CORA Cork Open Research Archive
Cartlann Taighde Oscailte Chorcaí

Title	Surveying more than two decades of music information retrieval research on playlists
Authors	Gabbolini, Giovanni;Bridge, Derek G.
Publication date	2024
Original Citation	Gabbolini, G. and Bridge, D. (2024) 'Surveying more than two decades of music information retrieval research on playlists', ACM Transactions on Intelligent Systems and Technology, 15(6), pp. 1–68. https://doi.org/10.1145/3688398
Type of publication	Article (peer-reviewed)
Link to publisher's version	10.1145/3688398
Rights	© 2024, the Authors. This work is licensed under a Creative Commons Attribution International 4.0 License. - https://creativecommons.org/licenses/by/4.0/
Download date	2026-11-02 09:03:56
Item downloaded from	https://hdl.handle.net/10468/17343



UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh



Surveying More Than Two Decades of Music Information Retrieval Research on Playlists

GIOVANNI GABBOLINI and DEREK BRIDGE, Insight Centre for Data Analytics, School of Computer Science and IT, University College Cork, Cork, Ireland

In this article, we present an extensive survey of music information retrieval (MIR) research into music playlists. Our survey spans more than 20 years, and includes around 300 papers about playlists, with over 70 supporting sources. It is the first survey that is self-contained in the sense that it combines all the different MIR research into playlists. It embraces topics such as algorithms for automatic generation, for automatic continuation, for assisting with manual generation, for tagging and for captioning. It looks at manually constructed playlists, both those that are constructed for and by individuals and those constructed in collaboration with others. It covers ground-breaking research into enhancing playlists by cross-fading consecutive songs and by interleaving consecutive songs with speech, similar to what happens on a radio show. Most significantly, it is the first survey that can fully incorporate the paradigm shift that has taken place in the way people consume recorded music: the shift from physical media to music streaming. This has wrought profound changes in the size of music collections available to listeners and thus the algorithms that support the construction, curation and presentation of playlists and the methods adopted by users when they also construct, curate and listen to playlists.

CCS Concepts: • **Information systems** → **Multimedia streaming**; **Collaborative and social computing systems and tools**; **Personalization**;

Additional Key Words and Phrases: playlists, music, information retrieval, recommender systems

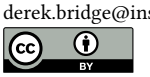
ACM Reference format:
Giovanni Gabbolini and Derek Bridge. 2024. Surveying More Than Two Decades of Music Information Retrieval Research on Playlists. *ACM Trans. Intell. Syst. Technol.* 15, 6, Article 114 (November 2024), 68 pages.
https://doi.org/10.1145/3688398

1 Introduction

The technological revolutions of the late 20th century, such as the Internet, have shaped many parts of our contemporary lives, including how we interact with recorded music.[1] In the digital era we are living in, music streaming services are one of the most popular ways to interact with music.

[1]Music listening can also happen at live performances. But the focus of this article is on recorded music.

Giovanni Gabbolini is now at Apple Inc.
This work has been supported by a grant from Science Foundation Ireland (SFI) under Grant Number 12/RC/2289-P2, which is co-funded under the European Regional Development Fund.
Authors' Contact Information: Giovanni Gabbolini, Insight Centre for Data Analytics, School of Computer Science and IT, University College Cork, Ireland; e-mail: giovanni.gabbolini@insight-centre.org; Derek Bridge (corresponding author), Insight Centre for Data Analytics, School of Computer Science and IT, University College Cork, Ireland; e-mail: derek.bridge@insight-centre.org.



This work is licensed under a Creative Commons Attribution International 4.0 License.

© 2024 Copyright held by the owner/author(s).
ACM 2157-6912/2024/11-ART114
https://doi.org/10.1145/3688398

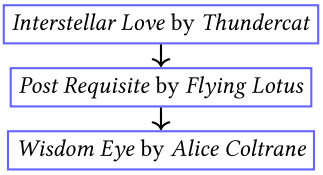


Fig. 1. A playlist of three songs.

According to the global music report of the **International Federation of the Phonographic Industry (IFPI)**, streaming accounted for 17.5 billion dollars of revenue in 2022, which is nearly 67% of the total global recorded music industry revenues for that year [160]. In the context of this article, we consider music streaming services as the default medium to interact with recorded music. In exchange for a monthly subscription fee, or for free in exchange for exposure to advertisements, music streaming services allow their users to access an enormous catalogue of music, from different devices, such as smartphones and personal computers, at any time.

The abundance of available music raises the risk that users of streaming services will be overwhelmed [140]. For example, at the time of writing (early 2024), the music streaming services Spotify[2] and Deezer[3] have catalogue sizes of respectively 100 and 120 million tracks. The need, therefore, for efficient modalities of music access becomes apparent. In this scenario, playlists, which can be defined as "sequence[s] of tracks intended to be listened to together" [306], have become one of the preferred ways of accessing music. An example of a three-song playlist is in Figure 1. Listeners use playlists to structure their listening, efficiently accessing the right music at the right time [112].

The importance of playlists is evident by looking at the home page of the music streaming service Spotify, which features playlists as a prominent element, as shown in Figure 2. Notice that the playlists of Figure 2 are personalized, that is they are chosen to meet the user's preferences and requirements. The value of playlists is also highlighted by several statistics: in 2016, playlists accounted for 31% of music streaming time among listeners in the USA, which is more than albums (22%), but less than single tracks (46%) [306].

Playlists are created for users by professional curators and algorithms, and by users for themselves and other users, for convenience and self-expression [358]. A study conducted in 2017 revealed that 58% of users in the USA create their own playlists, and that 32% of users share their playlists with other users [306]. In total, the music streaming service Spotify was hosting more than four billion playlists in 2021.[4] The statistics that we have quoted reveal the commercial value of playlists. Partly in consequence, notable research efforts have ensued during the last two decades. Much of the research on playlists is concerned with automatic generation of playlists, e.g. [111, 127, 343, 351]. Other work looks into how humans manually construct playlists, e.g. see [140, 281].

In this article, we present an extensive survey of **music information retrieval (MIR)** research into music playlists. It spans more than 20 years, and includes around 300 papers about playlists. It is the first self-contained survey to include all the different MIR research on music playlists, combining in particular the two topics mentioned above (automatic generation and human construction). It also includes research that, as far as we know, has not been surveyed before, such as research into playlist tagging, playlist captioning, and enhancing the delivery of consecutive songs through cross-fading or spoken links.

[2] https://spotify.com
[3] https://deezer.com
[4] https://backlinko.com/spotify-users

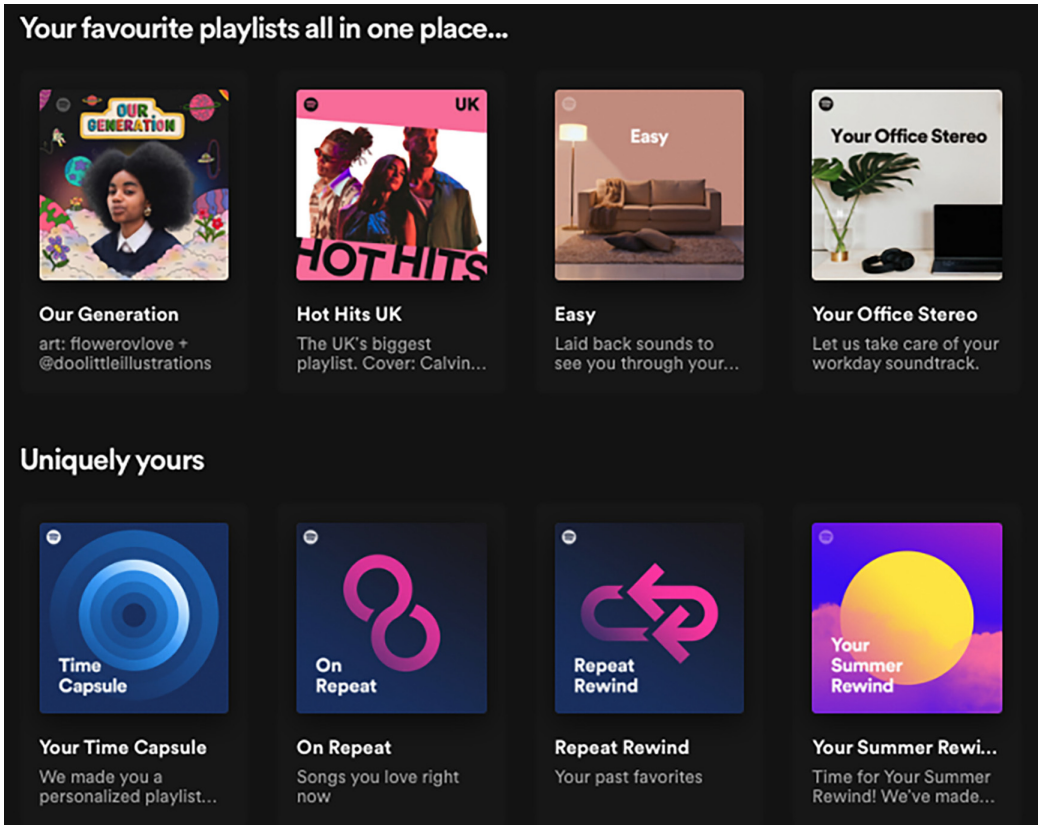


Fig. 2. The home page of the music streaming service Spotify, which features playlists as a personalized and prominent element. Picture taken 15 June 2022.

The remainder of this article is organized as follows. In Section 2, we formalize the concept of playlists, we divide the research on playlists into several topics, and we compare it with other related research. In Section 3, we summarize existing surveys on playlists, we describe the relationships between these and our own survey, and we highlight the contributions that ours make. In Section 4, we describe how we select relevant papers for inclusion. In the remaining sections, we present those relevant papers, by dividing them into the research topics we identify in Section 2. In particular, Section 5 reviews work on automatic playlist generation, and Section 6 surveys the field of **manual playlist generation (MPG)**. Section 7 looks at ways researchers are enhancing playlists by cross-fading consecutive songs, interleaving songs with speech, assigning them descriptive tags, and automatically captioning them. Section 8 offers possible directions for future research. Section 9 furnishes some conclusions.

2 Definitions and Research Landscape

This survey is about the consumption of recorded music, which, for conciseness, we refer to simply as "music." We refer to a person who is interacting with a music application as a "user," irrespective of whether they are listening to music, constructing a playlist, organizing their playlists, or any other activities that the application allows. For a music streaming service, "user" refers not only to a subscriber to the service but also anyone who uses a free version of the service.

In this article, we define the concept of playlists as follows:

Definition 2.1. A playlist is a sequence of songs, intended to be listened to together.

Definition 2.1 is equivalent to what we find in notable references, e.g. [107, 306, 364]. Some authors define playlists in simpler terms; for example, Bonnin and Jannach write that a playlist is "a sequence of songs" [39]. We fear that this simpler definition is too broad. For example, a random sample of songs from a song catalogue satisfies the simpler definition, but does not correspond to the concept of playlist that is commonly intended, i.e. a sequence of songs organized according to some principle [81]. Note also that Definition 2.1 entails a notion of ordering, that is a playlist is made of songs in a specific order. However, the importance of song order in music playlists is a debated topic. For example, Andric and Haus offer some evidence that the order does not matter [10], while De Mooij and Verhaegh find that the order does matter [86]. We further discuss this topic in Section 6.1.2.

One final remark about Definition 2.1 is its use of the word "song." Some authors draw a distinction between "songs" and "tracks." The latter are specific recordings of the former; multiple recordings (tracks) can exist for a single song, e.g. where the song is recorded by different artists. Using this distinction, Definition 2.1 would need to be changed to refer to a sequence of tracks, as in [306]. However, similar to the way we use the word "music" to only refer to recorded music, we have chosen to use only the word "song" from now on[5].

In the remainder of this section, we first present several research topics on playlists; then, we briefly relate the research on playlists to research on music **recommender systems (RSs)** and to the research on sequence-aware and session-based RSs in other domains, such as e-commerce.

2.1 Playlist Research Topics

We divide the research on playlists that we survey into several topics. We organize these topics with a diagram in Figure 3, and we briefly introduce them in the following sections.

2.1.1 Playlist Generation. Research on playlist generation is concerned with the construction of playlists. We adopt the definition of playlist generation presented in [39]:

Definition 2.2. Given (1) a catalogue of songs, (2) background knowledge, and (3) some target characteristics of the playlist, construct a sequence of songs fulfilling the target characteristics in the best possible way.

The target characteristics of the playlist are the organization principles which make a playlist of a sequence of songs to be listened to together, which is consistent with Definition 2.1, while the background knowledge allows the agent which carries out the construction to select the songs from the catalogue so as to match the target characteristics. For example, a target characteristic may be a playlist for a beach day, the background knowledge may be some notion of what musical genres are more suited for a beach day, along with some modeling of the users' musical tastes, and the catalogue of songs may be all the music hosted in a music streaming service.

Depending on the agent which carries out the construction, research in playlist generation can be divided into automatic and manual:

Automatic Playlist Generation (APG). In APG, a playlist is constructed automatically for the user by an algorithm. For example, Flexer et al. present an algorithm for creating a playlist that progresses smoothly from a specified start song to a specified end song [111]. APG is a major research topic,

[5]In a similar vein, we do not intend by using the word "song" to confine attention to music in which there is singing. Our use is intended to cover the whole range of musical pieces that are found on music streaming services.

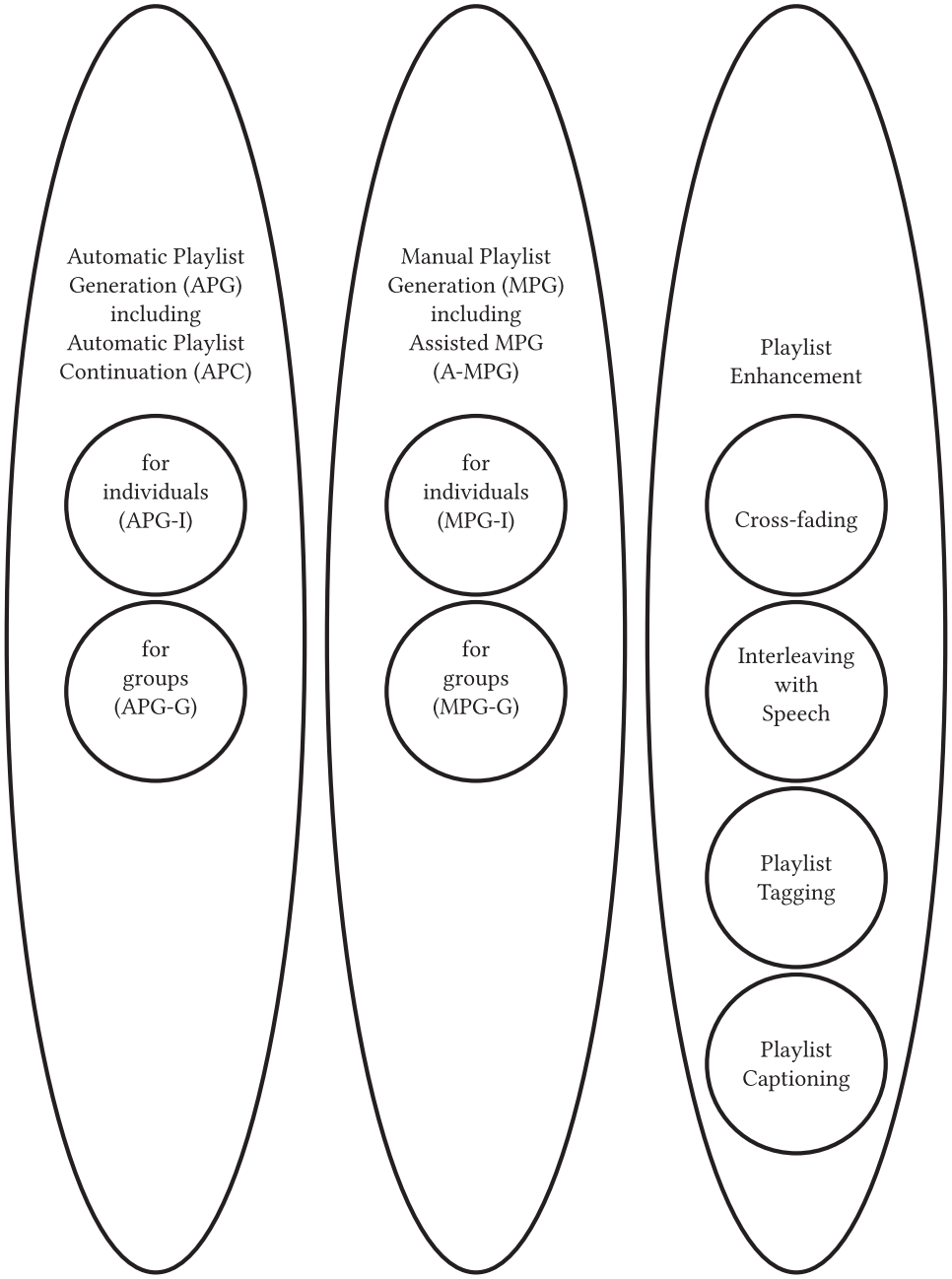


Fig. 3. Diagram of the research topics we survey.

counting hundreds of publications. The interest in APG is motivated by the fact that MPG (below) can be experienced as a tedious, time-consuming activity, and it may require special background knowledge [39]. APG research dominates our survey, which is one of the reasons we review it first.

One special case of APG is **automatic playlist continuation (APC)**. In APC, the goal is to extend an existing playlist with further songs. In terms of the definition of playlist construction above, the existing playlist provides the target characteristics since the goal is to add songs that are compatible with the original playlist. We discuss this further in Section 5.1.2.

The above characterization of APG tacitly assumes that playlists are constructed for a single user. However, there exists a special subset of work on APG which deals with the case in which playlists are constructed for a group of users. Where we wish to distinguish these from each other, we use **APG for individuals (APG-I)** and **APG for groups (APG-G)**.

Manual Playlist Generation (MPG). In MPG, the playlist is constructed manually by the user. MPG is an activity that dates before the digitization of music. For example, in the 1980s it was common to compile mix-tapes and to exchange those mix-tapes with other users [113]. MPG is an activity which is important also nowadays because users in streaming services frequently create playlists. They do this for two main reasons:

(1) Convenience [112]. Users create playlists, and give them a title, so as to have a personal interface to their music, built inside the streaming service. This helps users to access the right music at the right time, in a fast way.
(2) Self-expression [358]. In music streaming services, all users have access to the same catalogue of music, without actually owning any of it. But playlists allow users to select and organize music, giving them the opportunity to express their personality and musical tastes.

A study conducted in 2017 revealed that 58% of users in the USA of the Spotify music streaming service create their own playlists, and that 32% of these users share their playlists with other users [306].

Some research in MPG considers the scenario in which the user is assisted by an algorithm while manually constructing the playlist [96]. For example, in [182] the user is assisted by an algorithm that recommends the next song to add as the user constructs the playlist, while in [352] an algorithm organizes songs in a color map, and the user can create playlists by drawing on the map. We refer to these approaches as **assisted manual playlist generation (A-MPG)**. A-MPG combines MPG and APG, as the user is still in control of the playlist construction process, but the task is facilitated by an algorithm.

Just as work on APG can be for individuals (APG-I) or groups (APG-G), we can divide the work on MPG: MPG-I is where playlists are constructed for and by a single user; MPG-G is where they are constructed for and by a group of users.

2.1.2 Playlist Enhancement. Research on playlist enhancement is concerned with automatically decorating existing playlists with additional content, so as to increase such things as the enjoyability or accessibility of the playlist. The first two enhancements that we consider deal with the presentation of consecutive songs. One is based on mixing consecutive songs, so as to obtain a continuous music flow. The other is based on interleaving pairs of consecutive songs with speech, similar to what happens on a radio show.

The remaining two enhancements that we consider look into automatically describing playlist content at a semantic level that can be understood by humans. Playlist description is a useful way to cope with content overload. As we have already noted, music streaming services feature billions of playlists created by users, professional editors, or algorithms [88]. Playlist description allows for effective and automated organization and access to playlists [69]. One way to describe playlists is by using tags, which are closed-vocabulary short textual descriptions, naturally limited in expressiveness. Another way is by captioning a playlist with well-formed natural language, which is expressive but more complex to generate.

2.2 Related Research Topics

The MIR research on playlists that we describe in this survey has relationships with other research, most notably with research into RSs, and especially with the topics of **sequence-aware recommender systems (SARSs)**, **session-based recommender systems (SBRSs)**, and **music recommender systems (MRSs)**.

2.2.1 SARSs and SBRSs. SARSs are used to recommend sequences of items, typically by generating item after item, iteratively. As well as recommending products in the e-commerce domain, SARSs have been adopted in the tourism domain for recommending the next point-of-interest to visit in a tour. We refer the reader to the survey in [291].

In SBRSs, users interact with items within a continuous but bounded period of time—usually a fairly short period. The RS must take into account the user's short-term preferences, which may be specific to the session. Like SARSs, SBRSs can be found in e-commerce domains: in one session, a user might want a hotel for a business trip; in another session, the same user might want a hotel for a family vacation. We refer the reader to the survey in [354].

SARSs and SBRSs are different: in the former, order is significant; in the latter, the time period and its short-term preferences are paramount. But they often coincide. This happens when the order of item interactions within a session is significant.

Research in APG is very much related to the research in SARSs and SBRSs. A playlist typically has characteristics in common with a session: a playlist has a bounded duration and may be designed to satisfy short-term preferences, such as the user's mood or activity. Furthermore, Definition 2.1 states that playlists are sequences, and hence ordered—although, again, we refer the reader to Section 6.1.2 for further discussion of this. Many algorithms for APG can be seen as applications of SARSs and SBRSs in the music domain. This is especially true of the work in APC that we mentioned earlier: in the same way that many of these RSs recommend the item that comes next in a sequence and/or session, APC systems recommend the next song to add to the playlist based on the songs already in the playlist; see Section 5.1.2.

However, the music domain is different from other domains, for a number of reasons, as argued by Schedl et al. in their survey on MRSs [305]. Songs are different from other items because they are consumed in a relatively short time and because they are often consumed more than one time. Owing to these peculiar characteristics, much research effort has been put into building RSs explicitly for the music domain, which are of interest for this survey.

2.2.2 MRSs. MRSs are RSs that work in the music domain, tackling tasks such as the recommendation of a personalized selection of songs, artists, or albums [305]. Since algorithms for APG are also tasked with recommending a selection of songs, i.e., the playlist, they can be seen as MRSs. However, APG algorithms are different from general MRSs because the selection of songs must satisfy some additional soft constraints, such as matching some user-defined target characteristics, and general characteristics, such as the right level of song diversity/coherence, as well as a non-jarring song ordering; see Section 6.1.2.

3 Related Surveys and Contributions

MIR research on playlists is a two-decades-old research field: the oldest of the citations to playlist research that we include dates back to 1997 [86]. We are not the first authors to survey the literature. To the best of our knowledge, there are two others that are closely related to ours:

(1) Bonnin and Jannach's "Automated Generation of Music Playlists: Survey and Experiments," published in 2014 [39]; and
(2) Dias et al.'s "From Manual to Assisted Playlist Creation: A Survey," published in 2017 [96].

Survey (1) is specific to APG, as it presents algorithms for APG based on three attributes: (a) what background knowledge the algorithms employ; (b) how the target characteristics of the desired playlist can be input to the algorithms; and (c) the type of algorithm. Each attribute admits a number of possible categories; for example, the background knowledge can be content-based data, metadata and expert annotations, social web data, and usage data. We give more details in Section 5.1. We borrow some ideas from (1), as we also characterize APG algorithms based on the same three attributes. However, we include novel categories of those attributes, so as to better accommodate the work that we survey but that is not cited in (1). For example, we cover APG algorithms based on (neural) sequence modeling and on **reinforcement learning (RL)**, which have grown in significance since 2014.

Survey (2) is mainly about A-MPG but, in order to better position A-MPG algorithms in the literature, it also includes a discussion of MPG and APG. The part of (2) on APG is a subset of what is included in (1), while the part of (2) on MPG briefly discusses characteristics of manually constructed playlists, such as recurrent playlist themes, and notable manual construction styles. Nevertheless, the main contribution of (2) is a review of A-MPG algorithms. It presents several algorithms for A-MPG, all based on visualizations, i.e., that assist the user in the manual construction of a playlist by using visualizations of the song catalogue. Survey (2) identifies several categories of A-MPG algorithm, based on the kind of visualization that is employed: maps, graphs, dots, and radar. Our survey also includes the A-MPG algorithms that were covered by (2), as well as other work published from 2017 onward. However, the recent work does not belong to any of the categories proposed in (2). For example, recent work provides lists of song recommendations for addition to the playlist, which is not a catalogue visualization of any kind. Hence, we propose a novel categorization of A-MPG algorithms, to cover both recent and non-recent work. Specifically, we divide algorithms into two categories: visualization and recommendation.

