interactive Exploration of Large Music Collections. In *Proc. MM*. 1043–1046.

Markus Dopler, Markus Schedl, Tim Pohle, and Peter Knees. 2008. Accessing Music Collections Via Representative Cluster Prototypes in a Hierarchical Organization Scheme. In *Proc. ISMIR*. 179–184.

J. S. Downie. 2003. Music Information Retrieval. *Annual Review of Information Science and Technology* 37, 1 (2003), 295–340.

Robert B Ekelund, George S Ford, and Thomas Koutsky. 2000. Market Power in Radio Markets: An Empirical Analysis of Local and National Concentration. *The Journal of Law and Economics* 43, 1 (2000), 157–184.

Benjamin Fields. 2011. *Contextualize Your Listening: the Playlist as Recommendation Engine*. Ph.D. Dissertation. Department of Computing Goldsmiths, University of London.

Benjamin Fields, Christophe Rhodes, Michael Casey, and Kurt Jacobson. 2008. Social Playlists and Bottleneck Measurements: Exploiting Musician Social Graphs Using Content-Based Dissimilarity and Pairwise Maximum Flow Values. In *Proc. ISMIR*. 559–564.

Arthur Flexer, Dominik Schnitzer, Martin Gasser, and Gerhard Widmer. 2008. Playlist Generation Using Start and End Songs. In *Proc. ISMIR*. 173–178.

D Gartner, Florian Kraft, and Thomas Schaaf. 2007. An Adaptive Distance Measure for Similarity Based Playlist Generation. In *Proc. ICASSP*. 229–232.

Arthur Germain and Jacob Chakareski. 2013. Spotify Me: Facebook-Assisted Automatic Playlist Generation. In *Proc. MMSP*. 25–28.

Dan Grabham. 2011. Last.fm: 'Free on-demand not successful for us'. *TechRadar* (2011).

Maarten Grachten, Markus Schedl, Tim Pohle, and Gerhard Widmer. 2009. The ISMIR Cloud: A Decade of ISMIR Conferences at Your Fingertips. In *Proc. ISMIR*. 63–68.

Jiawei Han and Micheline Kamber. 2006. *Data Mining: Concepts and Techniques (Second Edition)*. Morgan Kaufmann.

Derek L. Hansen and Jennifer Golbeck. 2009. Mixing It Up: Recommending Collections of Items. In *Proc. CHI*. 1217–1226.

Negar Hariri, Bamshad Mobasher, and Robin Burke. 2012. Context-Aware Music Recommendation Based on Latent Topic Sequential Patterns. In *Proc. RecSys*. 131–138.

Negar Hariri, Bamshad Mobasher, and Robin Burke. 2013. Personalized Text-Based Music Retrieval. In *Proc. ITWP*. 24–30.

Pitoyo Hartono and Ryo Yoshitake. 2013. Automatic Playlist Generation from Self-Organizing Music Map. *Journal of Signal Processing* 17, 1 (2013), 11–19.

David Hauger, Johannes Kepler, Markus Schedl, Andrej Košir, and Marko Tkalcic. 2013. The Million Musical Tweets Dataset: What Can We Learn From Microblogs. In *Proc. ISMIR*. 189–194.

David B Hauver and James C French. 2001. Flycasting: Using Collaborative Filtering to Generate a Playlist for Online Radio. In *Proc. WEDELMUSIC*. 123–130.

Otmar Hilliges, Phillipp Holzer, Rene Klüber, and Andreas Butz. 2006. AudioRadar: A Metaphorical Visualization for the Navigation of Large Music Collections. In *Proc. Smart Graphics*. 82–92.

Thomas Hornung, Cai-Nicolas Ziegler, Simon Franz, Martin Przyjaciel-Zablocki, Alexander Schatzle, and Georg Lausen. 2013. Evaluating Hybrid Music Recommender Systems. In *Proc. WI-IAT*. 57–64.

Jia-Lien Hsu and Shuk-Chun Chung. 2011. Constraint-Based Playlist Generation by Applying Genetic Algorithm. In *Proc. SMC*. 1417–1422.

Yajie Hu and Mitsunori Ogihara. 2011. NextOne Player: A Music Recommendation System Based on User Behavior. In *Proc. ISMIR*. 103–108.

