Music Recommendations*

Dietmar Jannach[1], Iman Kamehkhosh[2], and Geoffray Bonnin[3]

[1]AAU Klagenfurt, Austria
[2]TU Dortmund, Germany
[3]Loria, Nancy, France

Abstract

Today's online music services like Spotify provide their listeners with different types of music recommendations, e.g., in the form of weekly recommendations or personalized radio stations. Such recommendations are often based, at least in parts, on collaborative filtering techniques. In this chapter, we first review the different types of music recommendations that can be found in practice and discuss the specific challenges of the domain. Next, we discuss technical approaches for the problems of music discovery and next-track recommendation in more depth, with a specific focus on their practical application at Spotify. Finally, we further elaborate on open challenges in the field and revisit the specific problems of evaluating music recommendation systems in academic environments.

1 Introduction

Music was one of the first application fields of collaborative filtering (CF) recommender systems. The *Ringo* system presented in Shardanand and Maes (1995) went online as early as 1994 and was designed to recommend albums and musical artists to users, initially as an email-based service. It was based on explicit rating information provided by users for artists to construct preference profiles. Recommendations in that system were then made based on a user-to-user or item-to-item neighborhood scheme as described in Chapter 1 and sent via email to Ringo's users.

Today, more than twenty years later, the social web has led to new dimensions of social, "word of mouth" information filtering. At the same time, the way we listen to music and how we discover new artists or albums has dramatically changed. Millions of music tracks are nowadays available to us instantaneously through various online music streaming services. As a result, almost all of today's major online music services, including Spotify or the services by Google or Microsoft, provide some form of music recommendation functionality.

1.1 Music Recommendation Tasks

The Ringo system supported three main tasks: (i) suggest artists/albums, (ii) list artists/albums that the user will most probably dislike, and (iii) predict the rating of a user for an album. At the time when Ringo was online, users could consider the recommendations the next time they visited a music store or when they placed a mail order at a catalog company.

Of the three tasks supported by the Ringo system, today mainly the artist and album recommendation services are common on online music platforms. However, the recent possibility

*In: Berkovsky et al., Collaborative Recommendations: Algorithms, Practical Challenges and Applications, World Scientific, 2019 to provide recommendations and play music instantaneously opened new opportunities for other recommendation scenarios. Spotify, one of today's market leaders in the online music streaming industry, as of 2018, provides recommendations through a number of features of their platform and apps. Specifically, they for example support *discovery* through personalized recommendations Johnson (2014). They furthermore generate user-specific playlists ("mix tapes") for listeners on a weekly basis and provide a "release radar" to point users to newly released tracks that might be interesting to them. In addition to these features, non-personalized recommendations, e.g., of trending tracks, are common on most music platforms.

While some types of music recommendations are nowadays already common on music-related sites and apps, some additional scenarios for music recommendation have been explored in the literature in the past. Table 1 provides an overview of common recommendation scenarios. We can categorize the different recommendation tasks into different groups.

- First, recommendations can be *non-personalized* and simply consist of currently trending tracks, artists, albums, concerts, etc. Another common (non-personalized) way of pointing users to something interesting is to provide them with *curated* playlists, which represents a convenient way for users to browse the catalog Johnson and Newett (2014).

- Second, recommendations can be *contextualized*, but *non-personalized*. On Last.fm, for example, virtually endless radio stations can be created based on a seed track or seed artist. In the research literature, a number of approaches to support users during manual playlist construction were proposed, where the system's task is to recommend additional tracks given a playlist beginning or a set of user-specified constraints.
- The third category consists of *personalized*, but *non-contextual* recommendations. Spotify's Discover Weekly feature is an example of such a functionality, where the system provides a set of recommended tracks based on the user's observed behavior.
- The final category comprises approaches that are both *personalized and contextualized*. An example would be a system that recommends items during playlist construction where these items both match the current playlist and the user's general tastes. Approaches in that category can mostly be found in the academic literature.

In this chapter, our main focus will be on approaches that are either contextualized, personalized, or both. We will specifically discuss collaborative filtering techniques and hybrid approaches that combine collaborative information with additional data, e.g., the musical features of the tracks.

1.2 Specific Challenges of Music Recommendation

From a computational perspective, some of the tasks listed in Table 1 seem quite similar to recommendation tasks in other domains, like e-commerce. In particular the *personalized and non-contextualized* recommendation scenarios can in principle be addressed with collaborative filtering approaches that are designed for relevance prediction and learning-to-rank scenarios, where the final goal is to create a ranked list of objects that are supposed to be *generally relevant* for a user. We will discuss a variant of such a standard collaborative filtering technique, as used by Spotify, later in Section 2. Similarly, the problem of providing a virtually endless playlist given the user's recently played tracks can be found in a comparable form as a *session-based recommendation* scenario in e-commerce Hidasi *et al.* (2016a); Jannach and Ludewig (2017); Quadrana *et al.* (2018). We will discuss the specifics of the problem setting for music recommendation in a later section as well.

Table 1: Examples of Music Recommendation Tasks.

	Non-personalized	**Personalized**
Non-contextualized	• Trending List: Provide a list of currently trending (or currently being played) items, e.g., tracks, albums, artists, concerts etc. Often used as a baseline for experiments in the research literature, e.g., Chen *et al.* (2016); Pálovics *et al.* (2014); Vasile *et al.* (2016). • Similar Objects: Find similar tracks or artists, available, e.g., on Spotify. This type of recommendation can often be found in the Music Information Retrieval literature, e.g., Germain and Chakareski (2013); Hartono and Yoshitake (2013); Moore *et al.* (2012). • Curated Playlists: Help users discover things through curated (editorial) playlists, e.g., on 8tracks.com. • Broadcasting Radios: Usually made by professional disc jockeys. Such playlists often contain popular tracks, and are often targeting specific audiences Ekelund *et al.* (2000).	• Track, Artist Discovery: Help users to find something new that matches their general preferences, as implemented, e.g., by Spotify in the "Discover Weekly" and "Release Radar" features Johnson (2014). • Album Discovery: Recommend albums to listen to. Such a functionality can be found on general e-commerce sites like Amazon.com as well. • Enjoyment Prediction: Provide an assessment if the user will like a certain track, artist or album; also, create a list of things that the user will presumably *not* like. • Static Playlist Generation: Generate a personalized playlist based on user tastes; like Spotify's "Mix Tape" feature. • Personalized Recommendation of Curated Playlists: Suggest hand-made playlists to users that are likely to generally match their taste, e.g., on Deezer. Rarely studied in the literature, see, e.g., Loni *et al.* (2016). • Recommendation of Radio Streams: Recommend broadcasting radio stations to users based on their profile and feedback. Also rarely studied, see, e.g., Moling *et al.* (2012).
Contextualized	• Virtual Radio Station: Create a virtually endless playlist, given a seed track or seed artist. To be found on Spotify, Deezer, Pandora, and other popular services, as well as in the research literature, see, e.g., Oliver and Flores-Mangas (2006); Cliff (2006); Moens *et al.* (2010); Jylhä *et al.* (2012). • Playlist Construction Support: Generate a playlist based on seed tracks or other information regarding the current session, like the user's mood; or provide suggestions of tracks during manual playlist creation. • Contextualized Playlist Recommendation: Recommend a curated playlist based, e.g., on the time of the day, day of the week, or season.	• Personalized Radio (next-track recommendation): Generate a virtually endless radio based on the last played tracks, while possibly taking into account the user immediate feedback (e.g., "like", "skip", and "ban" actions). • Personalized Playlist Construction Support: Generate a playlist based on seed tracks or other information, like the user's mood and past preferences.

A number of aspects are however very specific to music recommendation, and some others are at least more relevant for music than for other application areas of recommender systems. These aspects relate both to technical and non-technical issues and include, among others, the following.

Catalog aspects: While larger e-commerce shops can easily have hundred thousands of catalog items, the number of recommendable items on Spotify, as of 2018, is over 35 million tracks. This can make the application in particular of academic approaches challenging. Furthermore, constantly new tracks are released and added to the catalog. And, at least for some musical genres, the freshness or recency of the recommendations might be an important quality factor to consider. Moreover, the meta-data of the tracks in the catalog can be very noisy and incomplete, and significant efforts might be required in order to clean it and infer missing information, in particular as a huge number of new tracks is released every year.

Preference information: Users of the Ringo system were asked to indicate their preferences for 125 artists (popular ones and random ones). While also some of today's systems (such as Microsoft Groove) ask users to provide an initial set of preferences regarding artists and genres, music recommenders often have to rely on mostly implicit preference signals in terms of listening logs, sometimes in combination with explicit like statements or "skip" actions. Besides the problem of correctly interpreting very large amounts of implicit feedback, an additional challenge in that context is that preferences can change over time.

Repeated recommendations: Many recommendation algorithms, and in particular those that are based on the matrix completion problem abstraction, aim to predict the relevance of *unseen* items. In the music domain, repeatedly listening to the same tracks is however common. If such repeated consumptions should be supported, algorithmic approaches have to be able both to decide *which* tracks to recommend repeatedly and *when* to recommend these tracks.

Immediate consumption and feedback: Differently from many e-commerce domains, the recommendations provided on a music streaming service can be immediately "consumed" by the listeners, e.g., using a personalized radio station. A main challenge in that context is that the system should be able to provide the user with a means to "correct" recommendations or give feedback (e.g., in terms of a like or dislike button). Moreover, this feedback should be immediately taken into account in the recommendation process.

Mainstream might not be enough: In some sense, music is "more niche" than movies Johnson (2014). While in movie recommendation there are many blockbusters that are safe to recommend to a major fraction of the users, there are many musical genres which have their specific audiences (like jazz, classical music, or pop), and recommending generally popular items might easily lead to a bad user experience.

Context-dependence and time variance: Which music we want to listen to can be highly context-dependent. The relevant contextual factors can include, for example, the user's mood, the time of the day, or if the user listens to the music alone or as part of a group. Being able to capture and consider these contextual factors can be crucial for the quality perception and acceptance of a recommender.