Despite some overlap, our survey is substantially different from (1) and (2), as we give a fresh look at APG and A-MPG research, by including relevant work published after the publication of (1) and (2). A fresh look is needed. In recent years, we have witnessed a change of paradigm in how people access recorded music. It has shifted from physical media to music streaming. According to the IFPI [160], the share of revenue coming from music streaming was nearly 10% of the total revenue for recorded music in 2013, nearly 30% in 2016, approaching 57% in 2019, and nearly 67% in 2022. The rise of music streaming has had a dramatic impact on research on music playlists, especially in terms of sources of data and song catalogues.

In the case of sources of data, streaming services allow for the collection of enormous quantities of usage data. Examples of usage data are manually constructed playlists, as well as listening logs where the streaming service records the actions of its users when they are listening to music. A few years after publication of (1), large datasets of usage data became available for researchers to use. One example is the Million Playlists Dataset (MPD), released in 2018, which contains manually constructed playlists, and is one order of magnitude larger than the datasets commonly employed in the research included in (1) [39]. Others are the 30Music dataset [332], which contains nearly 60,000 playlists; the AOTM dataset [237], comprising over 100,000 playlists; and the ALF-200k dataset [365], containing more than 11,000 playlists. The availability of usage data has dramatically changed research in APG. The recent APG algorithms that we survey here, post-dating the ones in (1), rely mainly on usage data, while few of them employ content-based data, metadata and expert annotations, which is the predominant background knowledge used in APG algorithms included in survey (1) [39] (see Table 1 for more details). The availability of large quantities of usage data allows for the use of sophisticated machine learning algorithms, which are known to provide satisfactory results only when large quantities of data are available [200]. Many of the recent APG algorithms that we survey here, post-dating those in (1), use deep learning for computing

Table 1. Organization of APG-I Algorithms, Based on the Category of Background Knowledge They Use, and Dividing Those Published up to 2013 from Those Published from 2014 Onwards

Category	Up to 2013	From 2014
Content-based data	[14, 15, 21, 36, 43, 46, 55, 67, 74, 75, 98, 100, 108, 111, 123, 135, 153, 174, 194, 207, 226, 228, 233, 237, 244, 258, 259, 264, 284, 285, 294, 303, 309, 349, 360]	[1, 12, 32, 40, 49, 64, 71, 104, 115, 117, 129, 136, 157, 159, 161, 162, 165, 171, 192, 220–222, 273, 286, 299, 300, 324, 328, 336, 337, 339, 351]
Metadata and expert annotations	[5, 6, 26, 35, 55, 56, 66, 75, 79, 145, 153, 155, 158, 174, 207, 237, 243, 251, 252, 263, 274, 277, 283, 294, 349, 370]	[12, 124–126, 156, 157, 164, 165, 171, 185, 232, 286, 298, 301, 335, 346, 351]
Social web data	[56, 75, 98, 108, 130, 144, 194, 237, 246, 251, 252, 284, 293, 302, 309, 315]	[32, 170, 171, 181, 185, 262, 286, 304, 337, 339]
Usage data	[3, 18, 38, 61, 130, 144, 158, 233, 237, 246, 292, 367, 370]	[4, 12, 32, 40, 49, 59, 71, 73, 87, 102, 124, 126, 156, 157, 159, 162, 164, 166, 170–172, 185, 187, 189, 191, 199, 229, 232, 245, 264, 265, 283, 298–301, 312, 326, 330, 331, 333, 335–337, 339–342, 345, 346, 351, 361–363, 369, 372]

Those published up to 2013 are also reviewed by Bonnin and Jannach in their 2014 survey [39]. The categories are the ones used by Bonnin and Jannach.

embeddings and for sequence modeling and RL, while fewer of them rely on music similarity, which was the predominant approach of APG algorithms included in (1) [39] (see Table 3 for more details).

In the case of song catalogues, in the pre-streaming era users had access to small personal collections of music, which typically consisted of, at most, thousands of songs. With music streaming, users have access to large song catalogues, consisting of millions of songs. As such, recent APG algorithms need to scale with catalogue size, which is not always the case for many APG algorithms included in survey (1), especially those that work by solving expensive discrete optimization problems with non-linear complexity. As a result, we encounter almost no discrete optimization APG algorithms in the recent research, post-dating (1), see Table 3. The shift from small to large song catalogues has also impacted research on A-MPG, as the goal of recent A-MPG algorithms, post-dating (2), is to assist the user in the manual construction of playlists by providing recommendations for songs to add to the playlist. A-MPG work included in survey (2), instead, mainly focuses on visualizations that assist the user in the manual construction of playlists by visualizing the song catalogue in a map or in a graph, and which do not scale to catalogues of millions of songs.

Finally, we discuss a substantial amount of work that was not in surveys (1) and (2), such as the research on **group automatic playlist generation (APG-G)**, on **group manual playlist generation (MPG-G)**, as well as the research on playlist enhancement. We also include an extensive account of research on MPG. Survey (2) also contains research on MPG, but it gives only a partial account, from just five papers. We, by contrast, review over 50 papers on MPG, and we cover MPG topics not covered in (2), such as a detailed discussion of the role of song diversity and homogeneity when manually selecting songs.

4 Research Methodology

We collect relevant papers by following a well-defined procedure, where we first define a search string, and then we review all the relevant papers that match the search string, as well as all the relevant papers contained in the references of those papers, in "snowballing" fashion. For example, we retrieve a relevant paper p_1 and we scan its references. If we find a relevant paper p_2 in the references of p_1, we review p_2 and we scan its references. If we find a relevant paper p_3 in the references of p_2, we review p_3 and we scan its references, and so on, until we run out of relevant papers.

We consider a paper to be relevant by performing an initial scan of its contents, where we search for the keyword "playlist." If the paper never mentions "playlist," it is safe to assume that it is not relevant. If the paper does mention "playlist," then we look at it more carefully and we consider it to be relevant if playlists are the main topic of investigation of the paper. For example, a paper that creates a dataset of songs from a dataset of playlists and then extracts song embedding representations to power a song RS is considered not relevant since the playlists are incidental to the work.

We defined the search string by scanning the proceedings of the **International Society for Music Information Retrieval Conference (ISMIR)**,[6] which is the premiere venue for research in MIR. In particular, based on papers published in ISMIR, we crafted a search string that includes keywords from the titles of those papers.[7] We use the string to search the academic aggregator DBLP.[8]

In total, our survey reviews around 300 relevant papers, and it also cites over 70 additional supporting sources.

5 APG

APG is the most popular topic within research on playlists. Research in APG is concerned with the design, implementation, and evaluation of algorithms for constructing playlists. In this section, we survey the literature on APG, presenting first algorithms to generate playlists for individual users (APG-I, Section 5.1), and then algorithms to generate playlists to be listened to by groups of users (APG-G, Section 5.2).

5.1 APG for Individual Users

Bonnin and Jannach [39] surveyed the literature on APG-I in 2014, organizing APG-I algorithms based on three attributes: (1) what background knowledge they employ; (2) how the target characteristics of the desired playlist can be input to the algorithm; and (3) the algorithm type, e.g., content-based, **collaborative filtering (CF)**, and so on. According to Definition 2.2, three inputs

[6] https://ismir.net/

[7] The search string we use is "playlist continuation|continuing|expansion|expanding|creation|creating|recommendation|recommender | recommending | generation | generating | user | study | trial | evaluation | evaluating|interview|interviewing|sequencing|sequence|representation|representing|caption|captioning".

[8] https://dblp.org/

are required for an APG-I algorithm: a catalogue of songs, background knowledge, and some target characteristics of the desired playlist. The first two of Bonnin and Jannach's attributes correspond with two of the inputs that are mentioned in the definition of APG-I. In this section, we adopt Bonnin and Jannach's organization: we also characterize APG-I algorithms based on the same three attributes, but we complement their work, as we incorporate the newer literature up to the date of writing.

5.1.1 Background Knowledge. The background knowledge is the information used to choose the songs from the catalogue in order to construct a playlist that matches the target characteristics. The background knowledge should be represented in some machine readable form, in such a way that it can be used by algorithms. In their 2014 survey on APG-I, Bonnin and Jannach [39] identify several categories of background knowledge. In this section, we review those categories, while adding fresh details from the more recent work that we have surveyed. We organize APG-I algorithms based on their background knowledge in Table 1. We highlight differences from the research surveyed by Bonnin and Jannach [39] to the research we exclusively survey by dividing algorithms published up to 2013 from those published from 2014 onward.

Content-Based Data. Researchers in the field of MIR have been concerned for a long time with extracting information, or *features*, from the music audio signal, a research topic which is often referred to as content-based MIR [249].[9] These content-based features can be high-level, such as the emotions evoked by a musical piece [324, 366], its genre [76], timbre [279], chords [101], pitch [373], and **beats per minute (BPM)** [307], or low-level, such as representations of the audio signal, for example **mel frequency cepstrum coefficients (MFCC)** [371], or such as learned embedding representations, as extracted, for example, by **convolutional neural networks (CNNs)** [162, 273]. Often, low-level features are used for extracting high-level features, e.g. see [72, 287, 359]. We refer the reader to [249] for a survey of content-based MIR.

Some APG-I algorithms rely on high-level content-based features. For example, Griffiths et al. extract the emotion evoked by songs, and construct a playlist that matches the emotion of the user, which is extracted by using several sensors [135]. The same approach is taken in [136, 324]. The work in [40] is similar to the above, except that users manually input their current emotion, e.g., melancholy. Liebman et al. [221, 222] propose a RL algorithm for APG-I in which songs are represented as vectors containing timbre, pitch, BPM, and statistics thereof.

Some other content-based algorithms rely on low-level content-based features. For example, Pohle et al. compute the similarity between songs in the catalogue based on an MFCC representation, to create a playlist in which consecutive songs are as similar as possible, so as to guarantee a coherent listening experience [285]. A similar approach is followed by [21, 111, 226, 228]. In [71, 162], instead, song representations extracted by a CNN are used as input to a **recurrent neural network (RNN)**, so as to predict the next song in the playlist.

Metadata and Expert Annotations. Following [39], we use the word metadata to refer to any information describing the playlist or its songs that is not derived from the audio signal. An example of playlist metadata is the playlist title or caption, when assigned by an expert. (When assigned by an end-user, it might be preferable to classify them as examples of social web data, like user tags—see below.) Examples of song metadata are the year of release, the record label, the lyrics, and

[9]The term content-based has different meanings in different research communities. For example, in the RSs community, content-based features are any type of feature describing an item, such as the song lyrics or its musical genre [250]. We position our survey more in line with the MIR community, in which content-based features typically refer to those features which are extracted from the music audio signal [249].

the genre,[10] among others [118]. Usually, experts manually annotate songs with their metadata. A notable example is the Music Genome Project [50], a database of songs and their metadata created and maintained by experts employed by the Pandora music streaming service.[11]

Different types of metadata are sometimes represented in a single structure, for example in a knowledge graph, where nodes represent both songs and also heterogeneous types of metadata [103], and edges express the relationships between the songs and the metadata, and the metadata with other metadata. For example, song names, album names and years can be represented as nodes, and a "belongs to" edge can link a specific song name to its album name, and a "released in" edge can link the album name to its year of release [118]. Knowledge graphs are also used in [87, 165, 237, 251, 252, 333]. Edges in a knowledge graph might also represent relationships between classes, subclasses, and instances, e.g. between genres and subgenres. Indeed, ontologies and taxonomies offer an alternative to knowledge graphs, placing the focus on classes, subclasses, instances, and their properties. For example, Ben-Elazar et al. use a taxonomy of musical genres [32].

One way to use metadata in APG-I is by allowing users to specify constraints on the metadata and then generate a playlist that satisfies those constraints. For example, in [277] users can input constraints on the song genre, release year, and length (in seconds), and an optimization algorithm is used to generate a playlist which satisfies these constraints. The same strategy is followed in [5, 15, 55, 155, 263, 275, 276].

Social Web Data. Social web data is data shared online by Internet users. Following [39], we list three types of social web data:

User tags. A user tag[12] is a free text annotation that a user applies to a musical item, e.g. a song or an artist [201]. User tags can be very rich and varied, as they can cover a wide range of different topics, such as musical genres (e.g., "rock"), years (e.g., "90s"), countries (e.g., "Ireland"), activities (e.g., "chill"), seasons (e.g., "summer"), among others.

Ratings. A rating is a piece of explicit user feedback for a musical item, often expressed in a 1-to-5 rating scale or as a "like" or "dislike" judgment. The usage of ratings as background knowledge is becoming less and less common, as ratings are too difficult to gather for the majority of the user base [305]. In particular, we do not encounter any work that uses ratings as background knowledge in the literature from 2014 to the date of writing.

The social graph. A social graph connects people by different relationships, such as "friend" or "spouse," and to musical items, e.g. person x "likes" artist y, in social networks such as Facebook.[13] Social graphs are sometimes used as background knowledge, under the assumption that people who are closely connected in the graph have similar musical tastes [130]. A playlist for a user can be constructed, for example, by including music that is liked by the user's friends, giving an automated version of word-of-mouth recommendation.

[10]Note that genres appear in our classification of background knowledge both as examples of content-based and as examples of metadata. This is because song genres can be extracted by an algorithm from the audio signal, or they can be assigned by experts. For example, in early 2000s work, small catalogues of songs were manually annotated with their musical genres [263]; in recent work, accurate content-based MIR algorithms are often employed to extract the musical genres of large-scale song catalogues [287].

[11]https://pandora.com

[12]The word "tag" is ambiguous. In some work, it is used to indicate free text, e.g. [201]; in other work, it is used to indicate an item of text drawn from a fixed vocabulary, e.g. [68]. This survey needs to use the word "tag" in both of its meanings. For example, in this section, tags are free text, while in Section 7.3 tags are drawn from a fixed vocabulary. In order to make clear which of the two meanings we intend, we use "user tag" for free text, and we use simply "tag" where there is a fixed vocabulary.

[13]https://facebook.com

Usage Data. In streaming services, usage data record interactions between users and musical items.

Listening logs. Listening logs record the songs a user listens to, including those that they skip, those that they listen to completion, and those that they download. As such, listening logs provide indications about user tastes. For example, it is common for algorithms to interpret a skip as an indication of a song that the user does not like [32, 66, 100, 157, 192, 265], although, of course, it could just signal that the song does not suit the user's context, such as their mood or activity. Listening logs can be used to compute embedding representations. One strategy is to rely on the word2vec algorithm [240], by treating songs as words, and listening logs as phrases, by analogy with natural language.

Popularity. Usage data gives a clear indication of the popularity of musical items, e.g., of songs. We can, for example, simply count the occurrences of each song in the listening logs of all the users. Popularity is sometimes used as background knowledge for building simple but effective heuristics for APG-I. For example, Bonnin and Jannach show that it is possible to build high-quality playlists by simply including the most popular songs made by artists similar to the artists that the user likes [38]. Moreover, popularity can be employed as a fallback strategy for estimating the musical tastes of new users, i.e., those that are new to the streaming service.

Manually constructed playlists. Users frequently create playlists for convenience [112] and self-expression [358]. These user playlists can be used as background knowledge for creating new playlists. For example, McFee and Lanckriet learn song-to-song transition probabilities based on a database of user playlists, and they use these probabilities to generate new playlists [238]. Manually constructed playlists can also be used to compute embedding representations; see our discussion of listening logs, where we mentioned the word2vec algorithm as a possible way of computing these embeddings.

Discussion. The categories of background knowledge we review above differ in their availability, consistency, and abundance:

Availability. The availability of some background knowledge may not be guaranteed for all the songs in the catalogue. For example, recently added songs may have no user tags, or may occur few times, or never, in listening logs or in manually constructed playlists. The same goes for "long-tail" songs [195], i.e., those songs which are rarely listened to, that constitute the large majority of the catalogue [51]. The unavailability of background knowledge for such portions of the catalogue leads to biases against new songs (the cold-start problem [89]) and against long-tail songs (popularity bias [169]). In fact, algorithms cannot evaluate a song for inclusion in a playlist if there is no background knowledge to match the song to the target characteristics of the playlist. Content-based data is the only category of background knowledge which can be available for every song in the catalogue, as content-based data is extracted from the song audio itself. As such, content-based data allows for the construction of "fair" algorithms, in the sense that they can select any song in the catalogue for inclusion in the playlist.

Consistency. Some background knowledge may be noisier than other background knowledge. For example, the level of noise in metadata is often low because metadata annotations are typically made manually by domain experts. Nevertheless, inconsistencies in metadata may exist, especially because some metadata is not objective. For example, Flexer et al. find that different annotators may disagree on the musical genre of songs [110]. Similarly, the level of noise in content-based data also tends to be low because content-based data is extracted by automatic procedures. Nevertheless, inconsistencies in content-based data may arise because those automatic procedures are never

100% accurate. For example, predicting the tempo of a song is challenging and the results can be inaccurate [58, 278].

Usage and social web data are typically much noisier than content-based and metadata, as they record the unpredictable behavior of Internet users. For example, Lamere analyzes a dataset of user tags and finds misspellings, spelling variants and synonyms among user tags, as well as user tags with little to no relevance to music, such as the user tag "random" [201]. Similarly, Hagen finds that manually constructed playlists often do not have clearly defined target characteristics but may be used as a randomly arranged container of the user's favorite music [140].

Abundance. Usage and social web data are by far the most abundant category of background knowledge, as they are generated in large quantities by billions of Internet users every day. Content-based and metadata are less abundant as their extraction depends, respectively, on computationally bounded procedures [308] and on the expensive annotation work of domain experts.

Using one category of background knowledge rather than another influences the quality of the generated playlists. For example, there is some evidence that algorithms relying solely on content-based data produce playlists of low quality, especially when compared with other types of systems, e.g. those which rely on usage data [321] or metadata [338]. However, it is wrong to consider one category of background knowledge to be superior to another. A more correct view is to consider them as complementary: while content-based data can help to create coherent playlists in terms of acoustic properties, usage data gives information about the musical tastes of the user, allowing the creation of personalized playlists. Hence, it is common to combine different categories of background knowledge. There are several papers that show how the quality of generated playlists is enhanced by making effective use of more than one category of background knowledge, e.g. [32, 171, 237]. One way to readily include different sources of background knowledge is to organize them in a unifying structure, for example a knowledge graph (described earlier) [103]. In addition to representing songs and their metadata, knowledge graphs can represent listening logs: nodes for users would link to nodes for the songs they listened to [261].

If we refer back to Table 1, we can see the differences between the research up to 2013 (the *first period*), which was already surveyed by Bonnin and Jannach [39], with respect to the research from 2014 onward (the *second period*). The majority of algorithms from the *first period* rely on content-based and metadata for their background knowledge, especially because the song catalogue sizes before the streaming era allowed for the manual annotation of songs or the extraction of content-based data. In the *second period*, when streaming became the prevalent type of music access [160], the emphasis shifted to usage data, mainly due to the availability of that type of background knowledge, easily recorded by the music streaming service.

5.1.2 Target Characteristics. The target characteristics of a playlist are the organization principles which make a playlist of a sequence of songs to be listened to together. The target characteristics should be input in some machine readable form, so that they can be readily used by algorithms. In their 2014 survey on APG-I, Bonnin and Jannach [39] identify several categories of target characteristics. In this section, we review those categories and make a small update to better accommodate the more recent work.[14] We organize APG-I algorithms based on the target characteristics in Table 2. We highlight differences between the research surveyed by Bonnin and Jannach [39] and the research we exclusively survey by dividing algorithms published up to 2013 from those published from 2014 onward.

[14]Specifically, we update the category "free-form keywords" proposed in the survey by Bonnin and Jannach [39] to the more general "free-form text."

Table 2. Organization of APG-I Algorithms, Based on the Category (and Sub-Category) of target Characteristics They Receive as Input, and Dividing Algorithms Published up to 2013 from Those Published from 2014 Onwards

Category	Sub-category	Up to 2013	From 2014
Explicit preferences and constraints	Seed songs	[3, 18, 21, 38, 43, 46, 61, 67, 98, 108, 111, 123, 130, 144, 194, 207, 226, 228, 233, 237, 246, 264, 265, 274, 275, 283, 285, 292, 302, 303, 309, 315, 349, 360, 367, 370]	[4, 12, 32, 36, 59, 64, 71, 87, 102, 104, 115, 117, 124, 126, 129, 161, 162, 164–166, 170–172, 185, 187, 189, 199, 220, 222, 229, 245, 273, 286, 298, 304, 312, 326, 330, 331, 333, 335, 337, 339, 342, 345, 346, 351, 361, 369, 372]
	Free-form text	[75, 246, 293]	[12, 73, 102, 104, 187, 189, 191, 229, 245, 298, 324, 335, 345, 351, 361, 363, 369, 372]
	Explicit and pre-defined constraints	[6, 15, 55, 74, 107, 145, 153, 155, 174, 233, 243, 263, 275–277, 293]	[1, 40, 136, 328]
	Real-time feedback	[56, 66, 123, 158, 264, 265, 293]	[40, 157, 181, 192, 220, 222, 262]
Past user preferences		[56, 78, 108, 130, 158, 251, 252, 370]	[32, 40, 49, 156, 157, 170, 171, 185, 331]
Contextual and sensor information		[3, 26, 35, 79, 100, 135, 158, 244, 258, 259, 293, 294]	[116, 125, 126, 159, 232, 304, 324]

Those published up to 2013 are also reviewed by Bonnin and Jannach in their 2014 survey [39]. The categories and sub-categories are the same ones used by Bonnin and Jannach, except for a minor change we make so as to better accommodate recent work: we rename their category "free-form keywords" to "free-form text."

Explicit Preferences and Constraints. Some algorithms allow users to input the target characteristics manually, in different ways:

Seed songs. Users can guide the algorithms in their song selection by specifying the first song [32], or the first and last song [5, 8], or a list of songs already contained in the playlist [331], or the set of all the songs to include in the playlist[15] [194]. Some algorithms allow users to specify seed artists, instead of seed songs [32, 304].

Free-form text. In this case, users specify constraints on the songs by providing free-form text, which is used to select relevant songs [73]. The free-form text can be single keywords, such as artist names, musical genres, or moods [362, 363]. But free-form text can also be well-formed natural language phrases. For example, the APG-I algorithm proposed in [324] generates a playlist from a natural language text.

Explicit pre-defined constraints. Similar to free-form text, users can specify constraints on the playlist. However, in this case, the user does not have the flexibility of free-form text. Instead, constraints are pre-defined and users choose among them, e.g. the user chooses a desired mood from six categories [40], or the user chooses a tag from a tag-cloud [233], or colors from a color-picker [1].

Real-time feedback. Users provide feedback on the playlist as it is being played and generated. Feedback might be explicit, by liking or disliking a song [264], or implicit by listening to a song to completion or by skipping a song [265]. The playlist can be modified in real time according to the feedback [192]. As well as giving feedback on songs, users can give feedback about metadata associated with the songs. For example, in [181, 262] the user is shown the tags of the current songs and can select one or more of those tags in order to influence the selection of the next song.

Past User Preferences. The users' musical preferences are an important target characteristic. In fact, although users may input some explicit target characteristics, such as a seed song, they implicitly desire that the constructed playlist contains music that they like [11]. For example, Lee et al. find that a user's opinion about a whole playlist can be easily influenced by a single song that the user loves or hates, or even by a specific element of the song that the user loves or hates [205]. This means that music in a playlist should be highly personalized. The musical preferences of a user are usually estimated by considering usage data, such as listening logs and manually constructed playlists [38, 292] (see Section 5.1.1).

Contextual and Sensor Information. The listening context influences the musical choices of users [2]. For example, the user's mood and location can influence their musical choices [65, 90]. Therefore, context-awareness is an important target characteristic. Mood and location are only two examples of listening context, which is a broad concept. Indeed, Kaminskas and Ricci define the listening context as "any contextual conditions that might influence the user's perception of music" [186]. Other examples of listening contexts are user activities, e.g., "party" [68]; the time of day [149]; the weather conditions; characteristics of the user's listening device such as the battery level; ambient conditions such as light and noise levels; and motion, e.g., as measured by an accelerometer [304].

Acquiring the listening context of a user is a first, necessary step towards context-awareness. Some listening contexts may be easier to acquire than others. For example, the level of light can be easily acquired with sensors that feature in nearly any device. Other contexts may be more difficult to acquire, especially those contexts which are not observable by means of a sensor, such as the mood of the listener. In the literature, we find examples such as: building a model to infer a user's

[15] In this last case, algorithms are tasked with arranging the provided set of songs, without applying any song selection. These special APG-I algorithms are sometimes called sequencing algorithms [36].

mood from audio signals detected by microphones [260]; inferring the user's mood from free-form text [324]; and inferring what activity the user is engaging in from listening logs [159].

Once the listening context is determined, a playlist can be constructed by means of handcrafted rules that link the context to the music. For example, we find several papers that extract a model of user's pace from accelerometer data and construct a playlist where the BPMs of songs depend on the user's pace [35, 100, 244, 258, 259].

Discussion. The categories of target characteristics that we present above are complementary. For example, while a seed song broadly defines how a playlist should sound, past user preferences and contextual information can tailor the playlist to the tastes of the user and to their current context. Hence, some algorithms combine different target characteristics to construct high-quality playlists [171].