Dietmar Jannach, Lukas Lerche, Fatih Gedikli, and Geoffray Bonnin. 2013. What Recommenders Recommend–an Analysis of Accuracy, Popularity, and Sales Diversity Effects. In *User Modeling, Adaptation, and Personalization*. 25–37.

Dietmar Jannach, Markus Zanker, Mouzhi Ge, and Marian Gröning. 2012. Recommender Systems in Computer Science and Information Systems–A Landscape of Research. In *Proc. EC-Web*. 76–87.

Claus Aage Jensen, Ester M Mungure, Torben Bach Pedersen, and Kenneth Sorensen. 2007. A Data and Query Model for Dynamic Playlist Generation. In *Proc. ICDE*. 65–74.

D. Jurafsky and J.H. Martin. 2009. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Pearson Prentice Hall.

Antti Jylhä, Stefania Serafin, and Cumhur Erkut. 2012. Rhythmic Walking Interactions with Auditory Feedback: an Exploratory Study. In *Proc. AM*. 68–75.

Mohsen Kamalzadeh, Dominikus Baur, and Torsten Möller. 2012. A Survey on Music Listening and Management Behaviours. In *Proc. ISMIR*. 373–378.

Marius Kaminskas and Francesco Ricci. 2011. Location-Adapted Music Recommendation Using Tags. In *Proc. UMAP*. 183–194.

Peter Knees, Tim Pohle, Markus Schedl, and Gerhard Widmer. 2006. Combining Audio-based Similarity with Web-based Data to Accelerate Automatic Music Playlist Generation. In *Proc. MIR*. 147–154.

Paul B Lamere. 2012. I've got 10 million songs in my pocket: now what?. In *Proc. RecSys*. 207–208.

Aristomenis S Lampropoulos, Dionysios N Sotiropoulos, and George A Tsihrintzis. 2012. Evaluation of a Cascade Hybrid Recommendation as a Combination of One-Class Classification and Collaborative Filtering. In *Proc. ICTAI*. 674–681.

Jin Ha Lee, Bobby Bare, and Gary Meek. 2011. How Similar Is Too Similar?: Exploring Users' Perceptions of Similarity in Playlist Evaluation. In *Proc. ISMIR*. 109–114.

Kyogu Lee and Minsu Cho. 2011. Mood Classfication from Musical Audio Using User Group-Dependent Models. In *Proc. ICMLA*. 130–135.

Arto Lehtiniemi and Jarno Seppänen. 2007. Evaluation of Automatic Mobile Playlist Generator. In *Proc. MC*. 452–459.

Ning-Han Liu, Szu-Wei Lai, Chien-Yi Chen, and Shu-Ju Hsieh. 2009. Adaptive Music Recommendation Based on User Behavior in Time Slot. *Computer Science and Network Security* 9, 2 (2009), 219–227.

Beth Logan. 2002. Content-Based Playlist Generation: Exploratory Experiments. In *Proc. ISMIR*. 295–296.

Beth Logan. 2004. Music Recommendation from Song Sets. In *Proc. ISMIR*. 425–428.

Beth Logan and Ariel Salomon. 2001. A Music Similarity Function Based on Signal Analysis. In *Proc. ICME*. 190–193.

François Maillet, Douglas Eck, Guillaume Desjardins, and Paul Lamere. 2009. Steerable Playlist Generation by Learning Song Similarity from Radio Station Playlists. In *Proc. ISMIR*. 345–350.

Niitsuma Masahiro, Hiroshi Takaesu, Hazuki Demachi, Masaki Oono, and Hiroaki Saito. 2008. Development of an Automatic Music Selection System Based on Runner's Step Frequency. In *Proc. ISMIR*. 193–198.

J McDermott. 2012. Auditory Preferences and Aesthetics: Music, Voices, and Everyday Sounds. *Neuroscience of Performance and Choice* (2012), 227–256.

B. McFee, T. Bertin-Mahieux, D. Ellis, and G. Lanckriet. 2012. The Million Song Dataset Challenge. In *Proc. AdMIRe*.

Brian McFee and Gert RG Lanckriet. 2011. The Natural Language of Playlists. In *Proc. ISMIR*. 537–542.