Purposes of music listening: One related specificity of music is the fact that one often listens to music with a very specific purpose in mind: create a particular ambiance for a party, getting some motivation to wash the dishes, enhance the experience of reading a good book, getting relaxed before going to bed, etc. This means that the recommended items not only have to fit the current context, but also fit the purpose of the user.

Musical taste and stated preferences can be socially influenced: Which music we like and listen to is not only affected by our own mood, it can also be substantially affected by our social environment ("social bonding") and/or trends in the community as a whole. For some scenarios it can therefore be particularly helpful to consider a user's social environment and corresponding behavior in the past in the recommendation process. At the same time, when users share their tastes and preferences on social networks, it is not always clear if people actually listen to what they publicly "like" or if they merely use their public profiles to create a desired image of themselves.

1.3 Chapter Outline

In the remainder of this chapter, we will first discuss selected algorithmic approaches for music recommendation problems. Specifically, we will first discuss the application of matrix factorization techniques for the Discover Weekly feature of Spotify and will then review recent approaches of adaptive playlist generation (next-track music recommendation). Afterwards, we will outline open challenges both from a practical and academic perspective. We will then report how a music recommendation service is deployed and tested in industrial environments, again based on publicly available information about Spotify's solution. The final sections of the paper will be devoted to questions of how to evaluate music recommenders in practice and which resources are available for researchers in academic environments. The chapter ends with a summary of the lessons learned and open challenges.

2 Computational Tasks and Algorithms

Various algorithmic approaches have been proposed over the years for music-related recommendation scenarios. Three types of approaches generally exist: (1) collaborative techniques, i.e., recommending music liked by similar users, (2) content-based techniques, i.e., recommending music whose content (musical features, lyrics, etc.) is similar to the content of the tracks the user liked in the past, and (3) hybrid techniques, i.e., combining both previous approaches. Content-based approaches are mostly studied in the field of Music Information Retrieval (MIR). Correspondingly, the MIR literature often focuses on the problem of deriving such features from the low-level musical signal.[1] This chapter will not cover these aspects and will mostly focus on collaborative and hybrid techniques.

Companies usually do not publicly reveal the details of how they generate their recommendations or on which data they are based. Some discussions about the inner workings of the used machinery are sometimes revealed in public presentations. In many cases, presenters from industry report that they use a variety of algorithmic approaches for the different recommendation tasks. The streaming service *Pandora*, for example, in addition to its unique database of manually annotated musical tracks[2] which is used for *content-based* recommendations, relies also on collaborative techniques Bieschke (2014).

2.1 Implicit Matrix Factorization for Discovery and Item Search

In this section, we will briefly review how collaborative filtering was – among other techniques – applied at Spotify based on public presentations around the year 2014 Johnson and Newett (2014); Johnson (2014). In more recent presentations, such as Steck *et al.* (2015), the authors report that Spotify uses an ensemble of different techniques (including NLP models and Recurrent Neural Networks) as well as explicit feedback signals (thumbs up / down), and also audio features for certain recommendation tasks.

2.1.1 Distributed Matrix Factorization based on Listening Logs

At its core, the algorithm that was used at Spotify for its discovery feature, is based on a standard user-item rating matrix, where users are rows and the columns represent tracks. There are,

[1]The various aspects of automated music data analysis are discussed, e.g., in Weihs *et al.* (2016).

[2]Music Genome Project `https://en.wikipedia.org/wiki/Music_Genome_Project`

however, a number of specific aspects to be addressed in this domain, in particular the following two:

1. In comparison to the well-researched movie recommendation domain, the number of items is much larger and there are more than 35 million songs that can potentially be recommended. Also, the number of users is substantial, in particular when compared to public datasets that are used in academic research.

2. The entries in the matrix are often not explicit item ratings, but are based on the users' listening histories. As users often listen to tracks many times, the resulting play counts can be used as an implicit preference signal. Exploiting play counts instead of explicit ratings can actually lead to more accurate recommendations Jawaheer *et al.* (2010).

Let us recall a common formulation of the optimization problem for matrix factorization techniques as described, e.g., in Koren (2008).

$$\min_{q*,p*} \sum_{(u,i)\in K} (r_{ui} - \mu - b_u - b_i - q_i^T p_u)^2 + \lambda(\|q_i\|^2 + \|p_i\|^2 + b_u^2 + b_i^2) \tag{1}$$

Here, the goal is to minimize the squared prediction error for all (u,i) pairs in the training set (K), where the rating prediction r_{ui} for a user u and an item i is based on the rating average μ, some user and item bias factors b_u and b_i, and the user and item latent factors q_i and p_i; see also Chapter 2.

In the problem encoding of what is called "implicit matrix factorization" at Spotify, as proposed in Hu *et al.* (2008), the entries in the input matrix are zeros and ones, where a one in a cell indicates that a user has streamed a track at least one time. Multiple streaming events for the same track are therefore not considered in the encoding. However, the assumed higher preference for a track that was played multiple times is captured in the optimization function. Instead of optimizing the root mean squared error (RMSE), a weighted version of the RMSE is optimized, where a certain weight factor c_{ui} is used, which is based on the stream counts for a given user and track. Ignoring the average rating μ, the optimization function is therefore defined as Hu *et al.* (2008); Johnson (2014):

$$\min_{q*,p*} \sum_{(u,i)\in K} c_{ui}(r_{ui} - b_u - b_i - q_i^T p_u)^2 + \lambda(\|q_i\|^2 + \|p_i\|^2 + b_u^2 + b_i^2) \tag{2}$$

Details of how the weight function is calculated are not revealed. To find the optimal parameters, the Alternating Least Squares (ALS) method is applied. Due to the huge amount of required computations, a distributed computing architecture based on the map-reduce scheme (using Hadoop[3]) was utilized at that time. The exact details of the distributed computations are not available. Nevertheless, distributed ALS methods as described, e.g., in Zhou *et al.* (2008) are nowadays available as off-the-shelf components, e.g., within the Apache Mahout[4] software. The required infrastructure, however, seems to be substantial and Steck *et al.* (2015) report that Spotify, as of 2015, used a Hadoop cluster consisting of 1,000 nodes and collected one terabyte of user data each day.

Once the latent user and item vectors are computed, they can be used for the two following tasks:

1. *Relevance prediction*: As usual, the relevance of an item can be determined by computing the dot product of the user and the item vectors.

[3]`http://hadoop.apache.org/`
[4]`http://mahout.apache.org/`

2. *Item or user similarity assessment*: Two items or users can be compared using their coordinates in the latent factor space by computing the cosine similarity of the item and vectors, respectively.

2.1.2 Finding Similar Objects with Approximate Nearest Neighbors

For this latter task – finding similar objects, which is a key non-personalized functionality of Spotify – the problem is again that there are many users and items. Finding the exact set of the most similar objects ("neighbors") is therefore computationally challenging. As a consequence, an "Approximate Nearest Neighbor" method called *Annoy* was used Bernhardsson (2015).

Technically, the algorithm works by recursively splitting up the set of objects into two classes. In each step, two random points are chosen and the hyperplane that is equidistant to the points is used to split the data. This procedure is recursively repeated until a certain stopping criterion is met, leading to a binary tree where each object is assigned to exactly one leaf nodes. As a stopping criterion, one could for example define that each leaf node has at most K objects assigned.

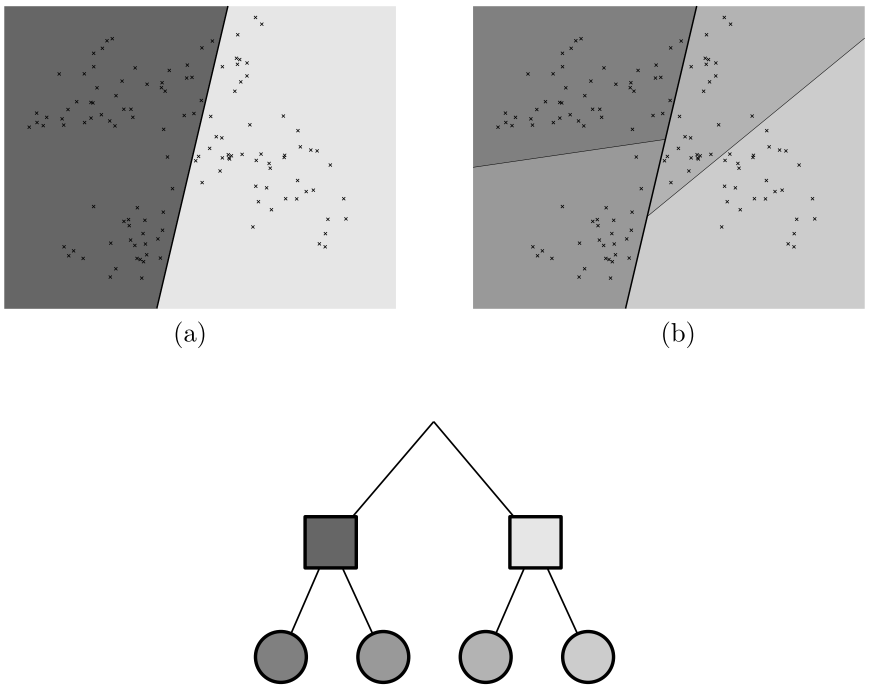


Figure 1: Building the binary tree in *Annoy*, adapted from Bernhardsson (2015). The set of points is first split into two halves according to the first selected hyperplane (a). Both halves are then split again using the same principle (b).

Figure 1 shows an example of such a tree, which has the property that most objects in the same leaf node have a higher probability to be similar to each other than to objects of another leaf node. When looking for the neighbors of a certain user or item, one can walk down the tree from the root node by assessing on which side of the separating hyperplane the input item will be. Such a simple algorithm, however, has its limitations. One can, for example, easily end up with a situation where there are (a) only very few objects in the resulting leaf node and (b) some nodes are actually close but not part of the leaf node.