Nevertheless, the most common way of specifying the target characteristics is via seed songs, as Table 2 shows. The other ways of specifying the target characteristics are generally under-explored, especially in the literature from 2014 until now. One notable way of specifying the seed songs is by providing a list of songs already contained in the playlist. In this latter case, the algorithm adds more songs to the playlist, so as to fit the same target characteristics as the original playlist [364], which is a task known as APC. APC has benefits both for listening to and for creating playlists: APC enables users to enjoy listening sessions that continue beyond the end of a finite-length playlist, while also making it easier to create longer, more compelling playlists without the need to have extensive musical familiarity [306].

APC was the focus of the ACM RecSys Challenge 2018,[16] in which participants were asked to add more songs to user-created playlists taken from the Spotify music streaming service [364]. In total, 113 teams participated in the Challenge, which represents a landmark in APC research. APC is the dominant research trend in APG-I: we estimate that over 40% of the works in APG-I from 2014 onward focus on APC.

5.1.3 Algorithm Type. We review algorithms for APG-I based on their type. In their 2014 survey on APG-I, Bonnin and Jannach [39] identify several types of algorithms. In this section, we review those types, while adding two types that emerge from the more recent work: sequence modeling and RL. We organize APG-I algorithms based on their type in Table 3. We highlight differences between the research surveyed by Bonnin and Jannach [39] and the more recent research by dividing algorithms published up to 2013 from those published from 2014 onward.

Similarity. Similarity algorithms use song similarity to construct playlists. Song similarity can be derived from different kinds of background knowledge, such as content-based data [14, 21, 46, 161, 265, 285], metadata [274, 283], tags [284, 304], manually constructed playlists [38, 233, 292], listening logs [346], ratings [56, 315], or any combination of the above [174]. For example, both Pohle et al. and Cai et al. compute the similarity between songs based on an MFCC representation [21, 46, 285]; Pauws and Eggen and Polignano et al. count the values of metadata features that two songs have in common [274, 286]; and Bonnin and Jannach consider how often two songs co-occur in manually constructed playlists [38]. More recently, we have seen increasing reliance on song embedding representations [85, 199, 346], learned using the word2vec algorithm [240]. These embeddings are obtained by treating songs as words and treating manually constructed playlists as phrases, by analogy with natural language. Song embedding representations can also be given as input to a clustering algorithm, such as k-means, to generate playlists of similar songs by sampling from the clusters [116].

[16] https://recsys.acm.org/recsys18/challenge/

Table 3. Organization of APG-I Algorithms Based on Their Type, and Dividing Algorithms Published up to 2013 from Those Published from 2014 Onwards

Algorithm type	Up to 2013	From 2014
Similarity	[21, 38, 43, 46, 56, 67, 74, 75, 98, 107, 108, 111, 123, 130, 153, 174, 194, 207, 226–228, 233, 251, 252, 264, 265, 274, 283, 285, 292, 302, 303, 309, 315, 349, 360]	[1, 12, 36, 40, 49, 64, 115, 117, 129, 161, 164, 165, 232, 273, 286, 304, 346]
Collaborative filtering	[3]	[4, 12, 59, 102, 104, 104, 166, 170–172, 185, 187, 189, 191, 229, 245, 298, 326, 331, 335, 337, 339, 351, 361–363, 369, 372]
Frequent pattern mining	[38, 61, 144] [156]	[39]
Statistical models	[61, 237, 238, 246, 367, 370]	[32, 73, 87, 199, 304, 330, 333, 345]
Case-based reasoning	[18]	[124–127]
Discrete optimization	[5, 6, 15, 98, 145, 155, 194, 243, 263, 275–277, 284, 285, 309]	[156]
Sequence modeling	—	[33, 34, 71, 162, 172, 191, 311, 312, 340, 342]
Reinforcement learning	[66, 158]	[192, 220–222, 299–301, 328]

Those published up to 2013 are also reviewed by Bonnin and Jannach in their 2014 survey [39], except for two papers on reinforcement learning, that were published before 2014 but did not make it into their survey. The algorithm types are similar to those used by Bonnin and Jannach, except for some changes we make so as to better accommodate the recent work that we exclusively survey. Specifically, we include two additional algorithm types: sequence modeling and reinforcement learning

Integrating multiple sources of background knowledge is beneficial when computing similarity. For example, in the context of judging the similarity between a song that could be added to a playlist and the songs already in the playlist, research shows that users consider content-based features, such as energy and tempo, as well as meta-data, such as musical styles and lyrical content [22, 323].

The perception of song similarity is subjective. Even expert listeners are found to disagree when asked to rate the similarity between songs [109]. For example, some people consider content-based data more than metadata while judging similarity, and other people may do the opposite [205]. Some work in APG-I integrates personalization in the similarity computation. For example, Sotiropoulos et al. allow users to set different weights for different features when assessing similarity, e.g., a user might weight content-based data more than metadata or vice-versa [316]. And, in [302], Sandvold et al. propose a system where users can assign tags to songs, drawing from a vocabulary of tags. Then, the system learns how to tag new songs, so that the predicted tags reflect the user's tagging

style. Finally, playlists can be created using a similarity measure based on both kinds of tags (those that are assigned and those that are predicted), which means that similarity is personalized based on the user's tagging style.

Once similarities are computed, songs can be chosen for their similarity with the seed songs [14, 38, 226], or for their similarity with other songs liked by the user [158]. Another possibility is to create playlists so as to maximize the similarity between songs [194].

Playlists generated with similarity-based algorithms are expected to be coherent. However, coherence is not the only quality criterion for playlists, as some other criteria exist, such as diversity [290]. Also, a risk with optimizing for similarity is that the playlist may become monotonous [205], e.g., containing songs from the same album. See Section 6.1.2 for a discussion of coherence and diversity.

One use-case for similarity algorithms is that of playlist sequencing, the special case of APG-I where the target characteristics are given as a set of songs, and algorithms are tasked simply with arranging the set of songs, without applying any further song selection, in such a way that the music is coherent, from one song to the next song [36]. For example, Bittner et al. and Cliff both use a similarity algorithm working on content-based similarity [36, 74]. Their approach compares the distance between songs based on several features and then arranges the songs in such a way that those distances are minimized. Sarroff and Casey [303] use the approach of building a machine learning predictor working on content-based features that can distinguish suitable from not-suitable song-to-song transitions. Finally, Furini and his co-authors go in the direction of personalized sequencing [115, 117]. They analyze song-to-song transitions in a user's playlists so that they can sequence playlists in a personalized way.

CF. CF is a common approach in the RSs literature. It is based on the heuristic that if the active user agreed with certain users in the past, then these users are similar to the active user, and items that these users liked should be relevant to, and can be recommended to, the active user [296]. Hence, the use of CF algorithms is facilitated by the existence of usage data, recording the preferences of other users. Specifically, they typically assume a sparse user–item matrix that may record user ratings for items or user interactions with items. CF is then a family of methods for predicting the rating a user would assign to an item or the relevance of an item to a user, based on the data that is given in the rest of the matrix. It is the fact that this data may come from other users that makes these approaches "collaborative."

The most common way of employing CF for APG-I is by applying the playlists-as-users analogy [38, 298], in which a user is a playlist and an item is a song: instead of a user–item matrix that records each interaction between a user and an item, we have a playlist–song matrix that records information about the presence of each song in a playlist. Another common analogy is the titles-as-users analogy [298, 351, 369], in which a user is a playlist title and items are again the songs. Which analogy to employ depends on the target characteristics. The playlists-as-users analogy fits the case in which the target characteristics are given as seed songs. The titles-as-users analogy fits the case in which the target characteristics are given as a playlist title, i.e. a special case of free-form text. For simplicity of exposition, most of the rest of this section uses only the playlist-as-user analogy.

In the playlist-as-user analogy, the playlists–songs matrix can be unary [172], i.e., recording a 1 if a playlist contains the song. Or, it can be non-unary; e.g., it may assign a value to a song according to its position in the playlist, giving more weight to the later songs [12, 331]. Given a playlists–songs matrix, CF algorithms that would ordinarily predict the relevance of an item to a user can be re-purposed to predict the suitability of a song for a playlist.

One option is to employ nearest neighbors CF algorithms [255][17]. For example, the system in [341] computes the similarity between the active playlist and the other playlists in the dataset as the cosine similarity of their row-vector representations in the playlists–songs matrix. Then, for each song in the catalogue, it computes a score by summing the similarities of the active playlist to the k most similar playlists in the dataset that contains the song, where k is a positive integer hyper-parameter. Finally, the highest-scoring song is selected to be added to the playlist. The algorithm above corresponds to the user-based k-nearest neighbors CF algorithm in the RSs literature [198]. The user-based k-nearest neighbors algorithm is also employed in [166, 172].

Another option is to use the item-based k-nearest neighbors algorithm [91]. For example, the system in [340, 342] computes the similarity between songs as the cosine similarity of their column-vector representations in the playlists–songs matrix. Then, for each song in the catalogue, it computes a score by summing the similarity of the last song in the playlist to the k most similar songs in the catalogue.

Some work uses similarity functions other than cosine; e.g., Jaccard [331]. Additionally, the similarity functions can be augmented with heuristics, e.g., by giving higher weight to unpopular items [189].

But there exist CF algorithms, other than nearest neighbors, for constructing playlists. For example, Aizenberg et al. [3] use a **matrix factorization (MF)**-based approach, in which the playlists–songs matrix is factorized into two low-dimensional matrices, containing the playlist embeddings of every playlist in the dataset and the song embeddings of every song in the catalogue. Then, for each song in the catalogue, the system computes a score by taking the dot-product of the playlist and song embeddings. Finally, the highest-scoring song is selected to be added to the playlist.

In some cases, MF is the first step of a two-step APG-I algorithm, especially in the case of APC, e.g. [298, 351]. MF is used to learn a model that can predict the relevance of every song to a playlist. But these relevances are used to filter to a more manageable (but still large) set of candidate songs to add to a playlist. These remaining songs are associated with features such as their popularity [351] or the degrees of homogeneity and diversity they would bring to the playlist [298]. A second model learns to re-rank the remaining candidates based on these features. Re-ranking algorithms commonly used in APG-I include gradient boosted trees [351], for example, XGBoost [62]

The years since the publication of the Bonnin and Jannach survey have also seen the rise of deep learning. Deep learning is a form of machine learning that is based on the use of many-layered artificial neural networks. It has led to advances in different application fields of AI, such as natural language modeling [42] and object classification in images [200]. Given those promising results, deep learning has recently been applied to the task of APG-I. We will discuss its use in sequence modeling in a later subsection, but here we can see the effect it has had on CF-style approaches to APG-I.

One example is to be found in the work of Zhao et al. [369]. They take a similar approach to Aizenberg et al. above, i.e., using an MF algorithm to factorize the playlists–songs matrix. But then, where Aizenberg et al. compute scores between playlist and song embeddings, Zhao et al. provide for extra learning: the playlist and song embeddings are fed into a feed-forward neural network, which outputs a score indicating the fit of the song for the playlist.

A well-known family of deep learning models are the autoencoders. In the simplest case, an autoencoder consists of two components, encoder and decoder, both of which are usually feed-forward neural networks. A more sophisticated autoencoder is the adversarial autoencoder, in

[17]Nearest neighbors CF algorithms can be considered similarity algorithms, but we review them in this section and not in the "Similarity" section as they are commonly categorized as CF, especially in the RSs community [296].

which the hidden vector distribution is regularized so as to match a Gaussian prior distribution [234]. Autoencoders of both kinds are used for APG-I by setting the input to be a binary vector indicating which songs are in the playlist [335, 361]. This, in effect, is the playlist-as-user analogy and so we can regard these as CF approaches to APG-I. The output is a vector approximating the input vector, that can be used for selecting other songs for addition to the playlist. Vagliano et al. [335] successfully integrate additional background knowledge into an adversarial autoencoder for APG-I, by concatenating embeddings of textual data, such as the playlist title, to the hidden representation.

The playlists-as-users analogy has three main limitations: (1) the constructed playlists are not personalized, (2) the performance depends on the number of songs in the playlist, and (3) they may perform poorly on songs that occur in very few playlists. Concerning limitation (1), the songs are chosen so that they are tailored to those already in the playlist, but not to the listener's musical tastes. Some work tweaks CF approaches so that they become personalized. One approach, for example, is to modify the active playlist by adding songs from other playlists created by the same listener [3, 170, 185]. Concerning limitation (2), CF algorithms are affected by the cold-start problem, which manifests with small or newly created playlists [59]. In fact, the accuracy of CF algorithms is positively correlated with the number of seed songs, i.e., the algorithm will generate a better playlist when provided with more seed songs [364]. And, CF algorithms are not able to generate a playlist if no seed songs are provided. In such cases, it is necessary to resort to a fall-back strategy, for example by working with other target characteristics or by employing simple heuristics based on song popularity. The solution to limitation (3) is usually some form of hybrid. For example, Val et al. combine song embedding representations extracted from content-based data, metadata, and manually curated playlists by means of a deep feed-forward neural network, so as to model the probability that a specific song is a good fit for a specific playlist [337, 339].

The titles-as-users analogy shares the same three limitations. Some authors propose a solution to alleviate the cold-start problem when using the titles-as-users analogy, which consists of clustering similar titles together, so as to increase the number of songs for each title [369]. One way to cluster titles is to rely on simple text pre-processing pipelines, which transform the text to a common format, for example by removing special characters [362, 363, 369]. Another way to cluster titles (although here employed in a recurrent neural network) is by employing text-embedding procedures, such as FastText [175], and by running a clustering algorithm on those embeddings [245]. Yang et al. [361], on the other hand, treat playlist titles as sequences of characters and use a CNN to process the characters, obtaining an embedding vector that can be used to predict the songs in the playlist, given the title. They combine the CNN with an autoencoder to obtain a system that has both a titles-as-users approach and a playlist-as-users approach.

One last drawback of CF approaches, however, is that they are not designed for the specific challenges of APG-I, and aspects such as song coherence have to be addressed separately [39].

Frequent Pattern Mining. Frequent pattern mining approaches work by mining patterns from a dataset of manually constructed playlists. A pattern can be expressed in the form $S_1 \Rightarrow S_2$, where S_1 and S_2 are two sets of songs. The pattern signifies that it is likely to find the songs in S_2 after the songs in S_1. In sequential pattern mining, S_1 and S_2 would be sequences of songs, not sets of songs.

Patterns can be used to generate playlists. For example, consider three songs: s_1, s_2 and s_3; given a playlist with s_1 and s_2 as seed songs, if we have extracted the pattern $S_1 \Rightarrow S_2$, where S_1 is the set $\{s_1, s_2\}$, and S_2 is the set $\{s_3\}$, then a candidate continuation for the playlist is s_3. Pattern mining is not often applied to APG-I. However, Bonnin and Jannach show that patterns and sequential patterns can, in fact, achieve comparable performance to other types of algorithms that were in use in 2014 [38, 39]. Chen et al. furnish one example of the use of sequential patterns [61]. They

propose to use a simple bigram model that extracts $S_1 \Rightarrow S_2$ rules by counting how frequently the set of songs S_2 follows the song S_1 in the dataset, corrected with Witten-Bell discounting [177].

One problem with this approach is that patterns based on songs will be very rare, and sequential patterns even rarer. For example, even in a very large dataset of playlists, the number of times that Alice Coltrane's *Wisdom Eye* follows *Post Requisite* by Flying Lotus will be small. The patterns will be associated with very low confidence values, and many reasonable patterns will be not be seen at all. The solution is to mine patterns based, not on songs, but on representations of songs, e.g., based on hand-crafted features or latent features. In [144], for example, the PrefixSpan algorithm [142] is used to mine sequential patterns on song latent embeddings. The song latent embeddings are obtained by applying Latent Dirichlet Allocation to the songs' tags [37].

Statistical Models. Statistical models work by modeling the probability of adding a song to the playlist.[18]

One class of statistical models are the Markov models, i.e., those that model the probability of adding a song to the playlist based on the current "state." In the APG-I research cited below, the state is defined as the last song in the playlist. Markov models are proposed in [61, 87, 237, 246, 333, 345, 367].

The core component of Markov models is the estimation of the song-to-song transition probabilities. McFee and Lanckriet [238] offer a comparison of a number of Markov models, which differ on how the probabilities are estimated: some of them count song co-occurrences in manually constructed playlists, while others rely on content-based or metadata similarity, and in some cases latent representations. This recalls the problem with frequent pattern mining (above): song co-occurrences will typically be low frequency; using song representations or Hidden Markov Models (e.g., [367]) can overcome this.

Markov models may lead to the construction of problematic playlists [342]. For example, adding a song based only on the previous one may lead to a lack of coherence throughout the playlist.

Some other statistical models are not Markov models, and model the probability of adding a song to the playlist based on the other songs in the playlist. For example, Hu and Ogihara [158] consider a playlist as a time series and use an autoregressive integrated moving average model [146] to predict the next song.

The most sophisticated statistical models are also personalized, i.e., they model the probability of adding a song to the playlist based on the other songs in the playlist and based on the user's musical tastes. For example, Ben-Elazar et al. [32] propose a Bayesian classification model whose parameters are estimated via variational inference based on the playlist songs and on the other songs liked by the user. Two similar models are proposed in [330, 370].

Other notable statistical models are proposed for the scenario in which the target characteristics are specified using natural language. For example, Chung et al. propose a statistical model for linking a word to a song [73]. It is trained on a dataset of manually constructed playlists and their titles. In practice, they learn an embedding for every word and song, in such a way that a particular song embedding is aligned with a particular word embedding if that song is likely to appear in a playlist which contains that word in its title.

Case-Based Reasoning (CBR). CBR is an approach to problem-solving that involves reasoning with prior experiences. CBR can be effective when two tenets hold [204]: similar problems have similar solutions; the types of problems an agent encounters tend to recur. Case-based APG-I assumes that

[18]Some of the other types of algorithm are also statistical models. In particular, CF, sequence modeling, and RL are statistical models. Also, similarity algorithms can be used to build statistical models. However, we devote a separate section to each of those as they are notable and recognizable algorithm types for RSs in general and for APG in particular. The algorithm type designated as statistical models, following Bonnin and Jannach [39], are ones that explicitly compute transition probabilities.

existing playlists encode the results of prior reasoning, and that it is therefore worthwhile to re-use existing playlists when creating new playlists.

Given a dataset or case base of existing playlists and an initial seed playlist, Gatzioura and Sànchez-Marrè use CBR to recommend a set of songs for playlist continuation (APC) [124]. Their system retrieves from the case base a set of k playlists that are similar to the user's seed playlist. In this system, similarity is an aggregate of song similarity, where song similarity is based on shared meta-data. The system recommends songs taken from the k playlists, based on the playlist similarity scores.[19] The approach is extended to include time-of-creation pre-filtering [125] and shared latent topic pre-filtering [126, 127].

By contrast, Baccigalupo and Plaza [18] deal with case-based playlist generation from a seed song (APG-I), rather than case-based playlist continuation and treat the order of the songs in the playlists in the case base as significant. In their approach, the system retrieves and combines a set of so-called relevant patterns. Relevant patterns are subsequences that contain the user's seed song and which recur across multiple playlists in the case base.[20] There is an even greater risk of low frequency patterns than there was in frequent pattern mining and Markov model algorithms: Baccigalupo and Plaza's algorithm works with songs, rather than representations of songs, and the patterns it mines must include the seed song.

Both Gatzioura and Sànchez-Marrè and Baccigalupo and Plaza equip their systems with additional scoring mechanisms that try to take coherence and diversity into account, these being concepts that we discuss further in Section 6.1.2.

Discrete Optimization. A different way to approach playlist generation is by setting up a discrete optimization problem. Given the catalogue of songs and a set of explicitly specified constraints that capture the desired target characteristics, the goal is to construct a sequence of songs that satisfies the constraints, while maximizing some utility function [39].

Discrete optimization approaches to APG-I differ in their constraints. Some approaches impose constraints on consecutive songs, for example by requiring that their similarity should be higher than some value, as measured by a song similarity measure [5, 263]. Other approaches impose constraints directly on metadata or content-based data [6, 15, 145, 155, 263, 275]. In the case of [15], for example, this is done by requiring that at least n songs in the playlist should have a specific musical genre. In [156], constraints are sequential patterns that have been mined from a user's listening history.

In addition to constraints that must be satisfied, there may be a utility function to be maximized. For example, the approaches in [145, 194, 284] seek to maximize the similarity of consecutive songs in the playlist, as measured by a song similarity measure [195].

Also, different approaches use different strategies to solve the optimization problem. Some use linear programming [5, 6]; others use constraint satisfaction [15, 263, 275]; yet others use simulated annealing [145, 277]; and at least one uses each of genetic algorithms [155], ant-colony optimization algorithms [243], and tabu search [156]. In all cases, the greatest challenge is coping with the combinatorial explosion that results when scaled up to large music collections.

Sequence Modeling. We have already discussed the recent contribution that deep learning has made to new CF algorithms for APG-I. We have also seen some algorithm types (most notably

[19]This CBR system could alternatively be classified as a similarity algorithm, but we review it in this CBR section and not in the "Similarity" section because of the way it computes not just song similarity but also playlist (case) similarity and because this is how the authors view their work and how it is viewed in [39].

[20]This CBR system could alternatively be classified as a frequent pattern mining algorithm, but we review it in this CBR section and not in the "Frequent Pattern Mining" section because this is how the authors view their work and how it is viewed in [39].

sequential pattern mining and statistical models) that depend on the sequence of songs in a corpus of playlists. In what we are here referring to as sequence modeling algorithms, we look at the application of deep learning methods to the song sequence data.[21]

One well-known family of deep learning models are RNNs [224]. RNNs are particularly suited to learning from sequential data. At their core, there is the concept of hidden state, which is updated at each step of the sequence, as a function of the current and past elements of the sequence. The hidden state contributes to the prediction of the next value in the sequence.

RNNs can be applied to APG-I by considering that a playlist is a sequence of songs. As such, RNNs can naturally predict the next song in the sequence, i.e., the song to add to the playlist. They can be used to construct a playlist from scratch, song by song, but may be particularly suited to APC, where they propose a continuation of an existing playlist. In [172, 340–342] a particular RNN, known as GRU4Rec [150], originally proposed for SARSs, is used for APG-I. They train the RNN on a dataset of manually constructed playlists, with the objective of correctly replicating those playlists, i.e., the RNN is fed the playlist up to song n, and its parameters are optimized so that it correctly predicts the song in position $n + 1$. Related work makes use of other RNN models, such as LSTMs [71, 162]. Shih and Chi [312] propose an additional RNN training step, in which other training objectives are included, such as song diversity and freshness, by resorting to a policy-guided RL algorithm [154]. Moreover, by treating a playlist title as a sequence of characters, it is possible to use RNNs to process the characters, obtaining an embedding vector, that can be used to predict the songs in the playlist, given the title [191].

Another family of deep learning models are the CNNs. The use of CNNs was popularized in computer vision, where they yield state-of-the-art accuracy in tasks such as image recognition [200]. While not always seen as sequence models, CNNs have been adapted to do language modeling [178] and this inspires ways of using CNNs for APG-I. For example, by applying the songs-as-words analogy, it is possible to use a CNN to predict the next song in the playlist [351].

Our final type of deep learning model is the transformer [347]. It is only very recently that transformers have been applied to APG-I, e.g., [33, 34, 311]. In the context of APG-I from seed songs, Bendada and his co-authors [33, 34] report a comparison in a A/B test of two approaches: a transformer and a latent factor model. The transformer model resulted in longer listening times, which is a positive result. However, for more mature users, this was accompanied by a reduction in actions such as adding songs from the playlist to a list of favorites. We mention transformers in Section 8 as a promising direction for future work. Indeed, sequential modeling in general is a promising approach, but it does require the availability of large quantities of reasonably high-quality playlists.

RL. RL is a form of machine learning in which an agent, through interaction with its environment, learns how to take specific actions so as to maximize a long-term numerical reward. In each step, the agent takes an action and the environment transitions from one state to another state. After each action, the agent may observe a reward. The agent aims to learn a policy that defines which action should be taken in each state in order to receive the greatest cumulative reward [325].

RL is suitable for modeling sequential problems, in which each action is taken as a consequence of the previous action. Playlist construction can be modeled as a RL problem, by considering an action to be the addition of a particular song to the playlist for which there is a reward. The goal is to learn a policy that maximizes cumulative reward. The survey of APG-I by Bonnin and Jannach does not contain any RL algorithm for APG-I, since most were published after 2014.

[21] The name sequence modeling could, of course, be used generically to cover sequential pattern mining, Markov models, and some of the other work cited in the previous section. We choose to use it more narrowly to refer to neural approaches that build models from sequential data.

In APG-I work that is based on RL, the current state is given by the list of songs in the playlist. For example, if the playlist has been constructed up to the 10th song, and the agent is tasked with choosing the 11th song, then the state is the list of those 10 songs; an action is the addition of a specific song to the playlist.