Brian McFee and Gert RG Lanckriet. 2012. Hypergraph Models of Playlist Dialects. In *Proc. ISMIR*. 343–348.

Cory McKay. 2010. *Automatic Music Classification with jMIR*. Ph.D. Dissertation. McGill University.

Owen Craigie Meyers. 2007. *A Mood-Based Music Classification and Exploration System*. Master's thesis. Massachusetts Institute of Technology.

Bart Moens, Leon van Noorden, and Marc Leman. 2010. D-Jogger: Syncing Music With Walking. In *Proc. SMC*.

Joshua L Moore, Shuo Chen, Thorsten Joachims, and Douglas Turnbull. 2012. Learning to Embed Songs and Tags for Playlist Prediction. In *Proc. ISMIR*. 349–354.

Miki Nakagawa and Bamshad Mobasher. 2003. Impact of Site Characteristics on Recommendation Models Based on Association Rules and Sequential Patterns. In *Proc. ITWP*.

Nuria Oliver and Fernando Flores-Mangas. 2006. MPTrain: A Mobile, Music and Physiology-Based Personal Trainer. In *Proc. MobileHCI*. 21–28.

Nuria Oliver and Lucas Kreger-Stickles. 2006. PAPA: Physiology and Purpose-Aware Automatic Playlist Generation. In *Proc. ISMIR*, Vol. 2006. 393–396.

François Pachet. 2001. Metadata for Music and Sounds: The CUIDADO Project. In *Proc. CBMI*. 411–415.

François Pachet, Pierre Roy, and Daniel Cazaly. 2000. A Combinatorial Approach to Content-Based Music Selection. *Multimedia* 7, 1 (2000), 44–51.

Elias Pampalk, Tim Pohle, and Gerhard Widmer. 2005. Dynamic Playlist Generation Based on Skipping Behavior. In *Proc. ISMIR*. 634–637.

Steffen Pauws and Berry Eggen. 2002. PATS: Realization and User Evaluation of an Automatic Playlist Generator. In *Proc. ISMIR*. 222–231.

Steffen Pauws and Sander van de Wijdeven. 2005. User Evaluation of a New Interactive Playlist Generation Concept. In *Proc. ISMIR*. 638–643.

Steffen Pauws, Wim Verhaegh, and Mark Vossen. 2006. Fast Generation of Optimal Music Playlists using Local Search. In *Proc. ISMIR*. 138–143.

Steffen Pauws, Wim Verhaegh, and Mark Vossen. 2008. Music Playlist Generation by Adapted Simulated Annealing. *Information Sciences* 178, 3 (2008), 647–662.

Geoffroy Peeters and Joachim Flocon-Cholet. 2012. Perceptual Tempo Estimation Using GMM-Regression. In *Proc. MIRUM*. 45–50.

John C Platt, Christopher JC Burges, Steven Swenson, Christopher Weare, and Alice Zheng. 2001. Learning a Gaussian Process Prior for Automatically Generating Music Playlists. In *Proc. NIPS*. 1425–1432.

Tim Pohle, Peter Knees, Markus Schedl, Elias Pampalk, and Gerhard Widmer. 2007. Reinventing the Wheel: A Novel Approach to Music Player Interfaces. *Multimedia* 9, 3 (2007), 567–575.

Tim Pohle, Elias Pampalk, and Gerhard Widmer. 2005. Generating Similarity-Based Playlists using Traveling Salesman Algorithms. In *Proc. DAFx*. 220–225.

R. Ragno, C. J. C. Burges, and C. Herley. 2005. Inferring Similarity Between Music Objects with Application to Playlist Generation. In *Proc. MIR*. 73–80.

Sasank Reddy and Jeff Mascia. 2006. Lifetrak: Music In Tune With Your Life. In *Proc. HCM*. 25–34.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Proc. UAI*. 452–461.

Gordon Reynolds, Dan Barry, Ted Burke, and Eugene Coyle. 2008. Interacting With Large Music Collections: Towards the Use of Environmental Metadata. In *Proc. ICME*. 989–992.

Ronald Rosenfeld. 2000. Two Decades of Statistical Language Modeling: Where Do We Go From Here? *Proc. IEEE* 88, 8 (2000), 1270–1278.