Two techniques are proposed to deal with these potential limitations:

1. An algorithm is used that explores both child nodes when a certain closeness threshold is met.
2. Instead of building only one decomposition tree, multiple trees are randomly generated and the set of neighborhood candidates is determined by collecting the elements of all leaf nodes for the given target items.

For the resulting set of candidate neighbors, the similarity values are computed and the N most similar ones are returned. While some of the true nearest neighbors might be missed by the procedure, the neighborhoods can be computed fast. Using the different algorithm parameters (e.g., the number of trees used), one can furthermore balance accuracy and computational costs.

Generally, *Annoy* is not the first method that combines clustering and nearest-neighbor search. Different other approximate techniques were proposed in the literature before, for instance, ball-trees Omohundro (1989). In principle, the proposed technique is only one of several possible ways of constructing so-called k-d trees, i.e., trees that partition points in the space using hyperplanes. According to the authors of *Annoy*[5], the method however has the advantage of having a small memory footprint and that it is engineered to be used in a distributed environment as index files can be shared between processes.

2.2 Adaptive Playlist Generation

The matrix factorization approach presented in the previous section is mainly designed to help users discover items that are assumed to be *generally* relevant or interesting to them. However, as discussed in Chapter 1, users of a recommendation service often have a specific intent or goal when they visit a website or use an app. In the music domain, the goal or intention of a user might be influenced by his or her current mood or contextual situation (e.g., being at a party, or doing sports). One common functionality of music streaming services therefore is to provide users with a means to play a virtual endless list of suitable tracks based only on some limited amounts of initial input.

2.2.1 Spotify's Adaptive Radio

On some platforms this functionality is called "Radio". Given, e.g., an artist or a track as an input, a playlist is automatically created by the service. Often, such playlists are based on tracks by related artists or by tracks that are in some sense similar to the seed track. In some cases, such radio stations are non-personalized, i.e., with the same input, every user receives more or less the same set of recommendations. In principle, however, the selection of tracks could be personalized as well, in case the listening history or some other preferences information about the current user is known.

Since in this scenario the system's recommendations are immediately consumed in the order determined by the system, the users should be enabled to give feedback on the recommendations and this feedback should be immediately processed by the recommender. Spotify's *Radio* feature supports both of these functionalities, personalized playlists and immediate feedback, in some form.

Bernhardsson (2013) summarizes the main principle of how Spotify optimizes their "artist radio" or "song radio" as sketched in Figure 2. The general idea is to use a number of different algorithms (which probably also use different optimization measures) and combine the recommendations in an ensemble with Gradient Boosted Decision Trees Steck *et al.* (2015), with explicit thumbs data and random negatives as an input for the optimization process. The outcome is a pool of tracks that can in principle be played on the current radio. The final ranking of the tracks is then done by computing scores for each possible next track. The following factors are taken into account. However, how the different aspects are combined, is not revealed in detail.

[5]See the documentation at `https://github.com/spotify/annoy/`

- The "global rank" of a track in the track pool, as determined by the algorithms;
- the estimated relevance of each track for a user based on the latent factor model;
- the estimated relevance to the user based on the given thumbs;
- the diversity of the tracks in the session in terms of artists and albums.

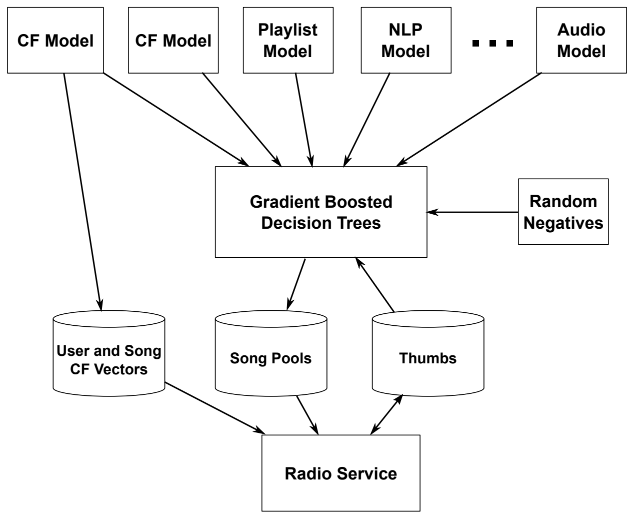


Figure 2: Conceptual Model of the Radio Service, adapted from Steck *et al.* (2015).

2.2.2 CF-based Next-Track Music Recommendation in the Literature

Academic research has been conducted on various forms of music recommendation. The addressed application scenarios for example include non-contextualized item (e.g., track, artist, or album) recommendation for discovery Bachrach *et al.* (2014), similar object retrieval Chen *et al.* (2016), the automated construction of finite-length playlists or playlists with some length bounds Pauws *et al.* (2008), as well as next-item recommendation (playlist continuation) Vasile *et al.* (2016).

In this section, we will focus on applying collaborative filtering techniques for next-item recommendation scenarios, i.e., on situations where the collective behavior of a larger user community is considered in the recommendation process. The general computational task in these settings is to compute a ranked list of tracks to be played next given some seed information like an artist or a start track. There are two main applications where such a functionality is needed. First, next-item recommendations can be the basis for a virtually endless radio station like Spotify's Radio feature. Second, this type of recommendations can be used to assist users when they manually create a finite-length playlist, as implemented, e.g., in Apple's iTunes Genius.

In the following, we consider scenarios where the provided seed information is one or more tracks, representing a playlist beginning or a set of recently played tracks. Given this ordered list of tracks and possibly some auxiliary information, e.g., track metadata, musical features of the tracks, or the user's past preferences, the goal is to compute a ranked list of tracks to be recommended as a playlist continuation. Generally, the setting therefore corresponds to a session-aware or session-based recommendation scenario, as discussed in Chapter 1 and in more detail in Quadrana *et al.* (2018).

Non-Personalized Approaches. A large part of the published approaches in the research literature are non-personalized, i.e., given a playlist beginning, they return the same set of tracks for every user. Technically, a variety of approaches from different algorithm families has been explored Bonnin and Jannach (2014). Generally, we can differentiate between two types of approaches:

1. *Sequence-agnostic techniques*: These algorithms only consider the co-occurrence of items in the current session and past listening sessions or playlists.

2. *Sequence-aware techniques*: Algorithms of this type also consider the order of the tracks, both in the current as well as in the past sessions.

Sequence-agnostic Approaches: A basic approach would be to apply frequent pattern rule mining techniques and look for tracks that often appear together in the listening logs of users in the past. These association rules can then be used to make Amazon-like recommendations of the form "Customers who bought this item also bought these items". A more effective method, however, is to apply a session-based k-nearest-neighbor (kNN) approach as discussed in Hariri *et al.* (2012) or Bonnin and Jannach (2014).

The kNN method takes the sequence of the last played tracks as an input and then in a first step determines the k most similar past sessions in the logs of all user sessions. Given the current session s, the set of the k nearest neighbors N_s, and a function $sim(s_1, s_2)$ that returns a similarity score for two sessions s_1 and s_2, the score of a recommendable item i is defined as

$$score_{\text{KNN}}(i, s) = \Sigma_{n \in N_s} sim(s, n) \times 1_n(i) \tag{3}$$

where $1_n(i) = 1$ if n contains i and 0 otherwise, see also Bonnin and Jannach (2014). Technically, different distance (similarity) measures can be used to compare two sessions or playlists, e.g., binary cosine similarity. To ensure scalability, neighborhood sampling can be applied, e.g., based on the recency of the listening logs of the community Jannach and Ludewig (2017).

Sequence-aware Techniques: A number of sequence-aware techniques were explored in the literature as well. *Sequential patterns* are a counterpart of frequent pattern techniques and were investigated, e.g., in Bonnin and Jannach (2014). In a related approach, Hariri *et al.* (2012) aimed at the identification of topic sequences in playlists instead of only looking at track sequences. Their results showed that considering these topic transitions leads to a performance improvement over a kNN technique. In another work, Park *et al.* (2011) presented a modified collaborative filtering method that uses session information to capture sequence and repetition patterns in listening sessions. They showed that their proposed session-based CF approach can outperform a basic CF method. A number of alternative and comparably simple sequence-based techniques were recently evaluated in the music domain and other domains in Kamehkhosh *et al.* (2017); Ludewig and Jannach (2018). The investigated methods include, for example, sequential patterns of size two or neighborhood-based techniques that use a similarity functions which consider the order of the tracks in a session.

More elaborate techniques are based on *sequence learning* models. Some approaches of that type that are mentioned in the literature for example include Recurrent Neural Networks (RNNs) Bernhardsson (2014). In principle, however, any form of algorithm that can be used for sequence-aware session-based recommendation can be applied to the music domain as well, including, for example, ones that use Markov-models in some form Rendle *et al.* (2010) or that rely on special types of RNNs Hidasi *et al.* (2016a); Hidasi and Karatzoglou (2017). A potential drawback of such more sophisticated models is that they can easily become computationally challenging. At the same time, today's more sophisticated models are not necessarily better than the simple techniques mentioned above in terms of the prediction accuracy, as discussed in Ludewig and Jannach (2018).

Personalized Approaches. While *item discovery* approaches are almost always personalized, the work presented in Jannach *et al.* (2017) represents one of the few attempts in the literature where the process of creating a playlist continuation is personalized. The personalization features in this work are embedded within a multi-faceted track scoring scheme. First, a baseline relevance score for each recommendable track is computed based on a k-nearest-neighbor method as described above. Then, a variety of other relevance signals are considered in a weighted approach.

Technically, the overall relevance score $score_{overall}$ for a possible next track t^*, given the playlist beginning h, is computed as follows Jannach *et al.* (2017):

$$\begin{aligned} score_{overall}(h, t^*) = w_{base} \cdot score_{base}(h, t^*) \; + \\ \sum_{pers \in P} w_{pers} \cdot score_{pers}(h, t^*) \end{aligned} \tag{4}$$

where P is a set of personalization strategies, each with a different weight w_{pers}, and w_{base} is the weight of the baseline. The functions $score_{base}$ and $score_{pers}$ compute the baseline score and the scores of the individual personalization components, respectively.