The observed rewards depend on the user, and on the newly added song. In some work, rewards are observed implicitly; for example, a skip is a negative reward, while listening to a song to completion is a positive reward [192]. In other work, rewards are observed explicitly, for example by asking the user to rate the newly added song on a numerical scale [221]. In some works, rewards are calculated from the system's background knowledge. For example, in [299, 300], reward is based on similarity in an embedding of acoustic features, and this is extended in [301] to similarity also in a knowledge graph embedding combined with measures of popularity and novelty. In this way, rewards help balance smooth transitions, diversity, and discovery.

In some APG-I work based on RL, the agent learns a policy from the observed rewards directly. The work in [66, 157, 192], for example, uses Q-learning [357] to learn a policy from observed rewards. In other work, the agent estimates a reward function from the observed rewards and uses this reward function to determine the policy. The work in [221, 222], for example, parameterizes the estimated reward function as a linear transformation of the newly added song's content-based data.

One apparent issue with the formalization of APG-I as RL above is that of combinatorial explosion. For example, if the catalogue size is just 10 million songs (much smaller than it is in some music streaming services [318]), then there are 10^{70} possible states just for playlists that contain 10 songs, and 10^7 possible actions. One way to tackle the state explosion problem is by factorizing the song representation in terms of their features, such as content-based features [192, 221, 222], metadata [157] or mood [66], and/or by applying windowing, e.g. by representing just the last three songs of a playlist in a state [66, 157].

At the time of writing, RL algorithms for APG-I are promising but relatively under-explored. Their need for reward data is their greatest limitation, and it is not clear that calculating rewards from other data is an adequate substitute for human reward data.

Having now reviewed the different algorithm types, we finish this section with some topics that are algorithmic but which cut across the different algorithm types.

Discussion: Trends. Our survey of algorithm types enables us to identify some trends in the research. We gave an overview of these in Section 3, where we were contrasting our survey with previous ones. But, now, at the risk of some repetition, we can use Table 3 and our presentation of the algorithms above to confirm them. There are differences between the research up to 2013 (the *first period*), which was already surveyed by Bonnin and Jannach [39], and the research from 2014 onward (the *second period*), that we exclusively survey. We can explain the changes in terms of at least three factors: (1) the paradigm shift in music consumption from small personal music collections to streaming services, requiring algorithms that scale well to millions of candidate songs; (2) consequentially, the availability of certain kinds of data, most notably usage data, such as the MPD, released in 2018 for the ACM RecSys Challenge, which enables approaches that train on collections of manually constructed playlists; and (3) the rise of deep learning across ML in general.

Accordingly, work on Similarity algorithms has declined a little since the *first period*, perhaps because some approaches do not scale well. The more recent Similarity algorithms use embeddings learned from datasets of manually constructed playlists. Use of CF algorithms has grown enormously, exploiting, for example, the MPD, mentioned above. A little of the CF work uses nearest-neighbors methods, but these do not always scale well. Instead, MF has become common, and the most recent work combines MF with models for re-ranking or with multi-layered neural networks,

autoencoders, and the like. Discrete optimization methods have largely not survived the transition to music streaming services. When there are millions of songs, it is near impossible to utilize discrete optimization algorithms, as their worst-case computational complexity is exponential.

Approaches that try to learn from the order in which songs appear in playlists have changed greatly. Neural approaches to sequence modeling, such as recurrent neural networks and transformers have superseded frequent pattern mining and, to some extent, statistical models. Finally, RL, which seems to promise much in terms of modeling but also the handling of user feedback, has grown strongly.

Discussion: Hybrids. The types of algorithms we reviewed differ in their performance. There is, for example, some evidence that CF and sequence modeling excel in generating high-quality playlists, especially in the case where the target characteristics are specified as a list of seed songs [39, 364]. However, it is wrong to consider one type of algorithm to be superior to another. A more correct view is to consider them as complementary. Algorithms of different types usually leverage different sources of background knowledge. For example, while CF algorithms are mostly limited to usage data, similarity algorithms can easily include content-based data. Also, algorithms of different types usually accommodate different ways of specifying the target characteristics. For example, while CF algorithms are mostly limited to the case in which a static playlist is generated from a list of seed songs, RL algorithms can generate dynamic playlists that adapt to the user feedback in real time.

It is therefore necessary to employ hybrids that combine algorithms of different types or that use different data, in different circumstances, especially depending on the background knowledge available, and on the way the target characteristics are specified. For example, Schedl et al. [304] use a similarity-based algorithm to generate a playlist, which is then adapted in real time based on the contextual information, gathered from sensors and processed by a statistical model. Frequently, different types of algorithms are combined with the goal of attaining playlists of higher quality [172, 248, 326, 364]. One common way of combining algorithms is to compute a weighted average of their predictions [229, 351].

Deep learning is often used as a powerful tool to combine heterogeneous features and information sources [320]. For example, Vall and his co-authors [337, 339] combine song embedding representations extracted from content-based data; metadata; and manually curated playlists, by means of a deep feed-forward neural network, so as to model the probability that a specific song is a good fit for a specific playlist.

We refer the reader to [44] for an understanding of the possible ways in which RS algorithms can be combined, many of which can be adapted to APG-I.

Discussion: Re-Ranking. Finally, we will discuss re-ranking, which can be thought of a particular type of hybrid algorithm. Systems that use re-ranking typically have a two-stage architecture. In the first stage, a model ranks the candidate songs for their relevance to a playlist. In the second stage, the candidates are re-ranked by a second algorithm, typically using data that was not used in the first stage.

In APG-I, there are at least two motivations for using re-ranking. One motivation is to improve scalability. The first stage would use a model that can score all the songs in the catalogue for relevance but at speed. A common choice is an MF model. Only those candidates with the highest scores from the MF model are passed to the second stage, where they are ranked by a model that takes different data into account and may not operate as quickly as the model in the first stage [298, 351].

The second motivation for re-ranking is to improve the top-n song recommendations that are selected for display to the user of, e.g., an APC system. Songs appearing lower in the ranking

produced by the first stage might be 'promoted' in order to produce a set of n recommendations that satisfy additional criteria. In [187], for example, the intuition is that the set of songs that is recommended for continuation of a playlist should match the level of diversity of songs that are already in the playlist. Kaya and Bridge use sub-profile aware diversification [188] to implement this intuition, measuring an increase in accuracy. A similar approach is taken in [229], where diversity is measured by means of content-based features. The intuition of [116] is, instead, that some songs recommended for continuation of a playlist should be familiar to the user, while some others should be novel. All these papers implement their intuitions using a re-raking approach.

5.1.4 Evaluating APG-I Research. Up to now, we have referred to playlist quality as the way we would measure the performance of APG-I algorithms. Playlist quality is, however, an ill-defined concept, difficult to pin down to a mathematical definition that would allow its measurement. In fact, playlist quality depends on the musical tastes of the user, on the user's familiarity with the music [356], and on the listening context [2]. For example, two different listeners may rate the quality of the same playlist differently, because they may have different musical preferences, because they may already know the songs, or because they are listening to the playlist in two different locations, e.g., at the beach or in the bus.

In their 2014 survey on APG-I, Bonnin and Jannach [39] review the different strategies for measuring the quality of playlists. They identify three categories of evaluation protocols:

(1) User studies, where users are involved in rating the quality of playlists;
(2) Objective measures, where statistics of the constructed playlists are computed, under the assumption that those statistics (e.g., coherence or diversity of the songs' musical genres) reflect the notion of playlist quality; and
(3) Ground truth playlists, where algorithms are tested for how well they can recreate manually constructed playlists or listening logs, under the assumption that the manually constructed playlists or listening logs reflect a gold standard.

These three categories are still valid today, covering also the evaluation protocols in papers published from 2014 onward. In the following, we focus on how APG-I algorithms are evaluated in the papers that are exclusive to our survey, i.e., papers published from 2014 onward.

Evaluation protocol (3) is probably the most common and can be described as a three step procedure: (a) preparation, in which a number N of songs are withheld from a ground truth playlist; (b) recommendation, in which an APG-I algorithm is used to get a ranked list of K candidate songs to be added to the playlist; (c) scoring, in which metric M is used to measure the fitness of the K recommended songs relative to the N withheld songs. K can assume any value from 1 to the size of the song catalogue. N can assume any value from 1 to the playlist size. The three steps are repeated for every ground truth playlist in the dataset, and the resulting values of M are averaged.

Different instances of evaluation protocol (3) differ for the choice of N, K and M. For example, in a comparative evaluation of APG-I algorithms, Bonnin and Jannach set N to 1 and allowed K to range from 1 to 1,000 [38, 39]. They used hit-rate as M, which, for a ground truth playlist, measures whether the set of K recommended songs contains the withheld song. They reported the percentage of ground truth playlists for which there was a hit. By contrast, in the ACM RecSys Challenge 2018, N is different for each ground truth playlist,[22] K is set to 500, and M is set to a number of different metrics related to hit-rate, including Normalized Discounted Cumulative Gain and a metric they called R-precision [364].

[22]Specifically, N was the length of the playlist less the number of songs that were not withheld. For some playlists, all songs were withheld; for others, five were not withheld; and there were other playlists where the numbers not withheld were 10, 25, and 100.

In general, there is no agreement on what combination of N, K and M to use, but there is work that offers insights into some combinations to adopt or avoid. For example, Bonnin and Jannach show that using average log-likelihood as M, which was used in some other work [238], leads to inconsistent conclusions with hit-rate-related metrics, and recommend to avoid its use [38, 39]. Kamehkhosh and Jannach carry out a user trial where users are asked to choose the most appropriate continuation for a playlist among four song alternatives, three of which are generated using an algorithm and the last is the withheld song [183]. They find that users are likely to select the withheld song as a favorite continuation. Their experiment provides evidence that the choice of N as 1, K as 1, and M as hit-rate is a reliable setting. In contrast, Vall et al. criticize the choice of setting a specific cut-off for K, by showing that the relative ordering in performance of a set of competing algorithms changes when varying K from 1 to the size of the song catalogue, while keeping N fixed to 1 and M to be hit-rate [341].

Evaluation protocol (3) is explicitly designed to work in the case where the target characteristics are specified using seed songs, which is the most common scenario in the recent literature, see Table 2. However, evaluation protocol (3) can be adapted to work when the target characteristics are specified in different ways. For example, Chung et al. propose an APG-I algorithm for which the target characteristics are input as free-form text, and they evaluate the algorithm using ground truth playlists, setting N to the playlist size, K to vary from 5 to 20, and hit-rate as M [73].

Evaluation protocol (3) works under the assumption that the ground truth playlists represent a gold standard, and thus the ability to replicate those playlists reflects the ability to construct high-quality playlists. However, the assumption may be too strong in some circumstances. For example, Hagen finds that manually constructed playlists often do not have clearly defined target characteristics [140]. For example, they may sometimes be used as a randomly arranged container of the user's favorite music, created for convenience of access. Similarly, although listening logs can be assimilated to the concept of playlist, they may contain spurious interactions, such as songs recommended by the automatic continuation features of streaming services during periods that the user is not paying attention to the recommendations. In some work, listening logs are filtered before running the evaluation, for example removing skipped songs [41]. Ideally, the quality of the ground-truth playlists must be checked before running the evaluation, which circles back to the original question of how to evaluate playlist quality. One guideline to distinguish suitable datasets of manually constructed playlists for evaluation is offered in [81], as they find that playlists manually constructed by users for sharing with other users usually satisfy high -quality standards and have clearly defined target characteristics.

Lastly, evaluation protocol (3) is undermined to a degree by several biases, most notably by popularity bias [31]. Since most of the ground truth playlists tend to contain popular songs [51], an APG-I strategy that constructs playlists in a popularity-driven fashion will usually yield good performance [38, 39]. However, while a playlist with popular songs would satisfy a large share of users, it would not suit minorities of listeners. There exist several strategies for de-biasing evaluation protocol (3) with respect to popularity, for example see [60]. A simple strategy is used in [341], where evaluation protocol (3) is run separately for popular songs and for the rest of the songs, finding that the relative ranking in performance of algorithms changes for these two segments of the song catalogue.

Evaluation protocol (2) is sometimes adopted for evaluating APG-I algorithms. It works by computing statistics on the constructed playlists, under the assumption that those statistics reflect aspects of playlist quality. For example, several papers measure song diversity and coherence from song tags or musical genres [73, 170, 171, 185]. Additionally, there are papers that measure song popularity [73], and others that measure song novelty, i.e., the degree to which songs are known by the listener, and freshness, i.e., the degree to which the songs are recently released [312]. These

statistics do offer insights into the characteristics of constructed playlists, but it is not clear how those characteristics relate to the concept of playlist quality. For example, it is not clear what value of song diversity in a playlist is ideal: research suggests that songs should be somewhat diverse, while staying coherent overall [86, 182]. See Section 6.1.2 for a discussion of coherence and diversity.

Evaluation protocol (1) consists of user studies. For example, RL algorithms require real-time interaction with the listener and are evaluated with user studies in which the quality of an algorithm is estimated by monitoring implicit user signals, such as the number of skips [192], or by explicitly asking the user if they like the music or not [221, 222]. As another example, Ikeda et al. employ a user study to evaluate the smoothness of song-to-song transitions in playlists [161]. However, user studies have the disadvantages that they may not be reproducible and they are costly and time consuming. Another strategy for involving users in the evaluation of playlists is via A/B tests, in which users of streaming services are partitioned, and each partition receives playlists constructed by a different algorithm. Users in each partition are monitored for their engagement with the playlists, for example by monitoring the play counts [40], which is an indicator of playlist quality. Unfortunately, A/B tests require resources not accessible to most researchers, such as the availability of a music streaming service platform in which the A/B test can be conducted. A middle ground between an A/B test and evaluation protocol (3) is counterfactual evaluation, which allows estimation of the performance of a candidate algorithm, as if it were in production, by relying on listening logs extracted from whatever algorithm is currently in production. Researchers at Spotify share their recipe for counterfactual evaluation in [138], showing that they can rely on high correlation with actual A/B tests.

This concludes our survey of APG for individual users. We turn now to the case of APG for groups of users.

5.2 APG for Groups of Users

The APG work that we reviewed above tacitly assumes that the playlist is constructed to serve a single user, suitable for private listening sessions. However, listening to music is often a collective activity, consisting of users enjoying music together. Collective listening allows the discovery of new music, gives insight into the music tastes of peers, and creates shared moments where the listening can bring people closer together [141, 225]. Instances of collective listening can happen during a shared car journey, at a party, or at the gym, for example. For these occasions, it is important to tailor the playlist for the group, so as to satisfy the musical preferences of every user. There exists a category of APG algorithms which are explicitly designed to generate playlists for groups, i.e., the APG-G algorithms.

The work in APG-G that we survey is closely related to the work on APG-I that we surveyed in the previous section, not least because the two research topics share the end goal of generating a playlist automatically. Definition 2.2 states that playlist generation is the problem of selecting a sequence of songs from a catalogue of songs, while using some background knowledge, in order to match some given target characteristics. The fundamental difference between APG-I and APG-G resides in the target characteristics. APG-I algorithms handle target characteristics coming from a single user, while APG-G algorithms handle target characteristics coming from multiple users, i.e., the musical preferences of the group members, which, as we will discuss below, are then aggregated to construct the playlist. The aggregation step is not present in APG-I algorithms, as the target characteristics are coming from a single user. Also, some APG-G algorithms support a way of giving target characteristics that is peculiar to the group setting: song requests (see Section 5.2.1). With a song request, any of the group members can suggest what song to play next in the playlist. This is similar to a seed song in APG-I algorithms. However, differently from seed songs, which

specify the music to start the playlist, a song request specifies a song to include at any point in the playlist, interactively.

APG-G algorithms are strongly related to group RS, which are tasked with recommending items to a group of users. In their survey of group RS, Masthoff and Delić [235] describe the typical design of a group RS as a three step procedure, that we borrow and adapt for our discussion of APG-G algorithms: (1) preference acquisition, in which the musical preferences of all group members are acquired; (2) preference aggregation, in which the musical preferences of all group members are aggregated into group preferences; and (3) playlist construction, based on the group preferences. In the rest of the section, we describe the APG-G algorithms in terms of their preference acquisition, preference aggregation, and playlist construction mechanisms. Also, we describe strategies for evaluating APG-G algorithms.

5.2.1 Preference Acquisition. APG-G algorithms differ in what kind of data are acquired as musical preferences. We can distinguish implicit from explicit data. Implicit data are obtained by monitoring users' behavior, without any explicit actions required by the users. Examples of implicit data include listening counts, which can be acquired by inspecting log data [78, 219]. Explicit data, instead, are filled in by the user manually. Examples of explicit data are ratings, such as a one to five rating [19], like/dislike [54], or a vote [257] for a musical item, as well as song requests [99, 319, 348], and pairwise ratings, indicating the preference of one song over another song [16].

Explicit and implicit data are limited in that they are restricted to the users' actions. For example, it is not possible to know the preference of a user toward a musical item if they never interacted with it. Some APG-G work uses explicit and/or implicit data to infer unknown preferences. For example, the Poolcasting system [19] infers unknown song preferences by taking the average of the known preferences for songs composed by the same artist. And, the Flytrap system [78] uses a taxonomy of musical genres to infer unknown song preferences based on the known preferences for songs of similar genres. Similarly to Flytrap, the PartyVote [319] system uses song similarity based on content-based data for preference inference. Finally, the Flycasting system [147] uses CF to infer the unknown song preferences based on the known preferences for songs of a community of users.

The musical preferences of group members can be acquired statically, if the acquisition happens only once, or dynamically, if, instead, the acquisition continues over time. For example, in the MusicFX system [236], the preferences are acquired statically during a registration process, where users are asked to rate musical genres in a range from minus two to two. As another example, in the Poolcasting system [19] the musical preferences are acquired dynamically based on the feedback that users give to the current song, which will influence the choice of the next song. We note that in the case of static acquisition the three steps of preference acquisition, preference aggregation, and playlist construction happen sequentially, while in the case of dynamic acquisition the three steps happen iteratively, because the next song in the playlist is selected by aggregating the group feedback for the previous song, e.g. see the Adaptive-radio system [54]. One advantage of the static acquisition technique is that the algorithm is ready to work for new users, while the dynamic acquisition technique requires the users to interact with the system. However, dynamic preferences are advantageous because they improve over time with usage. Some work combines the two techniques. For example, the Poolcasting system [19] selects the first song in the playlist by aggregating static group preferences acquired from listening logs and then selects the subsequent songs by also considering user feedback.

Some of the preference acquisition mechanisms above acquire "private" musical preferences, i.e., the musical preferences of users for their private music listening [19], and use those private

Table 4. Classification of the APG-G Work That we Survey Based on How the Musical Preferences of Group Members are Acquired, on How They are Aggregated, and on How the Playlist is Constructed on the Basis of Those Group Preferences

Reference	Preference acquisition					Preferences aggregation		Playlist construction	
	Mechanics		Type		Inference	Strategy	Requests	Strategy	Similarity correction
	Stat.	Dyn.	Impl.	Expl.					
[19]	✓	✓	✓	✓	✓	Avg wo misery		Deterministic	✓
[78]	✓		✓		✓	Avg		Stochastic	✓
[236]	✓			✓		Avg wo misery		Stochastic	
[147]		✓		✓		Avg	✓	Deterministic	✓
[16]		✓		✓		Avg		Deterministic	
[54]		✓		✓		Avg wo misery		Stochastic	
[99]		✓		✓		Avg	✓	Stochastic	
[319]		✓		✓	✓	Avg	✓	Deterministic	
[219]		✓	✓		✓	Avg		Deterministic	

We distinguish static (Stat.) from dynamic (Dyn.) preference acquisition; we distinguish the acquisition of implicit (Impl.) from explicit (Expl.) data as preferences; and we also indicate whether preference inference is done or not. Additionally, we distinguish two strategies for aggregating individual preferences, Average (Avg) and Average without misery (Avg wo misery), and we indicate whether song requests are handled or not. Finally, we distinguish the strategies to construct the playlists, and we indicate whether similarity correction is applied or not.

preferences in a group setting. This makes the assumption that a person's private tastes are also their communal tastes. However, the private and communal tastes of an individual may differ [84].

Table 4 classifies the APG-G work that we survey for how the preference are acquired, distinguishing implicit from explicit preferences, static from dynamic acquisition mechanics, and indicating whether preference inference is done or not.

5.2.2 Preference Aggregation. Once the musical preferences of all group members are determined, the next step is to aggregate those preferences into the group musical preferences. The APG-G papers that we survey perform preference aggregation in two different ways:

Average. The group preference for a musical item is the arithmetic average of the musical preferences of the group members towards that item. For example, the Flytrap system [78] averages the preferences of group members for songs.

Average without misery. This is similar to Average, but items for which at least one user has expressed an extremely low rating are avoided. For example, the MusicFX system [236] avoids musical genres that have been given extremely low ratings by at least one user in the group. And, the Adaptive-radio [54] system implements an interesting variant of Average without misery where group members are allowed to determine songs to avoid, while the remaining songs are treated as having equal group preference.

It is worth mentioning that the above preference aggregation mechanisms, Average and Average without misery, are two of the many mechanisms used in group RSs research. For example, Masthoff and Delić list 11 preference aggregation mechanisms to be found in group RSs research [235].

Along with the mentioned preference aggregation mechanisms, some of the APG-G work that we survey handles song requests separately. Song requests are a type of musical preference that consists of suggestions of songs to be played next in the playlist. One way to handle song requests separately from the other user preferences is via a queue, i.e., song suggestions are played in the

order that they arrive, until the last song in the queue is played. If no songs are in the queue, the next song to play is selected via the group musical preferences, aggregated as described above. When faced with multiple requests coming from the same user, systems may decide to play a subset of those requests, while still considering the other requests as explicit preferences [319].

Table 4 classifies the APG-G work that we survey for how the preference are aggregated and indicating whether song requests are handled or not.

5.2.3 Playlist Construction. Once the group preferences are aggregated, the last step is to construct the playlist. The APG-G papers that we survey follow two main strategies for selecting music for the group playlist: deterministic and stochastic. The deterministic strategy consists in selecting the music with highest group preference. For example, the CoCoA-radio system [16] applies the deterministic strategy, by selecting the next song in the playlist as the one with highest group preference. The stochastic strategy consists in producing a probability distribution based on the group preferences, e.g., songs with highest group preference have highest probability of being played. The stochastic strategy allows for discovery of new music, at the risk of selecting music that the group does not like [51]. For example, the MusicFX system selects a genre by drawing from a probability distribution over genres, determined by the group preferences for genres.

Together with group preferences, other considerations can be made for constructing the playlist. One common strategy is that of similarity correction, that is considering similarity to the previously played song, together with group preferences, for selecting the next song. For example, in the Flytrap system [78], the probability distribution over the candidates for next song is corrected with a multiplicative term, which takes into account the similarity with the previously played song.

Table 4 classifies the APG-G work that we survey on whether the music selection strategy is deterministic or stochastic, and indicating whether similarity correction is applied.

5.2.4 Evaluating APG-G Research. Let us turn our attention now to evaluation of APG-G algorithms. Recall from Section 5.1.4 that three protocols are used to evaluate research in the case of APG-I for individual users: (1) User studies; (2) Objective measures; and (3) Ground truth playlists.

Overwhelmingly, protocol (1) — conducting a user study — is the main approach to evaluation of APG-G research. One strategy for evaluating APG-G algorithms is to present a playlist to a group of listeners and to measure the playlist quality by monitoring the engagement of the group with the playlist, for example by means of a survey. For example, Chao et al. install their Adaptive-radio system in an office environment so as to provide co-workers with music during working hours; they report the results of a survey, where the co-workers are asked for their opinions about the musical choices [54]. Similarly, McCarthy and Anagnost install the MusicFX system in a gym and poll participants on whether the music playing in the gym has improved since the installation of MusicFX [236]. A similar experiment is reported in [147], and in [289], where the authors compare several preference aggregation and playlist construction strategies.

We do have an example of protocol (3), where listening logs provide a ground-truth [219]. Specifically, Li et al. take check-in data for a coffee-shop from Foursquare,[23] as well as the listening histories of the people in the shop from Last.fm;[24] they analyze the accuracy of algorithms in predicting songs in the listening histories of users currently in the coffee-shop.

We have no clear examples of protocol (2) for APG-G. Indeed, a conspicuous share of APG-G work that we survey (four of nine papers) does not offer any evaluation.

[23] https://foursquare.com/
[24] https://last.fm

6 MPG

MPG is a major topic within research on playlists. Research in MPG looks into how people construct playlists manually. Research in MPG is important as understanding how people construct playlists manually can help to improve research in APG, as well as improving user interfaces for playlist construction.

We define the task of playlist generation in Definition 2.2 as the problem of selecting a sequence of songs from a catalogue of songs, while using some background knowledge, in order to match some given target characteristics of the playlist. In MPG, users set their minds to some desired target characteristics (or themes, see Section 6.1.1) and manually select songs from a catalogue so as to match those desired target characteristics. One notable playlist construction style would start with the user selecting a handful of anchor songs to reflect the theme [81] and, on the basis of those anchor songs and on the theme, the user would then search through the catalogue for similar songs to add to the playlist [27, 323]. The search for songs might be based on familiarity, drawing from the user's musical knowledge, or by looking for other songs from the same artists, or of the same genre, or by exploring the catalogue for songs coming from similar artists or having similar genres [96].