Vegard Sandvold, Thomas Aussenac, Oscar Celma, and Perfecto Herrera. 2006. Good Vibrations: Music Discovery through Personal Musical Concepts. In *Proc. ISMIR*. 322–323.

Andy M Sarroff and Michael Casey. 2012. Modeling and Predicting Song Adjacencies In Commercial Albums. In *Proc. SMC*.

Ingo Schmädecke and Holger Blume. 2013. High Performance Hardware Architectures for Automated Music Classification. In *Algorithms from and for Nature and Life*. 539–547.

Malcolm Slaney and William White. 2006. Measuring Playlist Diversity for Recommendation Systems. In *Proc. AMCMM*. 77–82.

Malcolm Slaney and William White. 2007. Similarity Based on Rating Data. In *Proc. ISMIR*. 479–484.

George Tzanetakis. 2002. *Manipulation, Analysis and Retrieval Systems for Audio Signals*. Ph.D. Dissertation. Princeton University.

Rob van Gulik and Fabio Vignoli. 2005. Visual Playlist Generation on the Artist Map. In *Proc. ISMIR*. 520–523.

Igor Vatolkin, Mike Preuß, and Günter Rudolph. 2011. Multi-Objective Feature Selection in Music Genre and Style Recognition Tasks. In *Proc. GECCO*. 411–418.

Fabio Vignoli and Steffen Pauws. 2005. A Music Retrieval System Based on User Driven Similarity and Its Evaluation. In *Proc. ISMIR*. 272–279.

Hugues Vinet, Perfecto Herrera, and François Pachet. 2002. The CUIDADO Project. In *Proc. ISMIR*. 197–203.

Richard Y. Wang and Diane M. Strong. 1996. Beyond Accuracy: What Data Quality Means to Data Consumers. *Journal of Management Information Systems* 12, 4 (1996), 5–33.

Gary M. Weiss. 2004. Mining with Rarity: A Unifying Framework. *SIGKDD Explorations Newsletter* 6, 1 (2004), 7–19.

Erling Wold, Thom Blum, Douglas Keislar, and James Wheaten. 1996. Content-Based Classification, Search, and Retrieval of Audio. *Multimedia* 3, 3 (1996), 27–36.

Linxing Xiao, Lie Lu, Frank Seide, and Jie Zhou. 2009. Learning a Music Similarity Measure on Automatic Annotations with Application to Playlist Generation. In *Proc. ICASSP*. 1885–1888.

David Zax. 2011. The Echo Nest Makes Pandora Look Like a Transistor Radio. *Fast Company* (2011).

Elena Zheleva, John Guiver, Eduarda Mendes Rodrigues, and Nataša Milić-Frayling. 2010. Statistical Models of Music-Listening Sessions in Social Media. In *Proc. WWW*. 1019–1028.

APPENDIX

Examples for publicly available music datasets

Datasets provided by companies

Probably the most prominent source of music data provided by a commercial service is the database of last.fm, which contains information about 45 millions tracks [Grabham 2011] and 40 millions users. This includes track attributes, tags, number of listeners per track, play counts, artist information, album information, comments, top fans, user playlists, etc.

A similar source of music data is Musicbrainz (musicbrainz.org), which is thought as an encyclopedia where users can collaboratively collect information on music. Each submitted information is subject to a review and voting process by other users, leading to a highly reliable database. It contains track attributes, tags, artist information, album information, ratings, etc. but is smaller (15 million tracks) and less rich than the database of last.fm.

Finally, another publicly available music database is provided by The Echo Nest (the.echonest.com), which contains information about 30 million tracks [Zax 2011]. One particularity of this database is that it contains various types of information that is inferred from the audio signal, such as the danceability, the energy, the loudness, etc.

Datasets provided by research groups

Some smaller databases were released by researchers in order to allow others to make experiments on them and compare their results. One of the most prominent ones is the Million Song Dataset (MSD) [Bertin-Mahieux et al. 2011]. The dataset consists of one million tracks and corresponding meta-data. Each track is described by 55 attributes, some of which taken from The Echo Nest and Musicbrainz. Other smaller similar datasets also exist, as for instance the dataset of the CUIDADO project [Pachet 2001; Vinet et al. 2002].