The following signals were considered for the personalization step:

1. Favorite Tracks
2. Favorite Artists
3. Topic Similarity
4. Extended Neighborhood
5. Social Friends

Favorite Tracks: It is quite common in the domain that users listen to their favorite tracks again and again. Therefore, it is reasonable to also recommend tracks that the user has already heard (several times) in the past. Different strategies to select tracks for repeated recommendations are possible. One can, for example, repeatedly recommend tracks that are generally popular; or, one can recommend tracks that the user has listened to at the same time of the day in the past. Yet another approach is to repeatedly recommend tracks of artists that the user is listening to in the current session.

Since the literature suggests that users tend to repeatedly consume things that they have experienced more recently Anderson *et al.* (2014), one can furthermore give more weight to tracks that were more recently played by the user. Finally, the effectiveness of repeated recommendations can be further increased when the system can guess if the user is currently in the mood of discovering something new or rather prefers to listen to his or her favorites, as discussed in Kapoor *et al.* (2015).

Favorite Artists: Many music lovers have their favorite artists and it therefore might be a comparably safe strategy to recommend tracks of these artists to users, even when this might lead to limited discovery effects.[6] Technically, to compute an artist-based score, one can inspect the current and the past listening session of the user and assign higher scores to tracks that are performed by the artists that also appeared in these past listening sessions.

Topic Similarity: The assumption of this personalization score is that some users are interested in certain types of music, for example, mostly sad ballads or instrumental music. Therefore, recommendable tracks that in some respect are "content-wise" similar to those that the user listened to in the past should receive a higher rank. Generally, one can use all sorts of information to determine the similarity of objects, e.g., based on musical features. Since such information is not

[6]We will discuss questions of recommendation quality later on in more depth.

always available, one can also look at the publicly available tags that users assign to certain artists or tracks, for example, on Last.fm. Technically, again different similarity measures can be applied, for example, by using the cosine similarity between two TF-IDF encoded track representations.

Extended Neighborhood: The kNN-method described above merely looks for sessions that are similar to the current one. To consider the long-term personal preferences of the user, one can however also consider sessions that are similar to the *past* sessions of the user, maybe with a lower weight.

Social Friends: Finally, the last personalization approach considered in Jannach *et al.* (2017) takes the musical preferences of the user's social friends into account. Our musical tastes and preferences, as mentioned in the introduction, can to some extent be determined by the tastes of our social environment. One possible technical approach is therefore to recommend the favorite tracks of the user's social friends, giving more weight to social friends that are generally more popular and, e.g., have more followers.

Overall, experimental evaluations in Jannach *et al.* (2017) showed that all these personalization features can have a positive effect on the quality of the resulting recommendations. The best results are usually achieved when multiple signals are considered in parallel, which however requires that the importance weights are fine-tuned. Generally, the applicability of some approaches depends on the availability of the corresponding data, e.g., the information about the user' social connections.

3 Challenges

Building a successful music recommender system can be challenging in a variety of dimensions. A key challenge clearly is to understand what makes a good recommendation in the first place, e.g., because the perception of music is highly subjective and context-dependent, as will be discussed in Section 4.

3.1 Data-Related Aspects

Huge amounts of data can be available for providers of a music streaming service. For larger services, there are millions of recommendable items and even more users.[7] Furthermore, there can be billions of streaming events, which can potentially be leveraged for building better recommendation models. As a result, scalability aspects can be a main concern when choosing or designing a recommendation algorithm. For the case of the implicit matrix factorization approach at Spotify, for example, only the play counts for the tracks were considered when optimizing the models. Obviously, however, one could also consider the temporal sequences of the events or even the musical features of the individual tracks to end up with better recommendations Cai *et al.* (2007); Su *et al.* (2010); Dias and Fonseca (2013); Johnson (2014).

Another data-related issue is concerning potential biases in the available data. There is a very long tail of musical tracks available on music platforms that have been barely listened to by anyone and many modern algorithms might mostly focus on the more popular items Celma and Cano (2008). This might in turn lead to a "rich-get-richer" (popularity reinforcement) effect as a small fraction of the catalog receives most of the (mostly positive) feedback. Furthermore, when using listening logs as input, the recorded events might be, to some larger extent, influenced by a recommendation functionality that was provided by the platform.

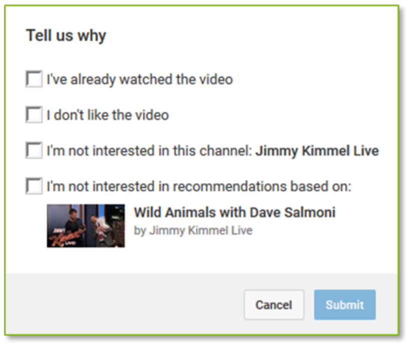
[7] As of 2018, Spotify reports to have over 70 million paying subscribers and almost 160 million active users of the service (`https://press.spotify.com/es/about/`, accessed 8 March 2018).

3.2 User Interaction Aspects

The recommendation techniques discussed in this chapter are mostly based on implicit feedback, i.e., play events for the tracks. However, music platforms also provide different mechanisms for users to explicitly state their preferences. Most platforms, for example, give the users the opportunity to rate individual tracks (recommendations), typically using a binary feedback scale. In addition, users can often skip individual tracks, which represents another form of explicit feedback.[8] At least on some platforms, like Microsoft Groove, users are initially asked to provide their preferences for certain artists or genres.

In the context of explicit preference statements, different challenges can arise and various design decisions have to be made.[9] The initial preference acquisition process, for example, should not represent a burden for the user. And, users should be given the opportunity to revise their statements later on. Regarding the ratings for individual tracks, we commonly see thumbs-up/thumbs-down approaches, probably because many users tend to give only extreme ratings or would find it too tedious to give fine-grained feedback, e.g., on a five-point scale. A specific phenomenon in that context mentioned in Steck *et al.* (2015) is that the provided explicit preference statements might not be fully reliable, and users sometimes use their preference statements to create a public image of themselves, but then in fact listen to other types of music more frequently.

Finer-grained forms of giving feedback are provided, however, on other media sites. Figure 3, for example, shows how users could give feedback to individual video recommendations at YouTube around the year 2015. Here, the users could not only state that they did not like a certain item, but also state that they either had seen it before or that they are not interested in a certain channel. Clearly, while such an approach gives more control to the users, it also increases the complexity and required user effort. At the same time, such fine-grained feedback forms can be difficult to operate on mobile devices, which are often used to consume streaming music.


Tell us why

- ☐ I've already watched the video
- ☐ I don't like the video
- ☐ I'm not interested in this channel: **Jimmy Kimmel Live**
- ☐ I'm not interested in recommendations based on:

Wild Animals with Dave Salmoni
by Jimmy Kimmel Live

Figure 3: Feedback on the YouTube platform (replaced with a simpler form as of 2018).

In order to acquire short-term listening preferences, music platforms, as described above, for example let the user provide some initial track or artist, and base the subsequent tracks on that seed information. Since (a) the system might misinterpret the user's intentions or (b) users sometimes only has a vague idea of what they want to hear Steck *et al.* (2015), at least some form of allowing the user to fine-tune the playlists has to be provided. For the Spotify app, it was decided to provide thumbs-up/thumps-down buttons for the user. A particular challenge in that context, however, is that the user's reactions should be "immediately gratifying", i.e., the playlist should be refreshed instantaneously, at least when there is negative feedback on the track. This might in turn lead to

[8]In fact, skipping a track can be considered both as implicit and explicit feedback.
[9]For a recent review on user interaction aspects for recommender systems, see Jugovac and Jannach (2017).

computational challenges and limited research exists in the literature that discusses algorithmic approaches to incorporate such feedback on the fly.

In addition, as mentioned in Johnson (2014), most of the provided feedback is positive and obviously only available for tracks that were actually recommended to users. As a result, this can easily lead to some algorithmic bias when optimizing only using feedback that was given to tracks that were presented to users.

Generally, not much research exists on how to design the user interfaces of music apps that feature a recommendation component. One typical aspect that is mentioned as a way to increase the adoption of the recommendations is the provision of explanations, so that the users have a chance to *understand* why a certain item was recommended Lamere and Celma (2011). Other open questions include how to determine the actual users' context, e.g., their mood, activities, surroundings etc., either through explicit elicitation methods or through additional sensor information.

3.3 Incorporating Song Feature Information

Relying mostly or even solely on the collective behavior of a community when recommending using collaborative filtering techniques can have certain limitations, e.g., low prediction accuracy when there are limited amounts of feedback signals for recently added tracks (*item cold-start*). Therefore, in this domain, considering the musical features of the recommendable tracks can be helpful to avoid unsuitable elements in a playlist, e.g., high-energy tracks within a chill-out playlist. This is in particular important since such unsuitable recommendations can have a measurable negative effect on the user's quality perception of the service Chau *et al.* (2013).

Traditional *content-based* recommendation approaches typically suffer less from such problems. The basic idea of such algorithms is to recommend items that are similar to the ones that the user has liked in the past. Technically, this is usually achieved by representing both users and items in terms of the features of the musical tracks. The features can include both musical characteristics as well as track meta-data such as release years. The shared form of representation can then be used to compute the match (similarity) between a given user profile and a recommendable track. A common way of representing the items is to use vectors whose dimensions correspond to content or meta-data features, such as the tempo, the loudness, the different possible genres, etc. Users are also represented based on their preferences, which either can be obtained by asking the user to explicitly provide some preference information (e.g., using a questionnaire), or can be extracted from the music that the user has listened to in the past.

In principle, a large number of musical features can be used, including low-level ones related to the timbre or rhythmic aspects as well as more high-level ones like the instrumentation or the "danceability" of a track. Pandora.com, as mentioned above, relies on manual annotations by experts. An alternative is to automatically extract the information from various sources, including the audio signal as well as other data sources like social annotations or lyrics. Extracting such features is a key topic in the field of music information retrieval and significant progress has been made in the last years, see, e.g., Casey *et al.* (2008); Müller (2015); Weihs *et al.* (2016). The extraction process can, however, lead to inaccurate results in some cases. At the same time, the process can be computationally challenging, given the huge number of available tracks.