6.1 MPG for Individual Users

In this section, we survey the literature on MPG for individual users, presenting common themes for playlists in Section 6.1.1, and criteria for selecting the songs, with a special focus on the issue of song ordering, in Section 6.1.2. Finally, in Section 6.1.3 we review a special case of MPG: assisted MPG, i.e. the case in which the user is assisted by an algorithm while constructing the playlist manually.

6.1.1 Themes. The target characteristics of a playlist are the organization principles which make the playlist of a sequence of songs to be listened to together. In MPG, the target characteristics are expressed by users for themselves, because they are the agents that carry out the playlist construction. This is different from APG, where the target characteristics need to be expressed in a machine readable form, which inevitably limits expressiveness to the kinds of things that algorithms can interpret. We distinguish the target characteristics as expressed in MPG from those expressed in APG, by using the word "themes" to refer to the target characteristics for MPG.

Hagen [140] analyzes the themes of playlists created by users of music streaming services and provides a categorization of those themes, employing four categories:

Standardized. Themes concerning standard music organization principles, including: genres, e.g., "Hawaiian music"; artists, e.g., "100% Prince"; albums, e.g., putting songs from one or more albums in a playlist; styles, e.g., "British & psychedelic"; instrumentation, e.g., "songs with cello"; and performance, e.g., "female singers"; years, e.g. "1970s and older." Other themes concern more specific music organization principles, e.g., producer, label or composer, or content features, e.g., BPMs or energy.

Contextual. Themes concerning the listening context. The concept of listening context is broad, one definition (repeated from earlier) being "any contextual condition that might influence the user's perception of music" [186]. Examples of contextual themes include: events and activities, e.g., "birthday party," "road trip," "Christmas"; mood, e.g., "happy hits"; and time of day, e.g., "night time."

Personal. Themes concerning the user's personal life, including: relationships: e.g., "breakup"; biographical histories, e.g., "the soundtrack of my life" or "Amsterdam 1999"; and memories and experiences, e.g., "memory lane."

Individual. Idiosyncratic themes. This last category is a container for a wide range of peculiar themes that arise from an individual's creativity. Examples include: messages and puzzles, e.g., a playlist that sends a hidden message by playing with song titles, artist names or lyrics, such as a playlist where all songs are about water; themes outside the music universe, such as soundtracks, e.g. "Gossip Girl," and cultural references, e.g., "Viva Las Vegas." Another notable type of example is a playlist containing favorite songs, e.g., "all time favorites" or "latest favorites."

The categorization above is consistent with other categorizations proposed in related work, e.g. see [81, 82, 92, 143].

It is worth noting that a playlist theme may not give a clear indication of the music included in the playlist. For example, personal themes strongly depend on a user's personal life, and their musical choices may make sense to the playlist constructor only. In some cases, even the playlist theme itself may make sense to the playlist constructor only, for example the individual theme about water that we gave earlier. Other categories of themes, such as standardized and contextual themes, may be more "predictable" in the sense that a third entity may be able to select a suitable song to add to those playlists.

Moreover, even though every playlist is thematic by definition, the theme may be more or less strictly followed by the playlist constructor. For example, Cunningham et al. report that playlists created for personal use have a less strictly defined theme, as they may be employed as a background for another activity, while playlists created for sharing with other users usually have a more clearly defined theme [81]. And Kamehkhosh et al. suggest that playlists sometimes have more than one theme [182].

A few studies investigate the popularity of different themes, i.e., how frequently users create playlists of specific themes. For example, Cunningham et al. analyze playlist requests posted by users in blog posts and find that the majority of requests are for standardized and contextual themes [81]. And Pichl et al. carry out a data-driven analysis of manually constructed playlists in Spotify, by clustering playlists on the basis of acoustic features and analyzing those clusters based on their semantics, as extracted from the most relevant song tags in the cluster [281, 282]. They identify five clusters. They call the biggest cluster "feel-good music" and find that 91% of playlist creators have at least one playlist in this cluster. The other clusters contain more niche music, as only a minority of playlist creators have a playlist belonging to these latter clusters.

6.1.2 Song Selection. In MPG, the songs are selected manually by the user so as to match the playlist theme, and according to different guidelines. One obvious guideline is musical taste since users commonly compile playlists of music that they like [86]. Interestingly, Hansen and Golbeck find that songs liked the most by users are usually picked first [143]. Two other guidelines are song coherence/diversity and song ordering.

Song Coherence/Diversity. Songs should remain coherent throughout the playlist. In the literature, the concept of song coherence is related to that of song similarity. For example, both Kamehkosh et al. and Lee et al. mention that songs are coherent if they are similar in terms of, e.g., tempo, mood, genre, time period, musical style, and/or lyrical content [182, 205]. Music similarity is a multi-faceted and subjective concept that we discussed in Section 5.1.3. The work we reviewed there measures coherence using metadata; for example, musical genres [168, 314] and tags [70, 290].

As well as being coherent, songs should also be diverse through the playlist [143, 327]. We use the concept of song diversity as a contrast to the concept of song coherence. For example, Lee et al. point out that songs should be varied in their metadata, e.g., there should be relatively few songs by the same artists, and there should also be some variety in musical genres [205].

Striking the right coherence/diversity balance is key to successful song selection in playlists [139, 182], almost as important as matching the listener's musical taste [86]. Some work tries to measure coherence/diversity in manually constructed playlists, with the goal of characterizing the ideal tradeoff between the two quantities. For example, Jannach et al. utilize an intuitive measure, counting the average number of songs from the same genre in manually constructed playlists, and find that typical values are around three to four songs per genre [168]. Porcaro et al. [290] measure coherence/diversity as a function of the year in which a playlist was created, by comparing playlists created in the 2000s, in the 2010s and in the 2020s. They find that earlier playlists featured a higher level of diversity, while in recent years playlists tend to be more coherent. They relate this change to the advent of music streaming services, and the phenomenon of filter bubbles [7]. Slaney et al. [314] measure coherence/diversity as a function of playlist length, finding that the longer the playlist, the higher the diversity, which can be related to the fact that a longer listening time requires more diversity so as to keep the interest of the listener alive. Choi et al. [70] measure coherence/diversity as a function of playlist theme, as identified by relevant tags associated with a playlist, finding that playlists with different themes are different in terms of diversity/coherence. These studies highlight the fact that the appropriate tradeoff between diversity and coherence is difficult to determine since it depends on several factors, including the year of creation, the number of songs, and on the theme. Additionally, Lee et al. find that the appropriate tradeoff between diversity and coherence depends on the preferences of the individual user [205]. Of course, this issue has also been explored in work on APG, including work we cited earlier on RL [299, 301] and on re-ranking [116, 187, 229].

Song Ordering. We refer to song ordering as the process of arranging the playlist's songs in a particular order. The relevance of song ordering in the process of manually constructing playlists is disputed. Some work maintains that song ordering is relevant; other work maintains the opposite. All of the work agrees that song ordering is *less important* than other factors, such as striking the right coherence/diversity balance [86].

There is work that investigates the relevance of song ordering via user studies, finding that song ordering is relevant. For example, Cunningham et al. observe the playlist creation behavior of users, reporting that they actively arrange songs so as to achieve a music flow, especially when they are creating a playlist to be shared with other listeners [81]. And, De Mooij and Verhaegh explicitly ask participants in their study about the importance of song ordering, and report a result of zero, on a scale from −4 to 4 [86], which we interpret as an indication that song ordering has some relevance. They also ask participants about the importance of other factors, such as the start and end songs, rated as −2, choosing songs according to taste, rated as 2, and balancing coherence and diversity, also rated as 2. In their playlist creation study, Kamehkhosh et al. ask study participants about the importance of song ordering and six other factors [182]. While song ordering was ranked fourth, suggesting low relevance, they noticed that about one third of their participants reordered the songs at least once during playlist creation, which implies that it is important for at least some people.

However, other work that also employs user studies finds that song ordering is not relevant. For example, Andric and Haus ask study participants to send in a playlist of songs. Later, participants are presented with the playlist they sent, along with an alternative playlist with the same songs but shuffled, and are asked to rate the two playlists based on their quality by sight, i.e., without listening to the music [10]. Andric and Haus find no difference in quality between the two options: half of the participants mention that they picked songs at random from songs that they like when building the playlist they sent in. Similarly, Tintarev et al. find that users do not expect playlists to be ordered in the first place [327].

Some work in APG also indirectly studies the relevance of song ordering. For example, Antenucci et al. weight the final part of a playlist more than the initial part when predicting the next song, observing an increase in performance compared with the case when both parts have equal weight [12]. This is an indication that song ordering is relevant. But, Vall et al. instead find that song ordering is not relevant [341]. They train RNN models for APG, which naturally take song order into account. They train on a set of playlists and, separately, on shuffled versions of the playlists, reporting the same accuracy in both settings, suggesting that ordering is not significant. A result similar to [341] is presented in [237].

Some work investigates the importance of song ordering via statistical analysis. For example, Schweiger et al. consider a set of manually constructed playlists and extract a variety of song features, including content-based data and metadata [310]. The content-based data are acousticness, danceability, energy, instrumentalness, liveness, loudness, speechiness, tempo, valence, key, and mode. The metadata is genre, artist, and popularity. They compute song similarity using this data, measuring higher similarity between songs that are close to each other in the playlists; the similarity decreases when considering songs that are further apart in the playlists, which is an indication that ordering is relevant.

We believe that the contrasting results presented above are due to the lack of a standardized experimental procedure. For example, Schweiger et al. adopt a statistical analysis which proves the relevance of ordering, as songs closer to each others in playlists are found to be more similar to those further away [310]. The difference in similarity is only slight, but can still be detected with statistical analysis. The slight relevance of ordering may not be detectable in user studies. The studies above also differ in their experimental settings, and song ordering may be relevant in some experimental settings, and not relevant in some others. One experimental setting in which song ordering is relevant is when users construct playlists to be shared with other people, as opposed to when they construct playlists for private use, which is one experimental setting in which song ordering is not so relevant [81]. Looking at the studies above, those that deal with playlists to be shared find that song ordering is important, e.g. [12, 86, 310], while those that deal with playlists for private use find that song ordering is not important, e.g. [10, 182, 327]. More generally, song ordering might not be relevant in cases when a playlist is constructed to be listened to in shuffle mode, such as playlists for private use [81]. The research of Leong et al. [215–217] looks at shuffle listening in playlists, highlighting the positive listening experiences allowed by shuffling, like serendipity, as well as studying the different use cases when shuffle listening is appropriate, which may be a good starting point for researching other use cases in which song ordering is relevant.

Song ordering has some connection with song coherence/diversity, as different orderings of the same set of songs may result in different perceived coherence/diversity. Dias et al. [96] present two empirical rules of thumb for song ordering in order to help to strike the right coherence/diversity balance. Rule (1): avoid putting songs by the same artist or with the same genre together in a sequence, unless there is a special link between the two songs, for example, two parts of the same continuous recording [80, 81]; Rule (2): place songs with complementary sounds and styles together consecutively, so as to avoid the "clash of one song against the other."

6.1.3 A-MPG. A-MPG is a special case of MPG, concerned with the development and evaluation of algorithms for assisting users in the process of manual playlist construction. An example of work in A-MPG is [182]. In [182], Kamehkosh et al. utilize algorithms for recommending the next song that the user may want to add to the playlist, in order to relieve users from the burden of searching for songs manually.

A-MPG represents a middle ground between MPG and APG. In MPG, the user is tasked with constructing the playlist, finding songs by manually browsing the song catalogue. In APG, an

Table 5. Organization of Work on A-MPG Based on How the User Is Assisted During the Playlist Construction Process

Category	Up to 2016	From 2017
Visualization	[48, 63, 77, 94, 132, 133, 152, 176, 196, 203, 213, 223, 242, 247, 254, 266, 329, 343, 344, 352]	—
Recommendation	[28, 29, 95, 131, 151]	[182, 184, 241, 288]

We distinguish work that assists users with visualizations and with recommendations. Additionally, we divide work published up to 2016 from work published from 2016 onward.

algorithm is tasked with constructing the playlist, aligned with some target characteristics input by the user. Both MPG and APG have advantages and disadvantages. An advantage of MPG is that users are left in control of the playlist construction process, which allows for self-expression through their own musical choices [358]. But, the process of MPG is time consuming, especially in music streaming services, where catalogues contain millions of songs. An advantage of APG is that it relieves the user of the task of manual song selection. But, APG lowers the control of users over the construction process.

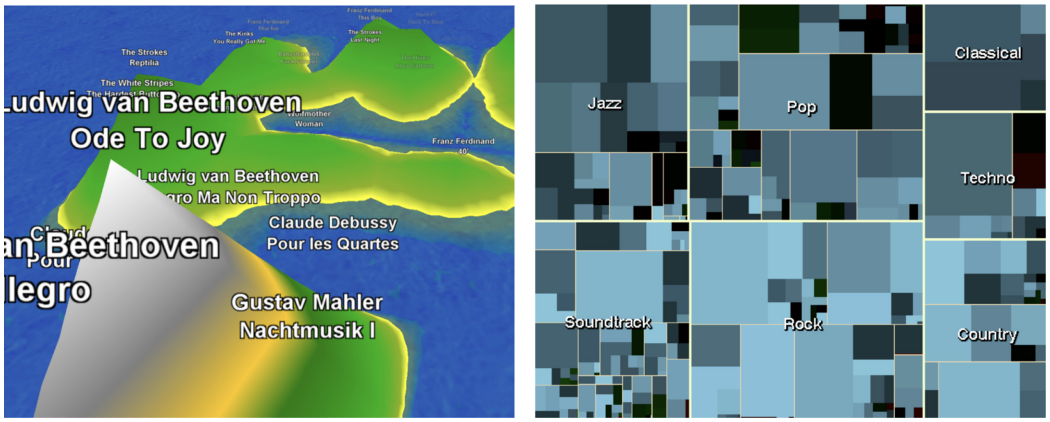
In A-MPG, the user is tasked with constructing the playlist, but algorithms are employed as facilitators, to assist users in selecting the songs.[25] A-MPG unites the two paradigms, leaving users in control of the playlist construction process, while alleviating users of the burden of manually searching enormous song catalogues. A-MPG thus strives to achieve a tradeoff between the time spent in creating the playlist and the user satisfaction with the playlist: MPG is very time consuming, but leads to highest user satisfaction with song choices, since users have full control over them; APG is faster, but may lead to low user satisfaction with the song choices [193]; and A-MPG is less time consuming than MPG, but more time consuming than APG, and can result in user satisfaction with the song choices that is lower than MPG but higher than APG [28].

In their 2017 survey, Dias et al. [96] review the research on A-MPG to that date. They present several categories of algorithms, all based on visualizations: maps, graphs, dots, and radar A-MPG algorithms. In our survey here we include the A-MPG algorithms that were covered by their survey, as well as more recent work. However, the recent work does not belong to the categories proposed in their survey. Hence, we propose a novel categorization of A-MPG algorithms, to cover both recent and non-recent work. Specifically, we divide algorithms in two categories: visualization and recommendation. In Table 5 we divide the work on A-MPG based on the categorization, and we distinguish the research up to 2016, which was already surveyed by Dias et al., from the research from 2017 onward, that we exclusively survey.

Visualization. Some of the work in A-MPG uses visualizations for assisting users in the manual construction of playlists. An example of visualizations are maps that represent the songs in the catalogue by similarity, i.e., similar songs are close in the map, and dissimilar songs are further away from each other. Maps can either be **two-dimensional (2D)**, e.g. [176, 266], or **three-dimensional (3D)**, e.g. [196, 203]. In Figure 4, we show examples of maps from two A-MPG algorithms.

Figure 4(a) is a map proposed in [196]. The authors represent the songs in a multi-dimensional space made of multiple types of content-based data. Then, they use **self-organizing maps (SOMs)**

[25] In MPG, the users are also assisted by algorithms, in particular by search engines, while manually browsing the catalogue. However, those search engines are not tailored to song selection for playlists.



(a) (b)

Fig. 4. Examples of visualization used in A-MPG to assist users in manually creating playlists. (a) Reproduced, with permission, from Peter Knees et al. [196]; and (b) Reproduced, with permission, from Marc Torrens et al. [329].

[197] to project the songs to a 2D space. Finally, they build a 3D map by analogy with a landscape, with hills in correspondence to dense clusters of songs, and valleys in correspondence to sparse areas, while the sea represents areas with no songs. SOMs are frequently used in this kind of work, especially for their ability to project songs not seen at training time, and for their scalability [266]. Similar work that uses SOMs includes [176, 213, 242, 247, 254, 344, 352].

Figure 4(b) is a map proposed in [329]. The authors represent songs as the leaves of a tree, whose upper levels represent, from top to bottom: musical genres, sub-genres, and artists. Then, they use treemaps [313] to transform the tree representation to a map representation. It is possible to draw a map representation at the different tree levels, i.e., at the level of genres, sub-genres, or artists. Figure 4(b) shows a map drawn at the level of genres. Another work that uses treemaps is [94].

Visualizations can also consist of graphs. For example, Gouyon et al. create a graph of artists [133], using similarity between artists, as retrieved from Last.fm, which is known to rely on usage data [47]. Artist-to-artist similarity is symbolized as edges, with longer edges joining less similar artists and shorter edges joining more similar artists. A similar work using graphs is [77].

Visualizations can be drawn using simpler strategies, such as associating songs to points in the valence-arousal space [134], without applying other transformations [48], or by associating songs to colors [152], and drawing a color-map representing a collection of songs.

The visualizations discussed so far in this section represent songs by similarity based on content-based data and metadata. However, using other sources of background knowledge is also possible. For example, Goussevskaia et al. rely on usage data [132]. Some visualizations even leave the user in control to decide their preferred background knowledge for computing song similarity. For example, van Gulik and his co-authors propose a map where song similarities can be computed based on different metadata, such as the year of release, the genre, or based on content-based data, and users are left in control of deciding which sources of background knowledge to use [343, 344].

Once the visualization is created, the user can interact with the visualization while creating a playlist. For example, in the case of maps, different types of interactions are possible, including manual song picking in the map [329], as well as drawing paths [176, 223] or shapes [254] in order to create a playlist of the songs included on the path or shape.

Recommendation. Another way in which algorithms can assist users in the process of manually constructing playlists is by providing recommendations for songs to add to the playlist. In principle, any APG algorithm we present in Section 5.1 can be used for providing those recommendations. For example, Kamehkosh et al. use the APG-I feature that is part of the Spotify API and an heuristic algorithm based on popularity [182, 184], while Goto and Goto use a similarity-based algorithm [131].

The recommendations are usually displayed as elements of the user interface. A standard approach is to present recommendations as a list of songs that can be clicked on [182, 184]. A more creative approach is in [131], which presents recommendations as discs that can be dragged and dropped on top of other discs, so as to build a playlist. Another approach is *Rush* [28]. *Rush* asks the user to select a seed song, and it then presents the user with a carousel of five songs that can be added to the playlist; as soon as the user selects one of these songs, another carousel of five songs is presented, with recommendations dependent on the previous song, and so on, until the playlist is fully built. The same authors present *Rush 2* [29], which works in the same way as *Rush*, but now carousels are replaced by wheels that have at their centre the last selected song, and recommended songs are displaced around the wheel based on their relevance to the seed song. The user can customize *Rush 2*'s recommendations by setting filters on BPMs and genre. *AudiRadar* is a system that is similar to *Rush 2* [151]. *MixTape* [288] is also an A-MPG algorithm similar to *Rush*, but it follows a different interaction paradigm: it shows the user just a single song recommendation, which the user can either accept or reject. As soon as the user gives feedback, another song recommendation is displayed, dependent on the user feedback. *MixTape* chooses between exploration and exploitation: it explores the song catalogue if the user skips a song recommendation, and it exploits similarity when the user accepts a song recommendation. *PlaylistCreator* [95] displays recommendations using a standard list strategy, but the interface allows users to customize the song recommendation algorithm. For example, they can specify the playlist theme, or they can filter songs by metadata. Another way to present recommendations is together with explanations. For example, Millecamp et al. provide explanations based on the visualization of content-based features and find that the presence of explanations increases the satisfaction of users with the process of playlist construction [241].

Kamehkhosh et al. [182, 184] conduct a user study where users select a theme among six themes and then compose a playlist around that theme. Users are partitioned in two different treatments: *rec* and *no-rec*. *Rec* users see 10 song recommendations while building the playlist. *No-rec* see no recommendations. Users in both treatments can search for songs to add to the playlist using a search bar. They find that 67% of *rec* users adopted at least one song recommendation. The users who did not adopt any recommendation were either experts in playlist construction or had low enthusiasm for music. Increasing user trust in song suggestions is mentioned as an area of improvement for increasing the adoption of song recommendations. Also, the mere presence of the recommendations is found to influence the songs that *rec* users search for, even in the case that the song recommendations are not selected.

Discussion. Table 5 reveals the differences between the research on A-MPG up to 2017 (the *first period*), which was already surveyed by Dias et al. compared with the research from 2017 onward (the *second period*). Research from the *first period* focuses mainly on visualizing the song catalogue. In the *second period*, the focus is shifting to recommending songs to be added to the playlist, probably due to the rise of music streaming services. In fact, with streaming services, the song catalogue shifted from personal collections to millions of songs [318], which are difficult to visualize on a map or on a graph. Recommendations, on the other hand, can help users by providing a small selection of relevant songs.

In Sections 5.1.4 and 5.2.4, we reviewed the evaluation of APG algorithms. There is not much to add when discussing the evaluation of A-MPG algorithms, so we do so here in a short paragraph, rather than in a separate section. Evaluation of A-MPG algorithms is primarily done using what we referred to previously as protocol (1), i.e., through user studies. In some cases, what gets evaluated is the usability of the A-MPG algorithm, which can be done using standard questionnaires, common in the Human-Computer Interaction field, e.g. see [218], or through semi-structured interviews. Other evaluation work investigates domain-specific aspects, for example satisfaction with playlists that are constructed with the assistance of an A-MPG algorithm [28]. Finally, some works in A-MPG do not report any experimental evaluation, e.g. [48, 132].

6.2 MPG for Groups of Users

Research in MPG-G looks into how a group of users constructs playlists, as well as the purposes for constructing these playlists, and the practices around listening to these playlists. Earlier, we defined MPG as the task of manually selecting songs from a song catalogue so as to match some desired target characteristics of the playlist. MPG-G is a special case of MPG where a group of users is involved in the manual selection of the songs, instead of a single user. For the purposes of this section, we refer to a playlist constructed by a group of users as a **collaborative playlist (CP)**; and we refer to a playlist constructed by a single user as a **personal playlist (PP)**. Finally, we refer to the users involved in the construction of a CP as the collaborators.

CPs can be constructed by groups of varying size. For example, Park and Kaneshiro report that the majority of CPs have a maximum of six collaborators [268]. The same study finds that collaborators are usually groups of friends, or family, while it is much rarer to construct a playlist together with strangers, which corroborates a similar study by Spinelli et al. [317]. CPs can be listened to in a group, which can be composed of the people that contributed to the playlist, or may include others, such as other guests invited to a party [83]. In a large proportion of cases, CPs are also listened to by individuals on their own, for example to accompany everyday activities [268]. CPs are very dynamic as it is common for the collaborators to frequently update a CP [209]. For example, during a party, a playlist could be modified so as to accommodate song requests from the guests [317].

The practice of constructing CPs is widespread nowadays, because most popular streaming services offer a dedicated feature where collaborators can add, delete, and re-order songs in a CP. In their study, Park and Kaneshiro report that more than half of Spotify users in the USA collaborate on CPs [269].

CPs serve a number of important purposes. Park et al. [270] divide the purposes of CPs into three categories: practical, cognitive, and social. Practical purposes include creating a playlist to be listened to in a group during a shared social occasion or speeding up the creation of a playlist via collaboration between users. Cognitive purposes include discovering new music, since collaborators may select songs which are novel to some of the other collaborators. Social purposes include to keep in touch with the other members of the group, as well as developing a social connection with them, by using music as a medium for bonding, or for sharing music between the members of the group. We note that the practical and social purposes of CPs align with the convenience and self-expression purposes of PPs (see Section 2.1). Several other works corroborate the three categories of CP purposes proposed in [270]. For example, Park and Kaneshiro find that users use CPs as a means of discovering new music added by the other collaborators, and that users welcome these social recommendations because of the social links with the collaborators who recommended them [268]. In a similar vein, Lehtiniemi et al. [210, 211] and Spinelli et al. [317] report that social recommendations are more appreciated than algorithmic recommendations. And, Park and various co-authors find that CPs facilitate the reinforcement of social connections [268,

272], because music is often used as a medium for discovering the personality of other individuals [214], and can spark conversations among collaborators [23]. Similarly, Lenz et al. find that CPs can be used to keep in touch, for example using songs that evoke shared memories, acting a little like souvenirs [214].

In the rest of this section, we continue our characterization of CPs. In Sections 6.2.1 and 6.2.2 we talk, respectively, about themes and song selection strategies in CPs. In Section 6.2.3, we talk about the group dynamics that arise in the construction of CPs.