A dataset of hand-crafted playlists was made available by [McFee and Lanckriet 2012]. The dataset contains all playlists of the Website Art of the Mix (http://www.artofthemix.org) created before June 2011. It contains about 100,000 playlists created by about 16,000 users. Each playlist contains artist names, track names, creator name, a categorical label (e.g., “Reggae”) and the identifiers of the tracks that could be found in the Million Song Dataset.

Computational complexity analysis

Table V. Computational complexity analysis

Algorithm	Analysis
kNN	The major drawback of this algorithm is its complexity. Because each tested playlist history corresponds to a new playlist, no similarity can be calculated in advance in an offline process. This means that all the playlists in the pool of available playlists P have to be scanned for each track selection. The time complexity for the computation of each similarity value between h and each playlist p can be reduced to $\mathcal{O}(\|h\| + \|p\|)$ using a direct matching algorithm. Assuming that the playlists have a maximum size of M, the complexity for the computation of the neighborhood of h is $\mathcal{O}(\|P\| \cdot (\|h\| + M))$. If all the tracks in the set of tracks R of the database are considered, the overall complexity is $\mathcal{O}(\|P\| \cdot (\|h\| + M) + k * \|R\|)$. In practice, the running time can be reduced by considering only the tracks contained in the resulting neighborhood of h. The space complexity depends on the space required for storing the pool of playlists. Speaking in terms of number of tracks, if the maximum size of playlists is M and the number of playlists $\|P\|$, the space complexity is $\mathcal{O}(M \cdot \|P\|)$.
Frequent patterns	By extracting a limited set of relevant patterns in an offline process, this approach theoretically allows a lower time and space complexity than that of the kNN approach. Both association rules and sequential patterns consider sub-histories of variable length inside a sliding window of size w. The number of sub-histories considered is thus: $$\sum_{n=1}^{w-1} \binom{w-1}{n} = 2^{w-1} + 1$$ Determining the rules that match each sub-history in the model can be done in both cases in $\mathcal{O}(n)$ with n the size of the patterns extracted [Nakagawa and Mobasher 2003]. The overall time complexity is thus: $$\sum_{n=1}^{w-1} \mathcal{O}(n) \cdot \binom{w-1}{n} \leq \mathcal{O}(w \cdot 2^w)$$ The space complexity of sequential patterns corresponds to the number of permutations with repetition of n elements from $\|R\|$. The space complexity is thus $\mathcal{O}(\|R\|^n)$. The space complexity of association rules corresponds to the number of combinations with repetition of n elements from $\|R\|$, which is less than $\|R\|^n$. In practice, long patterns are not frequent enough to be of any value. In our experiments, the maximum sizes that led to the best results were 2 and 3. The time complexity is then reduced to $\mathcal{O}(w)$ and $\mathcal{O}(w^2)$ and the space complexities to $\mathcal{O}(\|R\|^2)$ and $\mathcal{O}(\|R\|^3)$, respectively. Finally, the selection of the most relevant patterns in the offline mining process in practice means that the number of elements to scan is much lower than the theoretical complexity bounds.
SAGH, CAGH	The time complexity of the SAGH algorithm is particularly low. Given a playlist history h, the algorithm only has to retrieve the corresponding artists. The complexity of this task is $\mathcal{O}(\|h\|)$. Retrieving the greatest hits of a given artist can be done in $\mathcal{O}(1)$. The overall time complexity for the computation of the score of all tracks is thus $\mathcal{O}(\|h\|)$, which is very low, especially when considering that playlists typically have average sizes between 7 and 20 in our datasets. The time complexity of the CAGH is a little higher because the algorithm also considers similarities between artists. As these similarities can be computed offline and stored, the algorithm only has to retrieve the similar artists for each artist in A_h from the database, and then to multiply the greatest hit counts with the calculated similarity values. The time complexity is thus $\mathcal{O}(\|h\| \cdot \|A\| \cdot \|R\|)$, where A is the set of artists and R the set of resources. The space complexity of the SAGH algorithm corresponds to the set of greatest hits stored, i.e., $\mathcal{O}(\|A\| \cdot \gamma)$ where A is the set of artists and γ the maximum number of greatest hits stored per artist. The space complexity of the CAGH algorithm is higher as the similarities between the artists have to be stored. The space complexity is thus: $\mathcal{O}(\|A\| \cdot \gamma + \|A\|^2)$.