Since also content-based methods have limitations, e.g., their tendency to recommend "more of the same", a large variety of *hybrid* algorithms were proposed in the literature on recommender systems. Such hybrids combine different techniques and are usually designed in order to overcome the limitations of the individual techniques. A key challenge in that context is how to merge the available types of information in the best possible way.

An early hybrid method was proposed by Yoshii *et al.* (2006), who used a probabilistic model in the form of a Bayesian network to integrate user ratings collected from Amazon and content data represented as mel-frequency cepstral coefficients. Their results showed that the hybrid method achieves higher accuracy and more diverse recommendations in terms of artists than the two underlying methods. In a more recent work, Wang and Wang (2014) developed an approach to combine a collaborative filtering method based on probabilistic matrix factorization Salakhutdinov and Mnih (2007) and content features that were learned automatically via a deep belief network Hinton *et al.* (2006).

Generally, a number of ways exist to combine the recommendations of different algorithms, e.g., by weighting the individual prediction scores or by using the recommendations generated by one algorithm as a pre-filtered input to another. Alternatively, machine learning models like Factorization Machines can be applied that combine the different types of input data in an integrated model.

Overall, in many application domains of music recommendation hybrid techniques can be considered the method of choice. When creating a playlist for a virtually endless radio station, for example, the application of collaborative filtering techniques can increase the probability that the recommendations include tracks that are new to the user ("discovery"). Using content-based techniques in parallel can, at the same time, help to ensure that the tracks played in the future do not deviate too much from the seed tracks in terms of their musical features. A general challenge in the context of such hybrid systems, however, is how to combine the different techniques in the best possible way.

4 Evaluation

The evaluation and comparison of different music recommendation strategies can be challenging, both for music service providers and for academic researchers. After discussing the determination of the relevant quality criteria in general and how to balance these criteria, this section will focus on the assessment of performance in real world settings and in academic environments.

4.1 Quality Criteria

Generally, the acceptance of a music recommendation service depends on the quality and utility perception of its users. Regarding the specific recommendation scenario, a number of different factors can have an influence on the users' perceptions. The following list gives examples of such factors, as mentioned also in Bonnin and Jannach (2014) in the context of the playlist generation problems. In general, these factors can be *music-related* and *purpose-oriented* Jannach and Adomavicius (2016).

Examples of *music-related* factors:

- *Diversity:* The recommendations (e.g., a list of tracks to be played next) should be sufficiently different from each other, e.g., in terms of their artist Slaney and White (2006); Lee *et al.* (2011); Kamalzadeh *et al.* (2012); Puthiya Parambath *et al.* (2016).

- *Homogeneity:* At the same time (e.g., in the context of playlist construction support), it is often desirable that the next tracks are not too different from each other, for instance, in terms of their genre, tempo, or mood Logan (2002); Balkema and van der Heijden (2010); Jannach *et al.* (2015).

- *Coherence:* Also, it is often important that the recommended tracks do not only fit to each other, but also represent coherent continuations to the previously played tracks Kamehkhosh and Jannach (2017).

- *Transitions:* In some situations, for example, when creating playlists for parties, the differences between subsequent tracks, e.g., in terms of their tempo, should in general not be too high Flexer *et al.* (2008); Sarroff and Casey (2012).

Examples of *purpose-oriented* factors:

- *Discovery / Novelty*: The recommendations could, for example, be designed to be helpful for users to discover new artists, tracks, or genres Celma (2010); Zhang *et al.* (2012).
- *Agreeableness*: In particular, when a set of music track recommendations should be listened to by a group of users (e.g., at a party), it might be important that the music is enjoyed by the majority, which can, for example, be achieved by mainly focusing on generally popular tracks Popescu and Pu (2012).
- *Context-fit*: As described in Bonnin and Jannach (2014), playlists are often created for a certain "theme" or context (e.g., sports workout). The quality of the recommendations then depends on their fit for the given theme or context.

As can be easily seen, the different quality criteria can be antagonistic, e.g., agreeableness vs. discovery or homogeneity vs. diversity, and trade-offs have to be found. How to balance these aspects can depend on the specific application scenario.

4.2 Balancing Different Quality Factors

Given these potentially conflicting goals, it is often insufficient to only consider the relevance of a certain item in isolation. Instead, the suitability of an entire list of tracks to be played next has to be taken into account. Usually, however, considering additional quality factors other than prediction accuracy leads to a trade-off situation as multiple, possibly competing goals have to be balanced.

In the research literature on recommender systems, a number of approaches were proposed to deal with such trade-off situations and to improve recommendations by considering additional quality factors, e.g., in Adomavicius and Kwon (2012); Ziegler *et al.* (2005); Bradley and Smyth (2001); Zhang and Hurley (2008), or Vargas and Castells (2011).

Most of these approaches however (i) consider only two quality factors (e.g., accuracy vs. diversity) and (ii) do not consider the user's individual tendencies (e.g., with respect to diversity in general). Some more recent works that try to overcome these limitations can be found in Kapoor *et al.* (2015); Shi *et al.* (2012); Oh *et al.* (2011), and Ribeiro *et al.* (2014). These approaches are however often designed for a very specific quality factor (e.g., diversity). Or, they implement the balancing strategy within their own recommendation algorithm so that existing algorithmic frameworks and approaches cannot be reused.

An alternative, two-phase approach that can be combined with any existing item-ranking algorithm was proposed in the context of next-track music recommendation problems in Jannach *et al.* (2015). Similar to works like Adomavicius and Kwon (2012), the general idea of the method is to take a limited set of k (e.g., $k = 30$) most relevant items according to an arbitrary item-ranking or scoring algorithm and then to re-rank these top items in a way that selection of the top-10 items optimizes some quality criterion. In the context of the "radio" problem, this could mean to take the top 30 tracks according to their predicted suitability as a continuation for the last played tracks and then re-rank the items in order to maximize the artist diversity within the 10 next tracks. To compute such a re-ranked list, a greedy approach can be applied, which can compute solutions in real time that are very close to the true optimum.

A general question in such problem settings, however, is to determine the *right amount* of artist diversity. Some approaches use a combined quality measure that considers both accuracy and diversity. This, however, assumes that there is a globally accepted level of, e.g., diversity. In reality, the appropriate level of diversity (or homogeneity etc.) can be dependent on different factors, including the diversity of current playlist or even the general long-term preferences of the particular user. The method proposed in Jugovac *et al.* (2017) takes such considerations into account. Specifically, the implemented optimization procedure can be configured to be guided by the characteristics of a "seed" of items and the optimization goal then consists of *minimizing the difference* between the seed items' characteristics and the characteristics of the recommendation list.

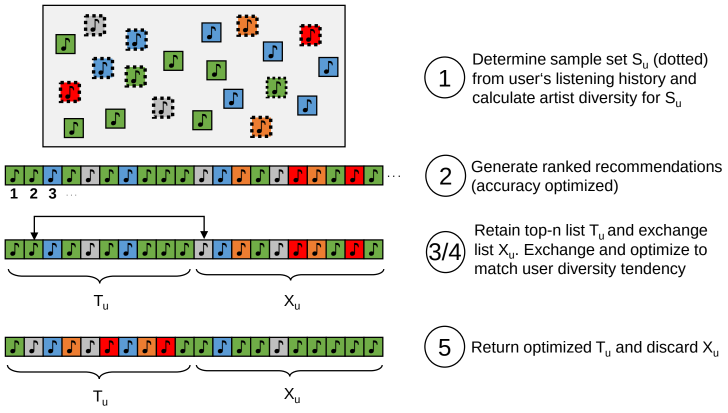

1 Determine sample set S_u (dotted) from user's listening history and calculate artist diversity for S_u
2 Generate ranked recommendations (accuracy optimized)
1 2 3 ...
3/4 Retain top-n list T_u and exchange list X_u. Exchange and optimize to match user diversity tendency
T_u
X_u
5 Return optimized T_u and discard X_u
T_u
X_u

Figure 4: Illustration of the re-ranking scheme, adapted from Jugovac *et al.* (2017).

Figure 4 illustrates the general idea based on the artist diversity problem. The elements in dotted lines in the upper rectangle (marked with ①) represent the selected seed items and the different colors represent different artists. The seed set can for example be taken from the set of the user's last played tracks or be a subset of the user's favorite tracks. Based on this seed set, the user's individual *tendency* towards artist diversity can be computed. In the next step, the ranked list of recommendations (next tracks for a given playlist history) is computed with any algorithm. Again, different colors in step ② represent different artists. Since the top-10 items of the list have a lower artist diversity than the items in the seed set, the algorithm then starts exchanging elements from the top of the list with elements from the end of the list, which probably have a slightly lower predicted relevance but help to improve the diversity of the top-10 list. Improving in that context means that the algorithm tries to minimize the difference between the diversity level of the top-10 list and the seed tracks. Therefore, if a user generally prefers lists with high diversity, the re-ranking will lead to higher diversity. Vice versa, a user who usually listens to various tracks by the same artist in a session will receive recommendations with a lower artist diversity. As a result, the definition of a globally desired artist diversity level can be avoided.

Generally, there can be multiple and possibly conflicting optimization goals that should be considered in parallel, e.g., high artist diversity, homogeneous tempo, smooth transitions. Such situations can be considered in the approach as long as the relative importance of the difference factors can be specified. The acceptable compromises on recommendation accuracy can also be fine-tuned by determining the size of the top-k list from which items can be picked during the optimization process.

4.3 Performance Assessment in Real-World Settings

In order to assess the performance of a music recommendation service, an understanding about its expected utility or value is required in the first place. Recommendation services on music platforms are primarily designed to support the user during a certain task like playlist construction or to provide some other functionality like a personalized radio station. While such services often do not directly lead to additional revenue for the provider as in the e-commerce domain, recommendation services on music platforms can represent an additional value for the consumer (e.g., in terms of better a user experience or an easier discovery of new things). This can, in turn, lead to a higher customer retention and an indirect business value.