6.2.1 Themes. The target characteristics of a playlist are the organization principles which make the playlist a sequence of songs to be listened to together. In Section 6.1.1, we make use of the word "theme" to refer to the target characteristics of a manually constructed playlist, and we analyzed the themes of PPs. In this section, we make the same use of the word "theme," as we analyze the themes of CPs.

The themes of CPs mostly overlap with the themes of PPs, which we covered in Section 6.1.1. Often, the themes of CPs tend to be related to group activities, for example a playlist for a house party, an intimate dinner with friends [27], or a road trip [17, 84]. As well as being tailored to the group activity, the themes of CPs are sometimes also tailored to the location where the group activity happens, and to the social interaction the music is hoped to encourage [27]. Intuitively, the theme of a CP needs to be understood by the collaborators, otherwise they will not be sure what songs to add to the playlist [268, 269], and whether the playlist matches a desired listening context [225]. The active engagement of the collaborators with a CP determines its success [268], and so the theme of a CP is usually clearly defined and easy to understand by the collaborators, which is different to what we reported for PPs in Section 6.1.1, as themes of PPs are sometimes only loosely defined, related to the playlist creator's personal life, and therefore difficult for other users to understand.

6.2.2 Song Selection. In Section 6.1.2, we analyze song selection in PPs. One obvious guideline for song selection in PPs is that the playlist constructor selects music according to their tastes. In CPs, there are a number of playlist constructors (the collaborators), and so the song selection should be tailored to the tastes of a group of listeners [268]. The group of listeners might be composed of the collaborators, but may include also other people, for example other people invited to a party. The degree of heterogeneity of the musical tastes among the group of listeners plays a role in song selection for CPs. If the listeners have similar tastes, it is easier to select songs that everybody will like, and the various collaborators can feel free to select music that they truly like, knowing it will be appreciated also by the other listeners. If the listeners have dissimilar musical tastes, song selection might deviate from each individual's musical tastes, because the collaborators might settle for something that pleases, at least partially, the musical tastes of all the listeners. This is especially true in situations where the playlist will be listened to in a large group, and especially with strangers. In such cases the collaborators might settle for a "safe" playlist of songs familiar to most people, made, for example, of current hits and classics [317], and with a high degree of diversity, so as to increase the chance that every listener likes at least some of the songs in the playlist. In circumstances of high diversity, it is common to navigate the playlist by skipping songs [83], and/or by sorting songs, for example by musical genre or by artist name [269].

6.2.3 Group Dynamics. The construction of CPs involves a group of collaborators who select the songs to add to the playlist. The involvement of multiple users in the decision making process raises questions about group dynamics, that we address in the rest of this section.

Collaborators' Roles and Comfort in Modifying the Playlist. The collaborators assume different roles in the process of constructing a CP. In particular, the user that initiates the playlist is sometimes

called the host, and the other collaborators are the guests. The analogy is that of a physical setting, for example a party, where a person, the host, invites other people, the guests [83]. Usually, the role of the host is that of specifying the playlist theme, as well as adding an initial set of songs that fit the theme, and inviting guests to contribute to the CP [83].

Since musical taste is linked to the personality of an individual, sharing music involves some revelation of the personality of an individual [214], which can be a delicate matter, especially when the collaborators are not linked by strong friendship relationships, and even more so when they are strangers [84].

When collaborators do add songs to a playlist, they do so consciously, trying to fit the theme of the playlist and the tastes of the other collaborators and any other intended listeners [210, 211]. Bauer and Ferwerda [25] set up an experiment where a user is asked to propose a song for a CP. The user then receives feedback from four other collaborators, which can either be positive (like) or negative (dislike). They find that, in most cases, it is enough to receive negative feedback from at least one collaborator for the user to reconsider the song that they added. At the same time, the user tends to give positive feedback to songs proposed by other collaborators, even if those songs do not really fit the musical tastes of the user.

In a similar vein, Park and Lee measure the comfort of a user in performing different tasks, namely adding, deleting, and re-ordering songs in a CP [271]. They measure comfort in the case that the user is the host, and in the case that the user is a guest. They also distinguish the case in which the user is deleting a song added by them or by another collaborator. They find that users are quite comfortable in performing addition and re-ordering of songs, both in the case that they are hosts and in the case that they are guests, even if the comfort slightly decreases in the case that they are guests. They also find that users are comfortable in deleting a song they added, but very uncomfortable in deleting a song added by some other collaborator. They mention that deleting someone else's song is a socially rude practice. The result corroborates the work in [268], which finds that users often add or re-order songs, but rarely delete songs. This can cause CPs to grow monotonically in size over time.

Privacy and Access Control. One purpose of CPs is to allow social connection between people by sharing music, to keep in touch with friends, and to use music as a medium for bonding. The music streaming services that allow their users to create CPs offer only basic features. For example, CPs in Spotify are like PPs, with the difference that song addition, deletion, and sorting can be done by a group of authorized users. Some works propose going a step forward, by developing a range of social features in the context of CPs. The proposals include the logging of modifications to the playlists, such as song additions, deletions, and re-orderings, as well as logging the author of the modification [209, 225, 269]; a chat functionality, that can be used by collaborators, for example to discuss whether a song addition is suitable for the playlist or not, or to simply socialize [209, 269, 271]; and a listening-together functionality, that would allow collaborators to tune-in to the playlist, so as to listen to the same music at the same time [269].

Some work reports that users are in favor of including these social functionalities in streaming services. For example, Park and Lee report that the chat functionality would increase the comfort of users in adding, deleting, and re-ordering songs [271]. But, other work reports that these social functionalities should be mindful of user privacy. For example, some users are opposed to the idea of logging the changes to a CP, as they are afraid to be judged for their musical tastes [9, 23], which circles back to the low comfort that collaborators may have in adding, deleting, and re-ordering songs in the playlist (see previous section).

Related to the issue of privacy is the issue of access control. The creator of a CP should be able to determine who can see the playlist, and who can modify it [269]. Some more fine-grained levels of

access control are also discussed; for example, Lehtiniemi et al. propose that the playlist creator should have the opportunity to approve any changes to the playlist [210, 211]. In general, the literature sees fine-grained access control as a positive feature to have in CPs.

So far in our survey, we have focused on work that deals with the construction of playlists, both automatic construction and manual construction. In the remainder of the survey, we focus on another topic, the construction of enhanced playlists.

7 Enhanced Playlists

Section 2 defines a playlist as a sequence of songs intended to be listened to together. Some work we survey goes one step forward, by enhancing playlists with additional features. In the following, we discuss four types of enhancements: song mixing, interleaving songs with speech, playlist tagging, and playlist captioning.

7.1 Cross-Fading

Some work in APG strives to create playlists but with the additional objective that there are smooth song-to-song transitions, so that the ending of one song flows smoothly to the beginning of the next song. For example, Sarroff and Casey propose a method to classify "good" song-to-song transitions from "bad" song-to-song transitions [303]. But even in a playlist with "good" song-to-song transitions, there is still a short silence between the end of one song and the start of the next. Research into automatic cross-fading seeks methods for, as seamlessly as possible, superimposing the beginning of one song over the end of its predecessor. Automatic cross-fading is similar to the practice of DJs who often superimpose consecutive songs so as to create a seamless music flow.

Algorithms for song cross-fading mainly rely on digital signal processing techniques [297] and on psychoacoustic studies [128]. Ishizaki et al. [163] propose a system that works in two steps. The first step is to align the beats of the two songs to be mixed. Beats alignment can be achieved by slowing down or speeding up songs, as well as shifting songs in time. Slowing down/speeding up songs may result in a jarring listening experience. The authors determine a formula that states the discomfort of users while listening to a particular song as a function of the slowing-down/speeding-up factor. Then, they propose an algorithm that determines the slowing-down/speeding-up factor for the two songs in such a way that the listening discomfort is minimized. The second step is to cross-fade the two songs. They devise a cross-fading algorithm that takes into account the energy of the beats, adjusting the energy of the beats of the first song to that of the following song when cross-fading.

A similar two-step strategy is followed in [74] and [36]. The latter work proposes a sophisticated algorithm for determining the transition points, i.e. the points that mark where the two songs should be cross-faded. The mechanism works by extracting the position of downbeats, and several features of those downbeats. For example, they classify downbeats that mark division points in the songs, e.g. those downbeats delimiting parts with lyrics and instrumental parts. Moreover, for every downbeat, they extract timbre features, chroma features, loudness, and vocalness. They pair each downbeat from the first song with each downbeat from the second song, and then they score all the pairs using an heuristic. Downbeats with similar values of timbre and chroma features are rewarded; downbeats with high loudness and/or vocalness are penalized; and downbeats happening on division points are rewarded. Finally, they select the transition points as the pair of downbeats with highest score.

The systems described above work particularly well with songs from the same genre, and especially with dance music, whose regular structure makes it easy to determine the position of the beats [137]. Basu [24] focuses on how to mix two songs coming from a variety of genres, e.g., classical music and techno, without relying on the two-step mechanism described above. Given two songs, Basu's system computes their energy signal. Then, it time scales and time shifts those

energy signals and computes their correction, for different combinations of time scales and time shifts in a range. Then, it selects the correlation signal with highest value, and converts the scaled and shifted energy signal back in the time domain, which results in scaled and shifted versions of the songs, which can be played together so as to achieve cross-fading.

The works on song cross-fading that we have reviewed are evaluated by means of user studies, where listeners are asked to listen to song mixes and to complete questionnaires that report on their listening experience, e.g. see [24, 36, 163].

7.2 Interleaving Songs with Speech

In playlists, one song follows another, from the first song to the last song. Playlists therefore do not facilitate easy access to music contextual information while listening.[26] However, studies report that information seeking is one of the most important motivations for listening to music [206], at least in certain contexts. In the era of music streaming, traditional radio remains a popular means of music access. A recent study finds that music listeners consider radio and streaming services as complementary, because radio allows them to satisfy their information seeking needs, while streaming services fail to do so [53]. The work that we review in this section is concerned with generating sequences in which every pair of consecutive songs is interleaved with speech. This resembles the concept of radio programs. We refer to the sequence of songs and speech as a music tour.

We are aware of two lines of work investigating music tours. The first line of work is that of analyzing real radio programs and developing tools and knowledge that can help to replicate those radio programs. For example, Jani et al. [167] construct a dataset of tuples, where each tuple contains one block of music (e.g., a song) and one block of speech, crawled from actual radio shows. Some tuples in the dataset are positive samples, i.e., were aired in sequence in radio shows, and some are negative samples, i.e., were not aired in sequence. They propose an algorithm that, using content-based data extracted from the speech and music, can distinguish positive samples from negative samples, i.e., that can recognize if one musical block and one speech block are suited to be listened in sequence. The algorithm has an accuracy of 75%. Lukacs et al. [230] construct another dataset by crawling actual radio shows that contain music and speech blocks. They annotate the radio shows, by marking the music blocks, the speech blocks, and hybrid blocks. Finally, they present statistics on the annotated dataset, e.g., statistics on how much speech there is in radio programs, compared to how much music is there. They show that the playlist theme and the time of the day influence the statistics. For example, when comparing pop music radio shows and classical music radio shows, they find that, on average, pop music radio contains more speech than classical music radio, but that the statistic is the opposite when restricting to the evenings.

The other line of work is concerned with generating music tours. For example, Behrooz et al. [30] propose an algorithm for generating small pieces of text, called *segues*, for connecting one song to the next one in a playlist. Segues are then converted to speech using a text-to-speech engine. The work of Behrooz et al. is limited in that they select the segue connecting two songs with author-defined segue scores. The work of Gabbolini and Bridge [118] goes one step forward, by defining a scoring function able to determine the interestingness of any segue. The authors validate their measure of interestingness, finding that it positively correlates with human perceptions of segue quality. The same authors follow-up their work by providing several algorithms to

[26] Some work offers music contextual information in textual form that can be accessed while listening to a song, e.g. the "behind the lyrics" feature on Spotify, https://genius.com/Genius-x-spotify-behind-the-lyrics-the-complete-experience-playlist-annotated. However, attending to this textual information requires actions from users, as they need to access and read the information while listening to the music. In this section, we are more interested in contextual information expressed as speech interleaved with the songs in a playlist.

generate music tours, with the objective being to maximize the interestingness of the segues in the tour [119, 120].

7.3 Playlist Tagging

Music streaming services feature billions of playlists created by users, professional editors, and algorithms [88]. In this content overload scenario, it is crucial to characterize playlists, so that music can be effectively organized and accessed [69].

One common approach for characterizing playlists is tagging, which is the task of assigning to a playlist one or more tags.[27] (Another approach is automatic playlist captioning, which we cover in the next subsection.)

Examples of playlist tagging can be found in [105] and [353], which describe datasets of playlists annotated with a variety of different tags, such as musical genres and decades. Similarly, Choi et al. [68] describe a dataset of playlists annotated with listening context tags. Examples of listening context tags are "workout" and "party," which characterize playlists as being suitable to be listened to by users while working out and while having a party.

Existing work on playlist tagging focuses on listening contexts tags. Choi et al. [68] set up a multi-label classification problem, in which playlists are classified for their listening contexts, proposing four classifiers: two MF-based classifiers, that work by counting how many times a song is associated with each listening context, and two CNN-based classifiers, that work with song audio. Gabbolini and Bridge [121] follow-up Choi et al. by proposing four other classifiers, that integrate metadata in the form of a knowledge graph, reporting state of the art accuracy.

7.4 Playlist Captioning

Tagging is limited to the usage of one, or to a set of, single words. However, playlists may sometimes be centered around elaborated themes, see Section 6.1.1, that may not be explicable by using a set of tags. For example, a playlist tagged as "Jamaica" and "UK" may refer to a playlist of UK songs influenced by Jamaican traditional music or to a playlist of top-charted Jamaican songs in the UK. Natural language, instead, allows for precise characterization of playlists at a high semantic level.

Playlist captioning is introduced in [69] as the task of automatically describing a playlist using natural language. In the same paper, Choi et al. [69] propose to use a sequence-to-sequence (seq2seq) model based on an RNN, similar to those common in machine translation [20], adapted to translate a playlist, as represented by song embeddings, to a caption. The song embeddings are extracted from song audio using a CNN. They benchmark their model on a small dataset of captioned playlists, reporting that their model fails to generalize to new playlists. Similarly, Doh et al. [97] use an RNN and a transformer [347] model to translate a playlist, as represented by song embeddings, to a caption. In their case, they rely on learned embeddings, and they use a larger dataset. Similar also is the work of Kim et al. [190]. Gabbolini et al. [122] also use a transformer architecture. In their case, their song embeddings capture musical knowledge from song audio and tags. Additionally, they utilize linguistic knowledge held by the GPT-2 language model [42], reporting state of the art performance in the captioning task.

8 Possible Future Research Directions

Despite more than two decades of research into playlists, much remains unknown or under-explored. The need for more research is heightened by the advent of music streaming services, which wrought

[27] We remind the reader that we use the word "tag" where there is a fixed vocabulary, and we "user tag" where free text is allowed.

a revolution in how people consume music. We conclude this survey with some possible future research directions. We do not attempt to be systematic or comprehensive. Inevitably, there is something subjective about our selection, and readers of this article will no doubt have ideas of their own that they feel should have been included but were not.

To organize the material a little, we have chosen to use four subsections. The first two correspond to APG and MPG. The second two cut across the APG/MPG distinction: playlist presentation and playlist recommendations. However, we should say in advance that this organization is quite porous since research in one category may feed into research in another.

8.1 APG

Obvious topics for future work include more on sequence modeling, especially with transformers; on deep learning in general; on RL; and on the use of knowledge graphs. But below, we select some topics that we think are more unusual and intriguing.

8.1.1 Lyrics as Background Knowledge. Section 5.1.1 reviews the many types of background knowledge that APG algorithms currently use. Notable is that the use of lyrics is under-explored. Lyrics are used in playlist construction in [237] but it is difficult to find other examples.

A first necessary step is to find out whether lyrics are important for constructing playlists or not. Such evidence as exists is somewhat mixed. In [22], for example, Barrington et al. try to discover which factors influence user evaluations of next-song recommendations. Several content-based features, such as energy, style, and so on, are the ones that count the most; the artist name also counts; lyrics are found to be the one that counts the least. In [323], Stumpf and Muscroft do a very similar thing, observing which song features are mentioned by users while constructing playlists, also finding that lyrics were among the least mentioned features. But the participants in the small-scale study in [205] do mention lyrical content and themes or stories, and some participants claim to construct some of their own playlists based on lyrical content. In [237] itself, it is claimed that lyrical features are most important for genres with salient lyrical content, such as folk music, and are less important for genres such as electronic music. This finding adds nuance that is missing from [22, 323].

There has been an explosion of recent work on **large language models (LLMs)**; see, e.g., the survey in [368]. For cases where lyrical or other textual content is important to APG and MPG, it might be possible to exploit the work that is being done on LLMs.

8.1.2 Innovative Specification of Target Characteristics. In most of the work that we reviewed in Section 5.1.2, the user specifies the target characteristics of a playlist via seed songs. Perhaps the most interesting research direction here comes from finding new, innovative ways of eliciting target characteristics.

There is, for example, a body of work from Nokia Research Centre and Tampere University of Technology in the context of A-MPG that could form the basis for future research on APG [153, 207, 210]. For example, in [207], users move their mouse through a space of album covers organized by genre; they hear audio clips when they hover; the final song that they alight on becomes the seed song. Lehtiniemi and Holm [208] do a similar thing but with a handful of "mood pictures" instead of album covers, where mood pictures are carefully chosen images or brief videos that represent moods. In the context of CPs, Lehtiniemi et al. [211] introduce "mood shapes," which are radar charts that summarize a playlist's moods.

Future work could also look further at *explicitly* inferring target characteristics from a set of seed songs. For example, in [74] the user supplies a set of songs, and the system extracts the BPM for each. The user can then specify a beat map, i.e., an outline of BPMs over time. In [74], the system

sequences the songs to follow the beat map, but one can imagine an APG algorithm that instead chooses songs to fit the map.

There are also innovative methods that do not rely on seed songs. Several are based on free-form text. For example, in the Reflektor system [27], users converse about music through a chat client; the system identifies and creates a visualization of keywords from the users' contributions to the conversation. Similarly, Chaganty et al. propose a conversational approach to playlist generation [52].

Dedicated music fans might appreciate innovative ways of providing explicit constraints. The SmarterPlaylists project, for example, provides a graph-based user interface that allows users to define programs that specify how playlists are to be assembled [202]; e.g., a user might write a playlist generating program that draws songs in alternating fashion from the top hits from today with the top hits from the 1980s but filtered to exclude anything appearing in the user's set of banned songs.

In the cases we have discussed above (i.e., seed songs, free-form text, and explicit pre-defined constraints), users specify target characteristics as an initial input to APG. But the user's feedback while the playlist plays also provides target characteristics that can guide subsequent real-time APG, e.g. [264, 265]. This is relatively under-explored, not least because RL APG algorithms, which are best-suited to using and learning from the feedback, are relatively under-explored. In the few examples of RL for APG, the user feedback is limited to, e.g., skips [192] and explicit, numeric ratings [221]. But feedback can come in other forms; in Section 5.1.2, for example, we described systems where users give feedback by clicking on meta-data (e.g., tags) that describe the current song and upcoming songs in the playlist [181, 262]. Instead of using this feedback heuristically as is done in [181, 262], RL could learn from this or other kinds of feedback, resulting in a more data-driven approach.

Of course, creative ways of specifying target characteristics, especially more cognitively demanding forms of feedback elicited during playlist playback, must be usable by users, and welcomed by them [96]. There is a need for future work of a similar kind to that in [180] that studies music listeners' desired levels of control and interaction, and how these relate to the listening context.

Target characteristics may also derive from contextual and sensor information (Section 5.1.2). Considering its relevance, there is a surprising paucity of context-aware approaches to APG in the literature. The context-aware approaches are mainly based on heuristics, and their accuracy is not always evaluated. This is an obvious opportunity for future research.

One example of the kind of research we are envisaging can be found in [295]. Their user study provides some evidence that user activity (i.e., what users are doing while listening to playlists) affects their mood, which in turn influences their musical choices. Therefore, they suggest to rely very much on activities in playlist generation algorithms, which can probably be more easily inferred than mood can. There is work on song recommendation (as opposed to playlist generation) based on daily activities [93] and on song recommendation based on listening contexts extracted from playlist titles [280] that could feed into this line of research, as can work on the prediction of playlist listening contexts [68, 121].

In fact, the previous paragraph hints at a general point which is that, while there is a relative paucity of work on context-awareness for playlists, some of the issues have been investigated in the literature on general music recommendation (e.g., [13]), from which it is possible to take inspiration.

Finally, growing deployment of sensors offers more data to context-aware approaches. A fun example is [173], where the mix of music is influenced by the mixing of drinks, detected by sensors in drinks coasters!

8.2 MPG

8.2.1 Studies on Playlist Characteristics. The literature on MPG has widely investigated the concepts of song coherence/diversity and ordering; see Section 6.1.2. One interesting avenue for future research is to take into account the playlists' listening contexts (Section 7.3), e.g. whether a playlist is for driving, for the beach, and so on. Future research could look, for example, at how playlist diversity varies between playlists that have different listening contexts. There may even be differences within listening context. For example, Mélo looks into playlists built for relaxation, finding that they can cluster users into five different groups based on features of the songs in the playlists of those users [239]. It would be interesting to know, not just how they vary in terms of song features but also how they vary in terms of playlist characteristics, such as diversity.

Moreover, there are studies which report that playlist characteristics other than song coherence/diversity and ordering are important too, such as achieving the right level of song popularity, the right level of song familiarity, and the right level of freshness [96, 168, 169, 231, 290]. These are under-investigated at present.

Of these, we think the most interesting is familiarity. It is a topic which has been explored a little in the MRS literature, where there is a recognition that recommendation lists should include a mix of items that the user is familiar with (e.g., known songs or known artists) and ones that are likely to be novel to the user [305, 355, 356]. There is a challenge in knowing whether an artist or song is unfamiliar to a user: the user may not have interacted with them on a given platform but may, e.g., have been exposed to them elsewhere [305].

Turning more specifically to playlists, the user study in [205] reveals the right mix of familiar and unfamiliar music is key: people usually like learning about new music and would love to have familiar and unfamiliar music in their playlists. The right mix of familiarity and unfamiliarity seems to be personal, and the unfamiliar music should be linked to the familiar music somehow, maybe by being very similar or by provision of explanations. In their study of A-MPG, Kamehkhosh et al. [182] also find that users are generally interested in getting both familiar and unfamiliar recommendations. There is very little other A-MPG literature (or even APG literature) on how to generate playlists with familiar and unfamiliar songs. This is despite the fact that streaming services do seem to include unfamiliar songs in the playlists they generate. For example, Bontempelli et al. [40] present a feature on the music streaming service Deezer that is about constructing a playlist containing familiar and unfamiliar songs, but the paper does not explain how unfamiliar music is chosen. A paper that does report a method for balancing known and unknown music in a playlist is [116]. They use the user's interaction history to determine both whether something is known or not and also the user's preferences for known versus unknown music. We can also mention Hu and his co-authors [157, 158], who use heuristics and RL to also integrate concepts such as freshness and diversity. However, much more work can be done.

In the context of further research into algorithms for A-MPG and APG that take familiarity into account, it may also be useful to see extensions to our own work on segues [118–120] (Section 7.2). These textual connections between songs in a playlist can act as the explanations of unfamiliar songs, linking them to familiar content, as found to be desirable in [205].

8.2.2 Studies of CPs. CPs are very popular among users of streaming services, but research on CPs is limited to user studies that look into: how group of users construct playlists (MPG-G); the purposes for constructing these playlists; and practices around listening to these playlists. We see great potential for statistical analysis of CPs, for example looking at the diversity/coherence of CPs, especially when compared to PPs. So while, for example, Popescu and Pu [289] report that song diversity is important in CPs, quantifying the levels of diversity of existing CPs, for example as a

function of group size, and comparing this to PPs might help to better understand CPs, and might help to provide fresh knowledge for constructing effective algorithms for APG-G.

Indeed, the APG-G work that we reviewed in Section 5.2 is, to the best of our knowledge, all the work on APG-G that exists. It is all framed for a physical setting, such as a gym or in a party, where a playlist has to be constructed for a group of physically co-located listeners. However, with the advent of music streaming, new ways of listening to co-created playlists have emerged, where the listeners can access a co-created playlist via the Internet. Music streaming services allow for deeper and more nuanced preference acquisition than in the physical setting, as there is access to all the users' listening histories. There is an opportunity for research into the automatic construction, or manually-assisted construction, of playlists that will be listened to in this way. Some of the work on on-the-fly radio programming, e.g., the work in [16, 148], might provide a starting point.

8.2.3 Studies of How Users Interact with Playlists. We do not have many studies on user listening habits, especially in the era of music streaming. For example, Leong et al. [217] investigate listening habits but the paper does so in terms of the listening modes of the day (e.g., sequential listening versus shuffle on iPods). More recent is the survey in [179], which comes at the onset of the streaming age. They look in particular at active and passive listening, finding that listening to single songs dominates active listening while listening to playlists is preferred during passive listening. Their later, larger-scale survey also seeks to answer questions about the degree of control their respondents desire during listening [180]. In a similar vein, the even larger study in [114] identifies seven different personas that capture different streaming service user listening behaviors and degrees of control (e.g., active curator, guided listener).

One aspect of curation is playlist editing, and there are only a few studies about this. In the context of MPG, Cunningham et al. [81] report that some playlists are designed to be edited and others are designed to be left as is. For CPs in particular, Park and Kaneshiro [268] found that 78% of respondents in their study listened to their favorite CP unchanged, while 30% continued to update it. Playlists may even transition to archival status and back again [140]. There is room for more research into the contexts of playlist re-use, the factors that motivate change, and the kinds of changes that are made.

8.2.4 Studies of Diverse Types of Music and Users. Most of the work considers playlists created by and for users located in certain parts of the world. Addressing multicultural diversity is a challenge that could be taken on in future MIR research on music playlists.

First, it may be that the criteria we are using when constructing playlists are best suited to certain types of music. It would be useful to see work that assesses how universal these criteria are, across the use of different scales, harmonies, rhythms, and instruments in the composition and performance of the music.

Second, it is well-known that song RSs have gender biases, disproportionately recommending male artists ahead of female artists, e.g. [106]. It would be useful to study whether these and other biases affect algorithms for APG and A-MPG. If they do, then it would be good to see research into a new generation of these algorithms that mitigate this bias, either using the kinds of methods already to be found in the MRSs literature [106] or perhaps novel, playlist-specific methods.

Finally, where there are user studies, there is little consideration of who the users are; the study in [114] is an exception, since it considers gender and personas. Participants are most likely university students based in a relatively small set of countries. Studies that embrace a wide range of participants, and analyze the results accordingly, might prove illuminating. Music streaming services have the kind of reach that can make these kinds of studies possible.

8.2.5 Integration into Playlist Generation Algorithms. One of the main reasons for studying manually constructed playlists is to discover what makes a good playlist, so as to integrate this knowledge into algorithms for APG and A-MPG. At the moment, for the most part, APG and A-MPG algorithms optimize for some notion of similarity or learn to replicate user-created playlists; see Section 5.1.3. There is room in the literature for more work that models the user-related concepts that are uncovered through study of manually constructed playlists and integrates them in the automatic playlist creation process. For example, Hu and his co-authors [157, 158] use heuristics and RL but also integrate concepts such as freshness and diversity. However, much more work can be done.

It follows too that the concepts that are uncovered should feature in the evaluation of algorithmically created playlists. An exemplar of work of this kind is [168, 169]. They split a set of user-curated playlists in half and use the first half for training APG algorithms, and the other half for testing. Their evaluation metrics include accuracy, popularity, coherency, and diversity. Among many results, they show, for example, that the APG algorithms that they use cannot replicate the levels of diversity exhibited in the user-curated playlists. More work like this would be very valuable.

We finish with two topics that are not specific to APG or MPG: playlist presentation and recommendation.

8.3 Presentation of Playlists

There is very little research that looks at how to present playlists (whether automatically generated or manually constructed) to the user, even though the topic is of importance. Of course, a linear presentation is most obvious. But, even then, there is little research to guide decisions such as: the balance between text and imagery, and the amount of metadata to display.

In systems that recommend continuations, there is little research into how to present the recommendations, for example how many competing recommendations, or how far into the "future" recommendations should extend. One exception is [212]. In [212], Lehtiniemi and Seppänen consider how to present recommendations, how many recommendations to present, as well as how to give explanations to accompany recommendations. But they do so in a very small-scale study and for small-screen devices only. Another small-scale study can be found in [262]. They distinguish the case in which upcoming songs are shown and not shown, reporting that users prefer the former. For upcoming songs, Nonaka and Nakamura [256] propose the concept of branching playlists. In this way, the user can see the decision process. Predating the streaming era, Goto and Goto [131] design an innovative interface for displaying multiple playlists and allowing users to extend and merge them.

Some systems allow users to give feedback and even to have control during playback. For example, streaming services may offer user interface elements such as those proposed in [205] that allow users to explicitly ban songs from generated playlists; PlaylistPlayer by Nakano et al. allows users to re-order the songs in playlists dynamically during playback [253]. We are not aware of any user studies to date that consider the design of these kinds of interface elements.

Above, we mentioned that Lehtiniemi and Seppänen [212] briefly look at explanations to accompany recommendations. But this is also under-explored. There is little work on how to generate explanations in the context of APG and A-MPG; how to present them; and when and whether they are valued by users. For example, Millecamp et al. [241] show only content-based explanations; Moscati et al. [248] can explain recommendations in terms of the simple algorithms that make up their hybrid RS. But there is room for other kinds of explanations.

The opportunity here is for explanations that are more oriented toward the properties of the playlist. For example, a system might explain why a song fits the playlist theme, or how it enhances

playlist diversity, or how it is compatible in music flow with the previous song. Playlist-based explanations such as these have not been investigated yet. Of course, such explanations will only have fidelity in APG and A-MPG algorithms that do in fact take these properties into account during playlist construction.

Up to now in this section, we have been assuming that playlists are presented using text, e.g., list of titles, perhaps with accompanying images. But there are alternatives. The techniques we discussed for visualizing music collections (Section 6.1.3) could be adapted to present playlists, as is done in [334], for example; and there are even proposals for interfaces that present playlists in an auditory fashion [322]. Both are interesting avenues for future research. More radical still is the work by Burnett et al. [45], which attempts to bring back some of the properties of mix-tapes. By storing playlist data on NFC tags attached to bracelets, they create playlists that are physical, and that might be gifted and treasured.

8.4 Recommendation of Playlists

Most of the literature that we have surveyed here is about algorithms for constructing playlists (APG and MPG). However, given that streaming services are filled with billions of playlists [88], relatively little is known about recommending existing playlists [305]. One reason may be that the research literature reflects the fact that streaming services seem to favor playlist generation over playlist recommendation. Presumably, only high-quality playlists should be candidates for recommendation. Among the multitude of playlists that a service hosts, quality is not guaranteed: algorithms for APG and A-MPG may not always produce playlists that are worth recommending; and, with MPG, users may be saving collections of songs that are not truly playlists, e.g. sets of favorites [140].[28] Research into automatic identification of high quality candidates might be a necessary part of research into playlist recommendation.

Once candidates are identified, it is possible to recommend playlists by relying on algorithms developed for the well-known task of song recommendation. Commonly used algorithms for song recommendation represent the active user as an embedding, represent the songs as embeddings, and recommend songs whose embeddings are closest to the user embedding [305]. It is possible to recommend playlists instead of songs by operating in the same manner, but at the level of playlists, computing a playlist embedding as, for example, an average of its song embeddings.

However, the strategy of computing a playlist embedding as the average of its song embeddings is sub-optimal. In [139], users are asked to rate each song in a playlist and to rate the playlist as a whole. The paper finds that the playlist rating cannot be predicted as the average of the song ratings. This is in part because there is evidence that including even a single song that the user does not like has a disproportionate effect on the user's opinion of the whole playlist [355, 356]. Moreover, factors like song coherence/diversity and ordering are important quality criteria for playlists (see Section 6.1.2), which are not taken into account by a simple average.

An alternative, found in some work, is the idea of *learning* a playlist embedding. An example of this can be found in the work of Patwari et al. [273]. However, their evaluation is small-scale, qualitative and concerned with the familiar task of suggesting a song to extend a playlist, rather than the task of playlist recommendation that is the subject of this section. A different approach can be found in [350]. The approach taken is to construct a dataset of playlists listened to in sequence by a user (e.g., by taking chronological slices of a user's listening log), and then learning a playlist embedding from "current" playlists that can predict "future" playlists in the sequence. In contrast to [273], Vintch validates the learned embeddings in several downstream tasks, including

[28]We are grateful to an anonymous referee for suggesting that this may be one reason why there is a preference for playlist generation over playlist recommendation.

recommendation, showing that learned embeddings largely outperform average embeddings [350]. Other work that uses learned embeddings includes [57, 267].

In general, we see great potential for research into developing powerful playlist embedding representations beyond a simple average, which can power the next generation of playlist recommendation systems and, more generally, of systems that operate on playlists.

We note too that recommending that an individual user listens to a playlist is only one setting. RSs might also recommend playlists to groups of users. Or we can imagine that they recommend CPs that the user could join, or that they recommend people with whom a user might collaborate. These possibilities are, to the best our knowledge, not yet investigated.

9 Conclusion

In this article, we have surveyed two decades of research into music playlists, adopting the definition that a playlist is a sequence of songs, intended to be listened to together. Our survey was conducted using a systematic and reproducible methodology. The result, while not claiming to being exhaustive in its selection of citations, is nevertheless the most comprehensive survey to date. At top-level, it covers three main subjects: APG (Section 5), MPG (Section 6) and enhanced playlists (Section 7).

We divide the section on APG into two: automatic generation of playlists for individual users (APG-I, Section 5.1), and for groups of users (APG-G, Section 5.2). For our review of APG-I, we adopt a classification from Bonnin and Jannach's 2014 survey [39], which allows us to show how interest in different topics has changed in the last decade. Specifically, we present the systems in the surveyed papers in terms of the background knowledge that they assume, the way they obtain the target characteristics of the playlists they are to construct, and the type of algorithms they use. We also discuss how researchers evaluate their APG-I algorithms.

We believe we are the first to survey work on APG-G. Inspired in part by work on group RSs, we survey APG-G systems in terms of preference acquisition, preference aggregation, and playlist construction, making connections to the classification of APG-I algorithms where relevant. As with APG-I, we conclude with a discussion of the evaluation of these algorithms.

Our survey has revealed many changes in APG algorithms. In terms of their background knowledge, we see a shift away from content-based data and metadata towards usage data— although, in the case of the latter, there has also been a shift away from ratings towards listening logs and manually constructed playlists. In terms of how a user specifies the target characteristics of a playlist, the use of seed songs has grown ever more prevalent. There has been some growth too in the use of free-form text. The recent explosion in the capabilities of natural language processing systems may see even greater use of free-form text in APG in the coming years. In terms of algorithm type, we see more use of embeddings across all the algorithm types. The rise of deep learning more generally has had a profound effect. Most obviously, it has given rise to algorithms that use RNNs and transformers for sequence modeling. But we find it too in some algorithms using MF for CF and in RL, which itself has grown hugely. A final observation is that our survey reveals that there has been very little recent work on APG-G.

Turning to MPG, we adopt the same division into two: MPG for individual users (MPG-I, Section 6.1) and for groups of users (MPG-G, Section 6.2). Most of this literature has not, to the best of our knowledge, been surveyed before. We consider the themes that users choose for the playlists that they construct (relating this to the target characteristics in APG), and the way they choose and order the songs. We reviewA-MPG, where algorithmic tools help the users to construct their playlists. Our survey of A-MPG extends the one by Dias et al. from 2017 [96]. We find that approaches to A-MPG have changed, reflecting the advent of large music collections: where visualization was common, recommendation is now dominant.

As with APG-G, we believe we are the first to survey work on MPG-G. Within this, the topics we survey are similar to those for MPG-I, but additionally we survey the group dynamics, such as the roles people play and their concerns for privacy. Interestingly, while there is not very much recent research on APG-G, most of the work on MPG-G is fairly recent, perhaps reflecting the fact that streaming services offer facilities that make it easy for people to construct playlists collaboratively.

Our survey also includes a review of work on what we call enhanced playlists—work that does not directly address playlist construction. We find four main topics: cross-fading one song into the next; interposing speech between consecutive songs; playlist tagging; and playlist captioning. Work on these is all in its infancy but it holds great promise, not least since it can be integrated into future work on playlist construction.

So much has been learned over the first two decades of research into music playlists. Yet we were still able to conclude our survey with a substantial set of directions for future research. We chose to highlight directions that we find intriguing. They will form only a small part of the research that we can expect to see over the next two decades.

Acknowledgment

The authors are grateful to the anonymous reviewers for their comments.

References

[1] Hiba Abderrazik, Giovan Angela, Hans Brouwer, Henky Janse, Sterre Lutz, Gwennan Smitskamp, Sandy Manolios, and Cynthia C. S. Liem. 2019. Spotivibes: Tagging playlist vibes with colors. In *Proceedings of the 6th Joint Workshop on Interfaces and Human Decision Making for Recommender Systems*, 55–59.

[2] Gediminas Adomavicius, Konstantin Bauman, Alexander Tuzhilin, and Moshe Unger. 2022. Context-aware recommender systems: From foundations to recent developments. In *Recommender Systems Handbook*. F. Ricci, L. Rokach, and B. Shapira (Eds.), Springer, 211–250.

[3] Natalie Aizenberg, Yehuda Koren, and Oren Somekh. 2012. Build your own music recommender by modeling internet radio streams. In *Proceedings of the International Conference on World Wide Web*, 1–10.

[4] Daniel Akimchuk, Timothy Clerico, and Douglas Turnbull. 2019. Evaluating recommender system algorithms for generating local music playlists. arXiv:1907.08687. Retrieved from https://doi.org/10.48550/arxiv:1907.08687

[5] Masoud Alghoniemy and Ahmed H. Tewfik. 2000. User-defined music sequence retrieval. In *Proceedings of the ACM International Conference on Multimedia*, 356–358.

[6] Masoud Alghoniemy and Ahmed H. Tewfik. 2001. A network flow model for playlist generation. In *Proceedings of the IEEE International Conference on Multimedia and Expo*.

[7] David Paul Allen, Henry Jacob Wheeler-Mackta, and Jeremy R. Campo. 2017. The effects of music recommendation engines on the filter bubble phenomenon. Worcester Polytechnic Institute, Worcester, MA.

[8] Marcos Alves de Almeida, Carolina Coimbra Vieira, Pedro Olmo Stancioli Vaz De Melo, and Renato Martins Assunção. 2019. Random playlists smoothly commuting between styles. *ACM Transactions on Multimedia Computing, Communications, and Applications* 15, 4 (2019), 104:1–104:20.

[9] Caroline Anbuhl. 2018. *Social and Cultural Practices Around Using the Music Streaming Provider Spotify – A Qualitative Study Exploring How German Millennials Use Spotify*. Master's thesis. Malmö universitet/Kultur och samhälle.

[10] Andreja Andric and Goffredo Haus. 2005. Estimating quality of playlists by sight. In *Proceedings of the International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution*. IEEE.

[11] Andreja Andric and Goffredo Haus. 2006. Automatic playlist generation based on tracking user's listening habits. *Multimedia Tools and Applications* 29, 2 (2006), 127–151.

[12] Sebastiano Antenucci, Simone Boglio, Emanuele Chioso, Ervin Dervishaj, Shuwen Kang, Tommaso Scarlatti, and Maurizio Ferrari Dacrema. 2018. Artist-driven layering and user's behaviour impact on recommendations in a playlist continuation scenario. In *Proceedings of the ACM Recommender Systems Challenge*, 1–6.

[13] William G. Assuncao, Lara S. G. Piccolo, and Luciana A. M. Zaina. 2022. Considering emotions and contextual factors in music recommendation: A systematic literature review. *Multimedia Tools and Applications* 81 (2022), 8367—8407.

[14] Jean-Julien Aucouturier and Francois Pachet. 2002. Finding songs that sound the same. In *Proceedings of the IEEE Benelux Workshop on Model Based Processing and Coding of Audio*, 1–8.

[15] J.-J. Aucouturier and François Pachet. 2002. Scaling up music playlist generation. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, Vol. 1, 105–108.

[16] Paolo Avesani, Paolo Massa, Michele Nori, and Angelo Susi. 2002. Collaborative radio community. In *Proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, 462–465.

[17] Fredrik Axelsson and Mattias Östergren. 2002. SoundPryer: Joint music listening on the road. In *Consuming Music Together*, K. O'Hara, and B. Brown, (Eds.), 173–190.

[18] Claudio Baccigalupo and Enric Plaza. 2006. Case-based sequential ordering of songs for playlist recommendation. In *Proceedings of the European Conference on Case-Based Reasoning*, 286–300.

[19] Claudio Baccigalupo and Enric Plaza. 2007. Sharing and combining listening experience: A social approach to web radio. In *Proceedings of the International Conference on Computer Music*, 228–231.

[20] Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*.

[21] Wietse Balkema and Ferdi van der Heijden. 2010. Music playlist generation by assimilating GMMs into SOMs. *Pattern Recognition Letters* 31, 11 (2010), 1396–1402.

[22] Luke Barrington, Reid Oda, and Gert R. G. Lanckriet. 2009. Smarter than genius? Human evaluation of music recommender systems. In *Proceedings of the International Society for Music Information Retrieval Conference*, Vol. 9, 357–362.

[23] Arianna Bassoli, Julian Moore, and Stefan Agamanolis. 2006. Tuna: Socialising music sharing on the move. In *Consuming Music Together*. K. O'Hara and B. Brown (Eds.), Springer, 151–172.

[24] Sumit Basu. 2004. Mixing with Mozart. In *Proceedings of the International Computer Music Conference*.

[25] Christine Bauer and Bruce Ferwerda. 2020. Conformity behavior in group playlist creation. In *Proceedings of the Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, 1–10.

[26] Jared S. Bauer, Alex Jansen, and Jesse Cirimele. 2011. MoodMusic: A method for cooperative, generative music playlist creation. In *Proceedings of the Annual ACM Adjunct Symposium on User Interface Software and Technology*, 85–86.

[27] Jared S Bauer, Aubury L. Jellenek, and Julie A. Kientz. 2018. Reflektor: An exploration of collaborative music playlist creation for social context. In *Proceedings of the ACM Conference on Supporting Groupwork*, 27–38.

[28] Dominikus Baur, Sebastian Boring, and Andreas Butz. 2010. Rush: Repeated recommendations on mobile devices. In *Proceedings of the International Conference on Intelligent User Interfaces*, 91–100.

[29] Dominikus Baur, Bernhard Hering, Sebastian Boring, and Andreas Butz. 2011. Who needs interaction anyway: Exploring mobile playlist creation from manual to automatic. In *Proceedings of the International Conference on Intelligent User Interfaces*, 291–294.

[30] Morteza Behrooz, Sarah Mennicken, Jennifer Thom, Rohit Kumar, and Henriette Cramer. 2019. Augmenting music listening experiences on voice assistants. In *Proceedings of the International Society for Music Information Retrieval Conference*, 303–310.

[31] Alejandro Bellogín, Pablo Castells, and Iván Cantador. 2017. Statistical biases in information retrieval metrics for recommender systems. *Information Retrieval Journal* 20, 6 (2017), 606–634.

[32] Shay Ben-Elazar, Gal Lavee, Noam Koenigstein, Oren Barkan, Hilik Berezin, Ulrich Paquet, and Tal Zaccai. 2017. Groove radio: A Bayesian hierarchical model for personalized playlist generation. In *Proceedings of the ACM International Conference on Web Search and Data Mining*, 445–453.

[33] Walid Bendada, Théo Bontempelli, Mathieu Morlon, Benjamin Chapus, Thibault Cador, Thomas Bouabça, and Guillaume Salha-Galvan. 2023. Track mix generation on music streaming services using transformers. In *Proceedings of the 17th ACM Conference on Recommender Systems*. ACM, 112–115.

[34] Walid Bendada, Guillaume Salha-Galvan, Thomas Bouabça, and Tristan Cazenave. 2023. A scalable framework for automatic playlist continuation on music streaming services. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 464–474.

[35] Jacob T. Biehl, Piotr D. Adamczyk, and Brian P. Bailey. 2006. DJogger: A mobile dynamic music device. In *Proceedings of the Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, 556–561.

[36] Rachel M. Bittner, Minwei Gu, Gandalf Hernandez, Eric J. Humphrey, Tristan Jehan, Hunter McCurry, and Nicola Montecchio. 2017. Automatic playlist sequencing and transitions. In *Proceedings of the International Society for Music Information Retrieval Conference*, 442–448.

[37] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research* 3, (2003), 993–1022.

[38] Geoffray Bonnin and Dietmar Jannach. 2013. Evaluating the quality of playlists based on hand-crafted samples. In *Proceedings of the International Society for Music Information Retrieval Conference*, 263–268.

[39] Geoffray Bonnin and Dietmar Jannach. 2014. Automated generation of music playlists: Survey and experiments. *Computing Surveys* 47, 2 (2014), 1–35.

[40] Théo Bontempelli, Benjamin Chapus, François Rigaud, Mathieu Morlon, Marin Lorant, and Guillaume Salha-Galvan. 2022. Flow moods: Recommending music by moods on deezer. In *Proceedings of the ACM Conference on Recommender Systems*, 452–455.

[41] Klaas Bosteels, Elias Pampalk, and Etienne E. Kerre. 2009. Evaluating and analysing dynamic playlist generation heuristics using radio logs and fuzzy set theory. In *Proceedings of the International Society for Music Information Retrieval Conference*, Vol. 9, 351–356.

[42] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 1877–1901.

[43] Karan Kumar Budhraja, Ashutosh Singh, Gaurav Dubey, and Arun Khosla. 2012. Probability based playlist generation based on music similarity and user customization. In *Proceedings of the National Conference on Computing and Communication Systems*, 1–5.

[44] Robin Burke. 2002. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction* 12, 4 (2002), 331–370.

[45] Daniel Burnett, Adrian Gradinar, Joel Porter, Mike Stead, Paul Coulton, and Ian Forrester. 2015. Physical playlist: Bringing back the mix-tape. In *Human-Computer Interaction (INTERACT '15)*. Springer International Publishing, 72–78.

[46] Rui Cai, Chao Zhang, Lei Zhang, and Wei-Ying Ma. 2007. Scalable music recommendation by search. In *Proceedings of the ACM International Conference on Multimedia*, 1065–1074.

[47] Joseph A. Calandrino, Ann Kilzer, Arvind Narayanan, Edward W. Felten, and Vitaly Shmatikov. 2011. "You might also like": Privacy risks of collaborative filtering. In *Proceedings of the IEEE Symposium on Security and Privacy*. IEEE, 231–246.

[48] Luís Cardoso, Renato Panda, and Rui Pedro Paiva. 2011. MOODetector: A prototype software tool for mood-based playlist generation. In *Proceedings of the Simpósio de Informática*.

[49] Arturo P. Caronongan and Rafael A. Cabredo. 2016. Modeling user music preference through usage scoring and user listening behavior for generating preferred playlists. In *Principles and Practice of Multi-Agent Systems*. M. Baldoni, C. Baroglio, F. Bex, F. Grasso, N.L. Green, M.-R. Namazi-Rad, M. Numao, and M. T. C. Suarez (Eds.), Springer International Publishing, 63–73.

[50] Michael Castelluccio. 2006. The music genome project. *Strategic Finance* 88, 6 (2006), 57–59.

[51] Oscar Celma. 2010. *Music Recommendation and Discovery: The Long Tail, Long Fail, and Long Play in the Digital Music Space*. Springer Science & Business Media.

[52] Arun Tejasvi Chaganty, Megan Leszczynski, Shu Zhang, Ravi Ganti, Krisztian Balog, and Filip Radlinski. 2023. Beyond single items: Exploring User Preferences in Item Sets with the Conversational Playlist Curation Dataset. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. Association for Computing Machinery, 2754—2764.

[53] Sylvia Chan-Olmsted, Rang Wang, and Kyung-Ho Hwang. 2020. Substitutability and complementarity of broadcast radio and music streaming services: The millennial perspective. *Mobile Media & Communication* 8, 2 (2020), 209–228.

[54] Dennis L. Chao, Justin Balthrop, and Stephanie Forrest. 2005. Adaptive radio: Achieving consensus using negative preferences. In *Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work*, 120–123.

[55] Amar Chaudhary. 2002. An extensible representation for playlists. In *Proceedings of the International Society for Music Information Retrieval Conference*.

[56] Zeina Chedrawy and Syed Sibte Raza Abidi. 2009. A web recommender system for recommending, predicting and personalizing music playlists. In *International Conference on Web Information Systems Engineering*, 335–342.

[57] Chih-Ming Chen, Chun-Yao Yang, Chih-Chun Hsia, Yian Chen, and Ming-Feng Tsai. 2016. Music playlist recommendation via preference embedding. In *Proceedings of the Poster Track of the 10th ACM Conference on Recommender Systems*.

[58] Ching-Wei Chen, Kyogu Lee, and Ho-Hsiang Wu. 2009. Towards a class-based representation of perceptual tempo for music retrieval. In *Proceedings of the International Conference on Machine Learning and Applications*, 602–607.

[59] Dawei Chen, Cheng Soon Ong, and Aditya Krishna Menon. 2019. Cold-start playlist recommendation with multitask learning. arXiv:1901.06125. Retrieved from https://doi.org/10.48550/arxiv:1901.06125

[60] Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He†. 2022. Bias and debias in recommender system: A survey and future directions. *ACM Transactions on Information Systems*, 67, 39 (2022), 1–67.

[61] Shuo Chen, Josh L Moore, Douglas Turnbull, and Thorsten Joachims. 2012. Playlist prediction via metric embedding. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 714–722.