In order to quantify the effectiveness of a recommendation service, different basic measurements and analyses can be made. First, one can measure the *adoption* of the service, i.e., determine which fraction of the users actually use the service over an extended period of time to a significant extent. For certain services like a personalized radio station, one can also analyze the specific user behavior and their feedback to the recommendations. Do users, for example, skip certain recommended tracks or are they continuing to listen to the recommended tracks most of the time? Such measurements regarding the acceptance of the recommendation can be made and compared in field (A/B) tests, possibly accompanied by user surveys. Finally, since the user interface (UI) design can have a significant impact on the adoption and effectiveness of some recommendation services Steck *et al.* (2015), the UI design should be evaluated as well through laboratory studies, A/B tests, or both.

The indirect business value of a music recommendation service can typically only be assessed through field tests and the specific measurement depends on the business model of the provider. For flat-rate subscription-based services like Spotify or others, user engagement and customer retention (subscription renewals) might be a measure that one wants to optimize. Another potential measure is the conversion of users of a free service to premium (paying) users.[10] If the music recommendation service on the other hand is used to recommend items in a pay-per-stream or pay-per-download model, the overall revenue or profit usually represent the target measures.

A main challenge in that context is that for many complex machine learning approaches a number of parameters have to be optimized, and that not all parameter settings can be tested in field studies. Therefore, algorithms are usually optimized regarding some *proxy* measures in an offline process (e.g., precision or recall). The challenge however is that for many of these proxy measures, it is unclear if they truly correlate with the target measure that one wants to optimize in reality (e.g., customer retention). In fact, a number of recent works suggest that success measures often used in academia, in particular the prediction accuracy, are often not good indicators of the true success of a recommendation service, see also the discussion of Netflix's recommendation service in Gomez-Uribe and Hunt (2015). Another challenge when optimizing accuracy measures in an offline process is that the observed data (e.g., thumbs) can be biased in different ways. Often, the large majority of the signals consists of only positive feedback. And, which items are actually listened to (and rated) by consumers can itself be heavily influenced by the existing recommendation service or other functionalities of the platform, e.g., lists of trending items.

4.4 Comparing Algorithms in Academic Environments

Similar to other application domains of recommenders, the predominant form of evaluating recommendation algorithms in the music domain is based on offline experiments, using either explicit user preference data or recorded user activity logs.

[10]See Bernhardsson (2017) for a discussion of potential issues when computing conversion rates.

Datasets. In case the goal is to predict user preferences (i.e., the relevance of individual items) for a user given a common user-item preference matrix, "standard" evaluation schemes for the matrix completion setup from other application domains can be applied.

For many application scenarios of music recommenders – and in particular for personalized radio stations – using recorded listening logs as a basis for offline experiments however seems more appropriate. A number of public datasets are available online. Some of them contain the listening histories of thousands of users over an extended period of time. Often, these logs were obtained through the public APIs of music services like Last.fm. Table 2 gives an overview of a number of public datasets.

Finally, some research works on collaborative music recommendation are based on publicly shared "hand-crafted" playlists (mixtapes), i.e., sequences of tracks that were put together and shared by users of music services for a certain purpose, e.g., for a road trip or a sports workout. Different datasets containing such playlists are publicly available today, see also Table 2.

Evaluation Protocols and Metrics. Comparing the prediction accuracy of different music recommendation strategies in terms of information retrieval or machine learning measures is common in the academic literature. In case sequential user activity logs or playlists are the basis for the evaluation, a typical evaluation procedure is to use a "session-wise" approach as discussed in Chapter 1. For each listening session or playlist, a defined number of tracks from the beginning of a session are revealed and the task of the recommender is to predict the next (hidden) track(s). Such a research approach was, for example, used by Hariri *et al.* (2012) or by Bonnin and Jannach (2014), where standard information retrieval measures were applied. An alternative evaluation approach based on the *Average Log-Likelihood* was proposed by Mcfee and Lanckriet (2011), a measure which can be used to assess how likely a system is to generate the tracks of a given playlist or listening session. This measure, however, has certain limitations, as discussed in Bonnin and Jannach (2014).

Focusing solely on accuracy measures has its limitations also in the music domain. Predicting mostly very popular tracks in fact proves to be a very competitive strategy. In Bonnin and Jannach (2013), the authors proposed a popularity-based method called "Collocated Artists – Greatest Hits". This method simply takes the artists who appear in the given listening session and then recommends to play the most popular tracks of these artists and of similar artists, where the artist similarity is determined based on their co-occurrence in past listening sessions. Experimental evaluations show that this method is particularly effective in guessing the immediate next tracks.

Recommendation lists that include only very popular tracks might however not be satisfying for many users, e.g., because these recommendations lead to limited discovery and low artist diversity. Therefore, researchers sometimes apply multi-metric evaluation schemes which consider both prediction or ranking accuracy and other quality factors like the homogeneity of the recommended tracks, artist diversity, coherence with the previous tracks, or the transitions between the tracks Jannach *et al.* (2016, 2017). As discussed above, different techniques can then be applied to balance the given quality factors, see Jugovac *et al.* (2017) for a comparison of recent approaches, which also showed that considering additional quality factors can even lead to an improvement in terms of ranking accuracy. Generally, the problem in these situations is not only to assess the relevance of the individual recommended items, but to consider quality factors that are determined by the characteristics of the recommendation list as a whole.

[11]The Echo Nest has been acquired by Spotify in 2014. A similar Web API has since been made available by Spotify, see `https://developer.spotify.com/web-api`

[12]`http://webscope.sandbox.yahoo.com`

[13]`https://labrosa.ee.columbia.edu/millionsong/tasteprofile`

[14]`http://www.kkbox.com`

[15]`https://www.kaggle.com/c/kkbox-music-recommendation-challenge/leaderboard`

[16]`http://www.artofthemix.org`

Table 2: Selection of public datasets.

Type	**Name**	**Description**
Music meta-data	Million Song Dataset	Made available in 2011 by researchers from the Music Information Retrieval community Bertin-Mahieux *et al.* (2011). The dataset consists of meta-data for 1 million tracks of 44,745 artists, including tags, date of release and several acoustic features from The Echo Nest.[11]
Rating data	Yahoo! Music	Artist and track ratings from data sets published by Yahoo! Research.[12]
	Taste Profile subset	An addition to the Million Song Dataset that contains real user play counts for a subset of the tracks (384,546 tracks) and was released by The Echo Nest.[13]
	Amazon product co-purchasing network metadata	Made available in Leskovec *et al.* (2007). It contains various reviews and review information such as ratings, number of votes, etc. on 548,552 items of different types (books, music CDs, DVDs and VHS video tapes).
	InCarMusic	Constructed for the work presented in Baltrunas *et al.* (2011). It contains user ratings of genres and car-related context information, such as driving style (relaxed or sport), road type (city, highway or serpentine), etc. for about 140 tracks of 10 different genres.
Listening logs	Last.fm	Celma (2010) published two datasets containing the listening logs of about 1,000 and 360,000 users, respectively, collected from Last.fm API.
	30Music	A collection of listening and playlists data retrieved from Internet radio stations through Last.fm API. The datasets consists of 31 million user play events, 2.7 million user play sessions, and 4.1 million user "love" statements and was published by Turrin *et al.* (2015).
	KKBOX	KKBOX is a music streaming service provider in East Asia.[14] In the context of a machine learning competition[15], listening logs of over 34,000 users were published together with some information about over 400,000 individual tracks.
Listening logs extracted from microblogs	MMTD	The Million Musical Tweet Dataset includes listening histories based on more than 1 million tweets referring to 133,968 unique tracks by 25,060 different artists created by 215,375 users. The dataset was introduced by Hauger *et al.* (2013).
	#nowplaying	Contains about 40 million listening events extracted from music-related tweets of users on Twitter. The dataset is enriched with additional information about the artist, the track title, and metadata about the tweet and was published by Zangerle *et al.* (2014).
Music playlists	Art of the Mix	A dataset of hand-crafted playlists made available by McFee and Lanckriet (2012). It contains all playlists (about 100,000) of the Website Art of the Mix[16] created before June 2011. 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.
	Kollect.fm	Contains about 35,000 playlists with feedback about them from about 16,000 users retrieved from kollect.fm, which is a music discovery website where users can receive recommendation of playlists.

Research works that are based on recorded user activity logs have additional limitations. In particular, it is not clear for datasets that are obtained, e.g., from Last.fm, to which extent the users' listening activity is "natural" in the sense that the users selected one track after the other when listening. In reality, larger parts of log entries might be the result of the existing "radio" service of the platform. Evaluating algorithms in terms of predicting the next track in the log then amounts to predicting what the existing recommendation service would play. Also, in some datasets, we can observe that users listen to entire albums and the sequence of the tracks then often corresponds to the order of the tracks on the album. Predicting the next track is then comparably trivial and an algorithm that is capable of detecting album listening sessions will achieve high prediction accuracy values for these cases.

Offline evaluation approaches have their limitations, as discussed above, in particular as it is not always clear if the chosen computational metrics are good estimators of (a) the quality perception of users and (b) of the business success of the recommendation service. User studies on music recommendation approaches are comparably rare. Examples of such studies include can be found in Barrington *et al.* (2009) and, more recently, in Kamehkhosh and Jannach (2017) and Kamehkhosh *et al.* (2018). In particular, the work presented in Kamehkhosh and Jannach (2017) indicated that using hand-crafted playlists can represent a reasonable "gold standard" for evaluating playlist generation techniques. The work, however, also revealed potential limitations of user studies in terms of familiarity effects, i.e., users prefer recommendations when they contain tracks they already know.

5 Lessons Learned, Open Challenges, and Outlook

Collaborative filtering based recommendations are nowadays a common functionality on several music platforms. And, as described in the chapter, at least some of these platforms rely on elaborate algorithms, e.g., based on matrix factorization, to create personalized recommendations.

Several challenges of building such recommendation services have been discussed throughout the chapter, including in particular the problems of scalability, the importance of considering short-term trends, or the general problem of finding good proxy measures for offline evaluation scenarios.

Pure collaborative filtering approaches, in general, have a number of limitations and this applies also for the music domain, where we for example have the phenomenon that constantly new items are available on the platform. To recommend such items, relying on "content" information (e.g., musical signals, metadata, lyrics) is a viable approach to assess the relevance of new tracks for individual users. How to combine the multitude of different signals in the best possible way in hybrid recommendation techniques is in our view an area which requires additional research. Recently, Jannach *et al.* (2017) investigated the use of rich user models that combine collaborative filtering techniques with such content information and social information, showing that considering all of these signals can be beneficial. This work was based only on a very limited set of musical features and simple weighting schemes. More elaborate approaches to derive and process a variety of signals, e.g., based on deep learning approaches, already exist today and should be further explored in the future van den Oord *et al.* (2013); Wang *et al.* (2015); Hidasi *et al.* (2016b).

Another music recommendation scenario that is not fully investigated in the literature is group-based music recommendation. Consider, for example, playlist creation for a party or music selection in public places. In such scenarios, the recommended items are consumed by a group of listeners rather than by individuals and the recommendations should, therefore, satisfy the entire group as a whole. Generally, group-based recommendation strategies either aggregate individual user models to build a group profile, or aggregate individual predictions for users to generate group-based recommendations Berkovsky and Freyne (2010).

More research is also required in terms of understanding which factors influence the quality perception by users. A number of questions are open in that context, for example: How much diversity, e.g., in terms of artists or genres, is appropriate? To which extent is the diversity level depending on the current user's context, e.g., mood? How do we quantify diversity with a computational metric and how do we know that the computational metric corresponds to the user's quality perception? How problematic are "bad" recommendations – can the recommendation of one or a few unsuitable items have a significant negative effect on the acceptance of the service?[17]

Finally, many of the mentioned quality factors can depend on the user's current context. Limited research exists on the contextualization of recommendations, see, for instance, Hariri *et al.* (2012) or Kapoor *et al.* (2015), which mainly try to derive the user's context and short-term preferences from their behavior. So far, to the best of our knowledge, only the InCarMusic dataset Baltrunas *et al.* (2011) contains additional information about the users' context.

References

Adomavicius, G. and Kwon, Y. (2012). Improving Aggregate Recommendation Diversity Using Ranking-Based Techniques, *IEEE TKDE* **24**, 5, pp. 896–911.

Anderson, A., Kumar, R., Tomkins, A., and Vassilvitskii, S. (2014). The dynamics of repeat consumption, in *WWW '14*, pp. 419–430.

Bachrach, Y., Finkelstein, Y., Gilad-Bachrach, R., Katzir, L., Koenigstein, N., Nice, N., and Paquet, U. (2014). Speeding Up the Xbox Recommender System Using a Euclidean Transformation for Inner-Product Spaces, in *RecSys '14*, pp. 257–264.

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

Baltrunas, L., Kaminskas, M., Ludwig, B., Moling, O., Ricci, F., Aydin, A., Lüke, K.-H., and Schwaiger, R. (2011). InCarMusic: Context-Aware Music Recommendations in a Car, in *EC-Web '11*, pp. 89–100.

Barrington, L., Oda, R., and Lanckriet, G. R. G. (2009). Smarter than Genius? Human Evaluation of Music Recommender Systems, in *ISMIR '09*, pp. 357–362.

Berkovsky, S. and Freyne, J. (2010). Group-based Recipe Recommendations: Analysis of Data Aggregation Strategies, in *RecSys '10*, pp. 111–118.

Bernhardsson, E. (2013). Music Recommendations at Spotify, Online `https://de.slideshare.net/erikbern/collaborative-filtering-at-spotify-16182818`.

Bernhardsson, E. (2014). Recurrent Neural Networks for Collaborative Filtering, Online `https://erikbern.com/2014/06/28/recurrent-neural-networks-for-collaborative-filtering.html`.

Bernhardsson, E. (2015). Approximate Nearest Neighbor Methods and Vector Models, Online `https://de.slideshare.net/erikbern/approximate-nearest-neighbor-methods-and-vector-models-nyc-ml-meetup`.

Bernhardsson, E. (2017). Conversion Rates - You Are (Most Likely) Computing Them Wrong, Online `https://erikbern.com/2017/05/23/conversion-rates-you-are-most-likely-computing-them-wrong.html`.

[17]See Chau *et al.* (2013) for a user study on the topic of bad recommendations.

Bertin-Mahieux, T., Ellis, D. P., Whitman, B., and Lamere, P. (2011). The Million Song Dataset, in *ISMIR '11*, pp. 591–596.

Bieschke, E. (2014). Pandora, presentation at MLconf2013, Online `https://de.slideshare.net/SessionsEvents/eric-bieschke-slides`.

Bonnin, G. and Jannach, D. (2013). Evaluating the Quality of Generated Playlists Based on Hand-Crafted Samples, in *ISMIR '13*, pp. 263–268.

Bonnin, G. and Jannach, D. (2014). Automated generation of music playlists: Survey and experiments, *Computing Surveys* **47**, 2, pp. 26:1–26:35.

Bradley, K. and Smyth, B. (2001). Improving Recommendation Diversity, in *AICS '01*, pp. 75–84.

Cai, R., Zhang, C., Zhang, L., and Ma, W.-Y. (2007). Scalable Music Recommendation by Search, in *MM '07*, pp. 1065–1074.

Casey, M. A., Veltkamp, R., Goto, M., Leman, M., Rhodes, C., and Slaney, M. (2008). Content-Based Music Information Retrieval: Current Directions and Future Challenges, *Proceedings of the IEEE* **96**, 4, pp. 668–696.

Celma, Ò. (2010). *Music Recommendation and Discovery in the Long Tail* (Springer).

Celma, O. and Cano, P. (2008). From hits to niches?: Or how popular artists can bias music recommendation and discovery, in *NETFLIX '08*, pp. 5:1–5:8.

Chau, P. Y. K., Ho, S. Y., Ho, K. K. W., and Yao, Y. (2013). Examining the Effects of Malfunctioning Personalized Services on Online Users' Distrust and Behaviors, *Decision Support Systems* **56**, pp. 180–191.

Chen, C.-M., Tsai, M.-F., Lin, Y.-C., and Yang, Y.-H. (2016). Query-based Music Recommendations via Preference Embedding, in *RecSys '16*, pp. 79–82.

Cliff, D. (2006). hpDJ: An Automated DJ with Floorshow Feedback, in *Consuming Music Together*, pp. 241–264.

Dias, R. and Fonseca, M. J. (2013). Improving Music Recommendation in Session-Based Collaborative Filtering by Using Temporal Context, in *ICTAI '13*, pp. 783–788.

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

Flexer, A., Schnitzer, D., Gasser, M., and Widmer, G. (2008). Playlist Generation Using Start and End Songs, in *ISMIR '08*, pp. 173–178.

Germain, A. and Chakareski, J. (2013). Spotify Me: Facebook-Assisted Automatic Playlist Generation, in *MMSP '13*, pp. 25–28.

Gomez-Uribe, C. A. and Hunt, N. (2015). The Netflix Recommender System: Algorithms, Business Value, and Innovation, *Transactions on Management Information Systems* **6**, 4, pp. 13:1–13:19.

Hariri, N., Mobasher, B., and Burke, R. (2012). Context-aware music recommendation based on latent topic sequential patterns, in *RecSys '12*, pp. 131–138.

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

Hauger, D., Schedl, M., Kosir, A., and Tkalcic, M. (2013). The Million Musical Tweet Dataset - What We Can Learn From Microblogs, in *ISMIR '13*, pp. 189–194.

Hidasi, B. and Karatzoglou, A. (2017). Recurrent Neural Networks with Top-k Gains for Session-based Recommendations, *CoRR* **abs/1706.03847**.

Hidasi, B., Karatzoglou, A., Baltrunas, L., and Tikk, D. (2016a). Session-based Recommendations with Recurrent Neural Networks, in *ICLR '16*.

Hidasi, B., Quadrana, M., Karatzoglou, A., and Tikk, D. (2016b). Parallel Recurrent Neural Network Architectures for Feature-rich Session-based Recommendations, in *RecSys '16*, pp. 241–248.

Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A Fast Learning Algorithm for Deep Belief Nets, *Neural Computation* **18**, 7, pp. 1527–1554.

Hu, Y., Koren, Y., and Volinsky, C. (2008). Collaborative Filtering for Implicit Feedback Datasets, in *ICDM '08*, pp. 263–272.

Jannach, D. and Adomavicius, G. (2016). Recommendations with a Purpose, in *RecSys '16*, pp. 7–10.

Jannach, D., Kamehkhosh, I., and Bonnin, G. (2016). Biases in Automated Music Playlist Generation, in *UMAP '16*, pp. 281–285.

Jannach, D., Kamehkhosh, I., and Lerche, L. (2017). Leveraging Multi-Dimensional User Models for Personalized Next-Track Music Recommendation, in *SAC '17*, pp. 1635–1642.

Jannach, D., Lerche, L., and Kamehkhosh, I. (2015). Beyond "Hitting the Hits" – Generating Coherent Music Playlist Continuations with the Right Tracks, in *RecSys '15*, pp. 187–194.

Jannach, D. and Ludewig, M. (2017). When Recurrent Neural Networks meet the Neighborhood for Session-Based Recommendation, in *RecSys '17*, pp. 306–310.

Jawaheer, G., Szomszor, M., and Kostkova, P. (2010). Comparison of Implicit and Explicit Feedback from an Online Music Recommendation Service, in *Workshop on information heterogeneity and fusion in recommender systems*, pp. 47–51.

Johnson, C. (2014). Algorithmic Music Discovery at Spotify, Online `https://de.slideshare.net/MrChrisJohnson/algorithmic-music-recommendations-at-spotify`.

Johnson, C. and Newett, E. (2014). From Idea to Execution: Spotify's Discover Weekly, Online `https://de.slideshare.net/MrChrisJohnson/from-idea-to-execution-spotifys-discover-weekly/12-Insight_users_spending_more_time`.

Jugovac, M. and Jannach, D. (2017). Interacting with Recommenders - Overview and Research Directions, *ACM Transactions on Intelligent Interactive Systems (ACM TiiS)* **7**.

Jugovac, M., Jannach, D., and Lerche, L. (2017). Efficient Optimization of Multiple Recommendation Quality Factors According to Individual User Tendencies, *Expert Systems With Applications* **81**, pp. 321–331.

Jylhä, A., Serafin, S., and Erkut, C. (2012). Rhythmic Walking Interactions with Auditory Feedback: an Exploratory Study, in *AM '12*, pp. 68–75.

Kamalzadeh, M., Baur, D., and Möller, T. (2012). A Survey on Music Listening and Management Behaviours, in *ISMIR '12*, pp. 373–378.

Kamehkhosh, I. and Jannach, D. (2017). User Perception of Next-Track Music Recommendations, in *UMAP '17*, pp. 113–121.

Kamehkhosh, I., Jannach, D., and Bonnin, G. (2018). How automated recommendations affect the playlist creation behavior of users, in *Proceedings of the Workshop on Intelligent Music Interfaces for Listening and Creation at IUI '18*.

Kamehkhosh, I., Jannach, D., and Ludewig, M. (2017). A Comparison of Frequent Pattern Techniques and a Deep Learning Method for Session-Based Recommendation, in *Workshop on Temporal Reasoning in Recommender Systems at RecSys '17*, pp. 50–56.

Kapoor, K., Kumar, V., Terveen, L., Konstan, J. A., and Schrater, P. (2015). "I Like to Explore Sometimes": Adapting to Dynamic User Novelty Preferences, in *RecSys '15*, pp. 19–26.

Koren, Y. (2008). Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model, in *KDD '08*, pp. 426–434.

Lamere, P. and Celma, O. (2011). Music Recommendation and Discovery Remastered, Tutorial at ACM RecSys 2011, Online `https://musicmachinery.com/2011/10/24/music-recommendation-and-discovery-remastered-a-tutorial/`.

Lee, J. H., Bare, B., and Meek, G. (2011). How Similar Is Too Similar?: Exploring Users' Perceptions of Similarity in Playlist Evaluation, in *ISMIR '11*, pp. 109–114.

Leskovec, J., Adamic, L. A., and Huberman, B. A. (2007). The Dynamics of Viral Marketing, *ACM Trans. Web* **1**, 1.

Logan, B. (2002). Content-Based Playlist Generation: Exploratory Experiments, in *ISMIR '02*, pp. 295–296.

Loni, B., Pagano, R., Larson, M., and Hanjalic, A. (2016). Bayesian Personalized Ranking with Multi-Channel User Feedback, in *RecSys '16* (ACM), pp. 361–364.

Ludewig, M. and Jannach, D. (2018). Evaluation of session-based recommenation algorithms, `arXiv:1803.09587 [cs.IR]`, `https://arxiv.org/abs/1803.09587`.

Mcfee, B. and Lanckriet, G. (2011). The Natural Language of Playlists, in *ISMIR '11*, pp. 537–541.

McFee, B. and Lanckriet, G. R. (2012). Hypergraph Models of Playlist Dialects, in *ISMIR '12*, pp. 343–348.

Moens, B., van Noorden, L., and Leman, M. (2010). D-Jogger: Syncing Music With Walking, in *SMC '10*.

Moling, O., Baltrunas, L., and Ricci, F. (2012). Optimal radio channel recommendations with explicit and implicit feedback, in *RecSys '12*, pp. 75–82.

Moore, J. L., Chen, S., Joachims, T., and Turnbull, D. (2012). Learning to Embed Songs and Tags for Playlist Prediction, in *ISMIR '12*, pp. 349–354.

Müller, M. (2015). *Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications* (Springer International Publishing), doi:10.1007/978-3-319-21945-5.

Oh, J., Park, S., Yu, H., Song, M., and Park, S. (2011). Novel Recommendation Based on Personal Popularity Tendency, in *ICDM '11*, pp. 507–516.

Oliver, N. and Flores-Mangas, F. (2006). MPTrain: A Mobile, Music and Physiology-Based Personal Trainer, in *MobileHCI '06*, pp. 21–28.

Omohundro, S. M. (1989). Five Balltree Construction Algorithms, Tech. rep., International Computer Science Institute, Berkeley, California.

Pálovics, R., Benczúr, A. A., Kocsis, L., Kiss, T., and Frigó, E. (2014). Exploiting Temporal Influence in Online Recommendation, in *ACM '14*, pp. 273–280.

Park, S. E., Lee, S., and Lee, S.-g. (2011). Session-Based Collaborative Filtering for Predicting the Next Song, in *CNSI '11*, pp. 353–358.

Pauws, S., Verhaegh, W., and Vossen, M. (2008). Music Playlist Generation by Adapted Simulated Annealing, *Information Sciences* **178**, 3, pp. 647–662.

Popescu, G. and Pu, P. (2012). What's the Best Music You Have?: Designing Music Recommendation for Group Enjoyment in Groupfun, in *CHI '12*, pp. 1673–1678.

Puthiya Parambath, S. A., Usunier, N., and Grandvalet, Y. (2016). A Coverage-Based Approach to Recommendation Diversity On Similarity Graph, in *RecSys '16*, pp. 15–22.

Quadrana, M., Cremonesi, P., and Jannach, D. (2018). Sequence-aware recommender systems, *ACM Computing Surveys* .

Rendle, S., Freudenthaler, C., and Schmidt-Thieme, L. (2010). Factorizing Personalized Markov Chains for Next-basket Recommendation, in *WWW '10*, pp. 811–820.

Ribeiro, M. T., Ziviani, N., Moura, E. S. D., Hata, I., Lacerda, A., and Veloso, A. (2014). Multiobjective Pareto-Efficient Approaches for Recommender Systems, *ACM Transactions on Intelligent Systems and Technology* **5**, 4, pp. 1–20.

Salakhutdinov, R. and Mnih, A. (2007). Probabilistic Matrix Factorization, in *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, pp. 1257–1264.

Sarroff, A. M. and Casey, M. (2012). Modeling and Predicting Song Adjacencies In Commercial Albums, in *SMC '12*.

Shardanand, U. and Maes, P. (1995). Social Information Filtering: Algorithms for Automating "Word of Mouth", in *CHI '95*, pp. 210–217.

Shi, Y., Zhao, X., Wang, J., Larson, M., and Hanjalic, A. (2012). Adaptive Diversification of Recommendation Results via Latent Factor Portfolio, in *SIGIR'12*, pp. 175–184.

Slaney, M. and White, W. (2006). Measuring Playlist Diversity for Recommendation Systems, in *AMCMM '06*, pp. 77–82.

Steck, H., von Zwol, R., and Johnson, C. (2015). Interactive Recommender Systems, Online `https://de.slideshare.net/MrChrisJohnson/interactive-recommender-systems-with-netflix-and-spotify`.

Su, J.-H., Yeh, H.-H., Yu, P. S., and Tseng, V. S. (2010). Music Recommendation Using Content and Context Information Mining, *IEEE Intelligent Systems* **25**, 1, pp. 16–26.

Turrin, R., Quadrana, M., Condorelli, A., Pagano, R., and Cremonesi, P. (2015). 30Music Listening and Playlists Dataset, in *RecSys '15 Posters*.

van den Oord, A., Dieleman, S., and Schrauwen, B. (2013). Deep Content-Based Music Recommendation, in *NIPS '13*, pp. 2643–2651.

Vargas, S. and Castells, P. (2011). Rank and Relevance in Novelty and Diversity Metrics for Recommender Systems, in *RecSys '11*, pp. 109–116.

Vasile, F., Smirnova, E., and Conneau, A. (2016). Meta-Prod2Vec: Product Embeddings Using Side-Information for Recommendation, in *RecSys '16*, pp. 225–232.

Wang, H., Wang, N., and Yeung, D.-Y. (2015). Collaborative Deep Learning for Recommender Systems, in *KDD '15*, pp. 1235–1244.

Wang, X. and Wang, Y. (2014). Improving Content-based and Hybrid Music Recommendation Using Deep Learning, in *Proceedings of the 22Nd ACM International Conference on Multimedia*, MM '14 (ACM, New York, NY, USA), ISBN 978-1-4503-3063-3, pp. 627–636, doi:10.1145/2647868.2654940, `http://doi.acm.org/10.1145/2647868.2654940`.

Weihs, C., Jannach, D., Vatolkin, I., and Rudolph, G. (eds.) (2016). *Music Data Analysis: Foundations and Applications* (CRC Press).

Yoshii, K., Goto, M., Komatani, K., Ogata, T., and Okuno, H. G. (2006). Hybrid collaborative and content-based music recommendation using probabilistic model with latent user preferences, in *ISMIR '06*, pp. 296–301.

Zangerle, E., Pichl, M., Gassler, W., and Specht, G. (2014). #Nowplaying Music Dataset: Extracting Listening Behavior from Twitter, in *WISMM Workshop at MM '14*, pp. 21–26.

Zhang, M. and Hurley, N. (2008). Avoiding Monotony: Improving the Diversity of Recommendation Lists, in *RecSys '08*, pp. 123–130.

Zhang, Y. C., Séaghdha, D. O., Quercia, D., and Jambor, T. (2012). Auralist: Introducing Serendipity into Music Recommendation, in *WSDM '12*, pp. 13–22.

Zhou, Y., Wilkinson, D., Schreiber, R., and Pan, R. (2008). Large-Scale Parallel Collaborative Filtering for the Netflix Prize, in *AAIM '08*, pp. 337–348.

Ziegler, C.-N., McNee, S. M., Konstan, J. A., and Lausen, G. (2005). Improving Recommendation Lists Through Topic Diversification, in *WWW '05*, pp. 22–32.