[62] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A scalable tree boosting system. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794.

[63] Ya-Xi Chen and Andreas Butz. 2009. MusicSim: Integrating audio analysis and user feedback in an interactive music browsing UI. In *Proceedings of the International Conference on Intelligent User Interfaces*, 429–434.

[64] Haonan Cheng, Ruyu Zhang, Xiaoying Huang, and Long Ye. 2022. Global-local similarity function for automatic playlist generation. In *Proceedings of the IEEE International Conference on Multimedia and Expo*. IEEE, 1–6.

[65] Zhiyong Cheng and Jialie Shen. 2014. Just-for-me: An adaptive personalization system for location-aware social music recommendation. In *Proceedings of International Conference on Multimedia Retrieval*, 185–192.

[66] Chung-Yi Chi, Richard Tzong-Han Tsai, Jeng-You Lai, and Jane Yung-jen Hsu. 2010. A reinforcement learning approach to emotion-based automatic playlist generation. In *Proceedings of the International Conference on Technologies and Applications of Artificial Intelligence*. IEEE, 60–65.

[67] Luca Chiarandini, Massimiliano Zanoni, and Augusto Sarti. 2011. A system for dynamic playlist generation driven by multimodal control signals and descriptors. In *Proceedings of the IEEE International Workshop on Multimedia Signal Processing*, 1–6.

[68] Jeong Choi, Anis Khlif, and Elena Epure. 2020. Prediction of user listening contexts for music playlists. In *Proceedings of the Workshop on NLP for Music and Audio (NLP4MusA)*, 23–27.

[69] Keunwoo Choi, George Fazekas, Brian McFee, Kyunghyun Cho, and Mark Sandler. 2017. Towards music captioning: Generating music playlist descriptions. arxiv: 1608.04868. Retrieved from https://doi.org/10.48550/arXiv.1608.04868

[70] Keunwoo Choi, George Fazekas, and Mark Sandler. 2015. Understanding music playlists. arXiv:1511.07004. Retrieved from https://doi.org/10.48550/arxiv:1511.07004

[71] Keunwoo Choi, George Fazekas, and Mark Sandler. 2016. Towards playlist generation algorithms using RNNs trained on within-track transitions. In *Proceedings of the Workshop on Surprise, Opposition, and Obstruction in Adaptive and Personalized Systems*.

[72] Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho. 2017. Convolutional recurrent neural networks for music classification. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, 2392–2396.

[73] Chia-Hao Chung, Yian Chen, and Homer H. Chen. 2017. Exploiting playlists for representation of songs and words for text-based music retrieval. In *Proceedings of the International Society for Music Information Retrieval Conference*, 478–485.

[74] Dave Cliff. 2000. *Hang the DJ: Automatic sequencing and seamless mixing of dance-music tracks*. Technical Report. HP Laboratories Technical Report.

[75] Filipe Coelho, José Devezas, and Cristina Ribeiro. 2013. Large-scale crossmedia retrieval for playlist generation and song discovery. In *Proceedings of the Conference on Open Research Areas in Information Retrieval*, 61–64.

[76] Yandre M. G. Costa, Luiz S. Oliveira, Alessandro L. Koericb, and Fabien Gouyon. 2011. Music genre recognition using spectrograms. In *Proceedings of the International Conference on Systems, Signals and Image Processing*, 1–4.

[77] Michel Crampes, Jean Villerd, Andrew Emery, and Sylvie Ranwez. 2007. Automatic playlist composition in a dynamic music landscape. In *Proceedings of the International Workshop on Semantically Aware Document Processing and Indexing*, 15–20.

[78] Andrew Crossen, Jay Budzik, and Kristian J. Hammond. 2002. Flytrap: Intelligent group music recommendation. In *Proceedings of the International Conference on Intelligent User Interfaces*, 184–185.

[79] Stuart Cunningham, Stephen Caulder, and Vic Grout. 2008. Saturday night or fever? Context-aware music playlists. In *Proceedings of the International Audio Mostly Conference*.

[80] Sally Jo Cunningham, David Bainbridge, and Annette Bainbridge. 2017. Exploring personal music collection behavior. In *Proceedings of the International Conference on Asian Digital Libraries*. Springer, 295–306.

[81] Sally Jo Cunningham, David Bainbridge, and Annette Falconer. 2006. 'More of an art than a science': Supporting the creation of playlists and mixes. In *Proceedings of the International Society for Music Information Retrieval Conference*.

[82] Sally Jo Cunningham, Matt Jones, and Steve Jones. 2004. Organizing digital music for use: An examination of personal music collections. In *Proceedings of the International Society for Music Information Retrieval Conference*.

[83] Sally Jo Cunningham and David M. Nichols. 2009. Exploring social music behaviour: An investigation of music selection at parties. In *Proceedings of the International Society for Music Information Retrieval Conference,* 747–752.

[84] Sally Jo Cunningham, David M. Nichols, David Bainbridge, and Hasan Ali. 2014. Social music in cars. In *Proceedings of the International Society for Music Information Retrieval Conference*, 457–462.

[85] Cedric De Boom, Rohan Agrawal, Samantha Hansen, Esh Kumar, Romain Yon, Ching-Wei Chen, Thomas Demeester, and Bart Dhoedt. 2018. Large-scale user modeling with recurrent neural networks for music discovery on multiple time scales. *Multimedia Tools and Applications* 77, 12 (2018), 15385–15407.

[86] A. M. De Mooij and W. F. J. Verhaegh. 1997. Learning preferences for music playlists. *Artificial Intelligence* 97, 1–2 (1997), 245–271.

[87] Igor de Oliveira Nunes, Gabriel Matos Cardoso Leite, and Daniel Ratton Figueiredo. 2019. A contextual hierarchical graph model for generating random sequences of objects with application to music playlists. In *Proceedings of the Brazilian Conference on Intelligent Systems*. IEEE, 72–77.

[88] Brian Dean. 2024. Spotify user stats. https://backlinko.com/spotify-Users
[89] Yashar Deldjoo, Maurizio Ferrari Dacrema, Mihai Gabriel Constantin, Hamid Eghbal-Zadeh, Stefano Cereda, Markus Schedl, Bogdan Ionescu, and Paolo Cremonesi. 2019. Movie genome: Alleviating new item cold start in movie recommendation. *User Modeling and User-Adapted Interaction* 29, 2 (2019), 291–343.
[90] Shuiguang Deng, Dongjing Wang, Xitong Li, and Guandong Xu. 2015. Exploring user emotion in microblogs for music recommendation. *Expert Systems with Applications* 42, 23 (2015), 9284–9293.
[91] Mukund Deshpande and George Karypis. 2004. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems* 22, 1 (2004), 143–177.
[92] Frederik Dhaenens and Jean Burgess. 2019. 'Press play for pride': The cultural logics of LGBTQ-themed playlists on spotify. *New Media & Society* 21, 6 (2019), 1192—211.
[93] Ricardo Dias, Ricardo Cunha, and Manuel J. Fonseca. 2014. A user-centered music recommendation approach for daily activities. In *Proceedings of the 1st Workshop on New Trends in Content-based Recommender Systems,* 26–32.
[94] Ricardo Dias and Manuel J. Fonseca. 2010. Muvis: An application for interactive exploration of large music collections. In *Proceedings of the ACM International Conference on Multimedia*, 1043–1046.
[95] Ricardo Dias, Daniel Gonçalves, and Manuel J. Fonseca. 2016. PlaylistCreator: An assisted approach for playlist creation. In *Proceedings of the ACM International Conference on Multimedia*, 711–713.
[96] Ricardo Dias, Daniel Gonçalves, and Manuel J. Fonseca. 2017. From manual to assisted playlist creation: A survey. *Multimedia Tools and Applications* 76, 12 (2017), 14375–14403.
[97] Seungheon Doh, Junwon Lee, and Juhan Nam. 2021. Music playlist title generation: A machine-translation approach. In *Proceedings of the Workshop on NLP for Music and Spoken Audio (NLP4MuSA '21).*
[98] Markus Dopler, Markus Schedl, Tim Pohle, and Peter Knees. 2008. Accessing music collections via representative cluster prototypes in a hierarchical organization scheme. In *Proceedings of the International Society for Music Information Retrieval Conference,* 179–184.
[99] Clemens Drews and Florian Pestoni. 2002. Virtual jukebox: Reviving a classic. In *Proceedings of the Annual Hawaii International Conference on System Sciences*, 887–893.
[100] Greg T. Elliott and Bill Tomlinson. 2006. PersonalSoundtrack: Context-aware playlists that adapt to user pace. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems,* 736–741.
[101] Daniel P. W. Ellis. 2006. Extracting information from music audio. *Communications of the ACM* 49, 8 (2006), 32–37.
[102] Guglielmo Faggioli, Mirko Polato, and Fabio Aiolli. 2018. Efficient similarity based methods for the playlist continuation task. In *Proceedings of the ACM Recommender Systems Challenge*, 1–6.
[103] Dieter Fensel, Umutcan Şimşek, Kevin Angele, Elwin Huaman, Elias Kärle, Oleksandra Panasiuk, Ioan Toma, Jürgen Umbrich, and Alexander Wahler. 2020. Introduction: What is a knowledge graph? In *Knowledge Graphs*. Springer, 1–10.
[104] Andres Ferraro, Dmitry Bogdanov, Jisang Yoon, KwangSeob Kim, and Xavier Serra. 2018. Automatic playlist continuation using a hybrid recommender system combining features from text and audio. In *Proceedings of the ACM Recommender Systems Challenge*, 1–5.
[105] Andres Ferraro, Yuntae Kim, Soohyeon Lee, Biho Kim, Namjun Jo, Semi Lim, Suyon Lim, Jungtaek Jang, Sehwan Kim, and Xavier Serra. 2021. Melon playlist dataset: A public dataset for audio-based playlist generation and music tagging. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*. IEEE, 536–540.
[106] Andres Ferraro, Xavier Serra, and Christine Bauer. 2021. Break the loop: Gender imbalance in music recommenders. In *Proceedings of the Conference on Human Information Interaction and Retrieval*. ACM, 249—254.
[107] Ben Fields and Paul Lamere. 2010. Finding a path through the juke box: The playlist tutorial. In *Proceedings of the International Society for Music Information Retrieval Conference*.
[108] Benjamin Fields, Christophe Rhodes, Michael A. Casey, and Kurt Jacobson. 2008. Social playlists and bottleneck measurements: Exploiting musician social graphs using content-based dissimilarity and pairwise maximum flow values. In *Proceedings of the International Society for Music Information Retrieval Conference*, 559–564.
[109] Arthur Flexer. 2014. On inter-rater agreement in audio music similarity. In *Proceedings of the International Society for Music Information Retrieval Conference,* 245–250.
[110] Arthur Flexer, Taric Lallai, and Katja Rašl. 2021. On evaluation of inter-and intra-rater agreement in music recommendation. *Transactions of the International Society for Music Information Retrieval,* 4, 2021 (1), 182–194.
[111] Arthur Flexer, Dominik Schnitzer, Martin Gasser, and Gerhard Widmer. 2008. Playlist Generation using Start and End Songs. In *Proceedings of the International Society for Music Information Retrieval Conference*, Vol. 8, 173–178.
[112] Mathew Flynn. 2016. Accounting for Listening: How Music Streaming Has Changed What it Means to Listen. *Kinephanos-Journal of Media Studies and Popular Culture* 6, 1 (2016).
[113] Ariana Moscote Freire. 2008. Remediating Radio: Audio Streaming, Music Recommendation and the Discourse of Radioness. *Radio Journal: International Studies in Broadcast & Audio Media* 5, 2–3 (2008), 97–112.

[114] John Fuller, Lauren Hubener, Yea-Seul Kim, and Jin Ha Lee. 2016. Elucidating User Behavior in Music Services Through Persona and Gender. In *Proceedings of the International Society for Music Information Retrieval Conference*, 626–632.
[115] Marco Furini and Sabrina D'arcangelo. 2021. Automatic and User-Tailored Playlist Sequencing. In *Proceedings of the Conference on Information Technology for Social Good*, 321–324.
[116] Marco Furini, Jessica Martini, and Manuela Montangero. 2019. Automated Generation of User-tailored and Time-sensitive Music Playlists. In *Proceedings of the IEEE Annual Consumer Communications & Networking Conference*. IEEE, 1–6.
[117] Marco Furini and Manuela Montangero. 2022. Automatic and Personalized Sequencing of Music Playlists. In *Proceedings of the IEEE International Conference on Distributed Computing Systems Workshops*. IEEE, 205–208.
[118] Giovanni Gabbolini and Derek Bridge. 2021. Generating Interesting Song-to-song Segues with Dave. In *Proceedings of the ACM Conference on User Modeling, Adaptation and Personalization*, 98–107.
[119] Giovanni Gabbolini and Derek Bridge. 2021. Play It Again, Sam! Recommending Familiar Music in Fresh Ways. In *Proceedings of the ACM Conference on Recommender Systems*, 697–701.
[120] Giovanni Gabbolini and Derek Bridge. 2022. A User-centered Investigation of Personal Music Tours. In *Proceedings of the ACM Conference on Recommender Systems*, 25–34.
[121] Giovanni Gabbolini and Derek Bridge. 2023. Predicting the Listening Contexts of Music Playlists using Knowledge Graphs. In *Advances in Information Retrieval: European Conference on IR Research*. Springer, 330–345.
[122] Giovanni Gabbolini, Romain Hennequin, and Elena Epure. 2022. Data-efficient Playlist Captioning With Musical and Linguistic Knowledge. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 11401–11415.
[123] Don Gartner, Florian Kraft, and Thomas Schaaf. 2007. An Adaptive Distance Measure for Similarity Based Playlist Generation. In *Proceedings of the 2007 IEEE International Conference on Acoustics, Speech and Signal Processing*, I-229–I-232. DOI: https://doi.org/10.1109/ICASSP.2007.366658
[124] Anna Gatzioura and Miquel Sànchez-Marrè. 2017. A Case-based Reasoning Framework for Music Playlist Recommendations. In *Proceedings of the International Conference on Control, Decision and Information Technologies*, 0242–0247.
[125] Anna Gatzioura and Miquel Sànchez-Marrè. 2017. Using Contextual Information in Music Playlist Recommendations. In *Proceedings of the International Conference of the Catalan Association for Artificial Intelligence*, 239–244.
[126] Anna Gatzioura, Miquel Sànchez-Marrè, and Alípio Mário Jorge. 2018. A Study on Contextual Influences on Automatic Playlist Continuation. In *Proceedings of the International Conference of the Catalan Association for Artificial Intelligence*, Vol. 308, 156–165.
[127] Anna Gatzioura, João Vinagre, Alípio Mário Jorge, and Miquel Sanchez-Marre. 2019. A Hybrid Recommender System for Improving Automatic Playlist Continuation. *IEEE Transactions on Knowledge and Data Engineering* 33, 5 (2019), 1819–1830.
[128] Roman B. Gebhardt, Matthew E. P. Davies, and Bernhard U. Seeber. 2016. Psychoacoustic Approaches for Harmonic Music Mixing. *Applied Sciences* 6, 5 (2016), 123.
[129] Thanawit Gerdprasert, Pipatpol Tanavongchinda, Pakpon Jetapai, and Hutchatai Chanlekha. 2018. A Study of Playlist Generation Technique for Moderate Computational Resource Environment. In *Proceedings of the 10th International Conference on Management of Digital EcoSystems*. ACM, 227–232.
[130] Arthur Germain and Jacob Chakareski. 2013. Spotify Me: Facebook-Assisted Automatic Playlist Generation. In *Proceedings of the IEEE International Workshop on Multimedia Signal Processing*, 25–28.
[131] Masataka Goto and Takayuki Goto. 2005. Musicream: New Music Playback Interface for Streaming, Sticking, Sorting, and Recalling Musical Pieces. In *Proceedings of the International Society for Music Information Retrieval Conference*, 404–411.
[132] Olga Goussevskaia, Michael Kuhn, and Roger Wattenhofer. 2008. Exploring Music Collections on Mobile Devices. In *Proceedings of the International Conference on Human Computer Interaction with Mobile Devices and Services*, 359–362.
[133] Fabien Gouyon, Nuno Cruz, and Luis Sarmento. 2011. A last.fm and youtube Mash-up for Music Browsing and Playlist Edition. In *Proceedings of the Late-Breaking Demo Session at the International Music Information Retrieval Conference*.
[134] Jacek Grekow. 2016. Music Emotion Maps in Arousal-valence Space. In *Computer Information Systems and Industrial Management: IFIP TC8 International Conference*. Springer, 697–706.
[135] Darryl Griffiths, Stuart Cunningham, and Jonathan Weinel. 2013. A Discussion of Musical Features for Automatic Music Playlist Generation using Affective Technologies. In *Proceedings of the Audio Mostly Conference*, 1–4.
[136] Darryl Griffiths, Stuart Cunningham, and Jonathan Weinel. 2016. An Interactive Music Playlist Generator That Responds to User Emotion and Context. *Electronic Visualisation and the Arts*, 275–276.

[137] Peter Grosche, Meinard Müller, and Craig Stuart Sapp. 2010. What Makes Beat Tracking Difficult? A Case Study on Chopin Mazurkas. In *Proceedings of the International Society for Music Information Retrieval Conference*, 649–654.
[138] Alois Gruson, Praveen Chandar, Christophe Charbuillet, James McInerney, Samantha Hansen, Damien Tardieu, and Ben Carterette. 2019. Offline Evaluation to Make Decisions about Playlist Recommendation Algorithms. In *Proceedings of the ACM International Conference on Web Search and Data Mining*, 420–428.
[139] Sophia Hadash, Yu Liang, and Martijn C Willemsen. 2019. How Playlist Evaluation Compares to Track Evaluations in Music Recommender Systems. In *Proceedings of the Joint Workshop on Interfaces and Human Decision Making for Recommender Systems*, 1–9.
[140] Anja Nylund Hagen. 2015. The Playlist Experience: Personal Playlists in Music Streaming Services. *Popular Music and Society* 38, 5 (2015), 625–645.
[141] Maria Håkansson, Mattias Rost, and Lars Erik Holmquist. 2007. Gifts from Friends and Strangers: A Study of Mobile Music Sharing. In *Proceedings of the European Conference on Computer-Supported Cooperative Work*. Springer, 311–330.
[142] Jiawei Han, Jian Pei, Behzad Mortazavi-Asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Meichun Hsu. 2001. Prefixspan: Mining Sequential Patterns Efficiently by Prefix-projected Pattern Growth. In *Proceedings of the International Conference on Data engineering*. IEEE, 215–224.
[143] Derek L Hansen and Jennifer Golbeck. 2009. Mixing it up: Recommending Collections of Items. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 1217–1226.
[144] Negar Hariri, Bamshad Mobasher, and Robin Burke. 2012. Context-aware Music Recommendation based on Latent Topic Sequential Patterns. In *Proceedings of the ACM Conference on Recommender Systems*, 131–138.
[145] Pitoyo Hartono and Ryo Yoshitake. 2013. Automatic playlist Generation from Self-organizing Music Map. *Journal of Signal Processing* 17, 1 (2013), 11–19.
[146] Andrew C. Harvey. 1990. *Forecasting, Structural Time Series Models and the Kalman Filter*. Cambridge University Press.
[147] David B. Hauver and James C. French. 2001. Flycasting: Using Collaborative Filtering to Generate a Playlist for Online Radio. In *Proceedings of the International Conference on WEB Delivering of Music*, 123–130.
[148] Conor Hayes and Pádraig Cunningham. 2001. Smart Radio — Community Based Music Radio. *Knowledge-Based Systems* 14, 3–4 (2001), 197–201.
[149] Perfecto Herrera, Zuriñe Resa, and Mohamed Sordo. 2010. Rocking around the Clock Eight Days a Week: An Exploration of Temporal Patterns of Music Listening. In *Proceedings of the Workshop On Music Recommendation And Discovery at the ACM Recommender Systems Conference*.
[150] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based Recommendations with Recurrent Neural Networks. arXiv:1511.06939. Retrieved from https://doi.org/10.48550/arXiv:1511.06939
[151] Otmar Hilliges, Phillipp Holzer, Rene Klüber, and Andreas Butz. 2006. Audioradar: A Metaphorical Visualization for the Navigation of Large Music Collections. In *Proceedings of the International Symposium on Smart Graphics*. Springer, 82–92.
[152] Jukka Holm, Antti Aaltonen, and Harri Siirtola. 2009. Associating Colours with Musical Genres. *Journal of New Music Research* 38, 1 (2009), 87–100.
[153] Jukka Holm, Arto Lehtiniemi, and Antti Eronen. 2010. Evaluating an Avatar-based User Interface for Discovering New Music. In *Proceedings of the International Conference on Mobile and Ubiquitous Multimedia*, 1–10.
[154] M. D. Zakir Hossain, Ferdous Sohel, Mohd Fairuz Shiratuddin, and Hamid Laga. 2019. A Comprehensive Survey of Deep Learning for Image Captioning. *ACM Computing Surveys* 51, 6 (2019), 1–36.
[155] Jia-Lien Hsu and Shuk-Chun Chung. 2011. Constraint-based Playlist Generation by Applying Genetic Algorithm. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, 1417–1422.
[156] Jia-Lien Hsu and Ya-Chao Lai. 2014. Automatic Playlist Generation by Applying Tabu Search. *International Journal of Machine Learning and Cybernetics* 5 (2014), 553–568.
[157] Binbin Hu, Chuan Shi, and Jian Liu. 2017. Playlist Recommendation based on Reinforcement Learning. In *Proceedings of the International Conference on Intelligence Science*, 172–182.
[158] Yajie Hu and Mitsunori Ogihara. 2011. NextOne Player: A Music Recommendation System Based on User Behavior. In *Proceedings of the International Society for Music Information Retrieval Conference*, Vol. 11, 103–108.
[159] Karim M Ibrahim, Elena V. Epure, Geoffroy Peeters, and Gaël Richard. 2022. Exploiting Device and Audio Data to Tag Music with User-Aware Listening Contexts. In *Proceedings of the 23rd International Society for Music Information Retrieval Conference*, 186–192.
[160] IFPI. 2024. Global Music Report. Retrieved from https://globalMusicreport.ifpi.org/
[161] Shobu Ikeda, Kenta Oku, and Kyoji Kawagoe. 2018. Music Playlist Recommendation using Acoustic-feature Transition inside the Songs. In *Proceedings of the International Conference on Advances in Mobile Computing & Multimedia*, 216–219.

[162] Rosilde Tatiana Irene, Clara Borrelli, Massimiliano Zanoni, Michele Buccoli, and Augusto Sarti. 2019. Automatic Playlist Generation using Convolutional Neural Networks and Recurrent Neural Networks. In *Proceedings of the European Signal Processing Conference*. IEEE, 1–5.
[163] Hiromi Ishizaki, Keiichiro Hoashi, and Yasuhiro Takishima. 2009. Full-Automatic DJ Mixing System with Optimal Tempo Adjustment Based on Measurement Function of User Discomfort. In *Proceedings of the International Society for Music Information Retrieval Conference*, 135–140.
[164] Toshi-Hiro Ito and Hiroaki Shiokawa. 2024. An Effective Graph-based Music Recommendation Algorithm for Automatic Playlist Continuation. In *Proceedings of the 2023 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. ACM, 459—463.
[165] Aleksandar Ivanovski, Milos Jovanovik, Riste Stojanov, and Dimitar Trajanov. 2023. Knowledge Graph Based Recommender for Automatic Playlist Continuation. *Information* 14, 9.
[166] Mátyás Jani. 2015. Fast Content Independent Playlist Generation for Streaming Media. In *Proceedings of the IEEE/ACS International Conference of Computer Systems and Applications*. IEEE, 1–6.
[167] Mátyás Jani, Gergely Lukács, and György Takács. 2014. Experimental Investigation of Transitions for Mixed Speech and Music Playlist Generation. In *Proceedings of the Proceedings of International Conference on Multimedia Retrieval*, 392–398.
[168] Dietmar Jannach, Iman Kamehkhosh, and Geoffray Bonnin. 2014. Analyzing the Characteristics of Shared Playlists for Music Recommendation. In *Proceedings of the RSWeb Workshop at the ACM Recommender Systems Conference*.
[169] Dietmar Jannach, Iman Kamehkhosh, and Geoffray Bonnin. 2016. Biases in Automated Music Playlist Generation: A Comparison of Next-Track Recommending Techniques. In *Proceedings of the Conference on User Modeling Adaptation and Personalization*, 281–285.
[170] Dietmar Jannach, Iman Kamehkhosh, and Lukas Lerche. 2017. Leveraging Multi-Dimensional User Models for Personalized next-Track Music Recommendation. In *Proceedings of the Symposium on Applied Computing*, 1635–1642.
[171] Dietmar Jannach, Lukas Lerche, and Iman Kamehkhosh. 2015. Beyond "Hitting the Hits": Generating Coherent Music Playlist Continuations with the Right Tracks. In *Proceedings of the ACM Conference on Recommender Systems*, 187–194.
[172] Dietmar Jannach and Malte Ludewig. 2017. When Recurrent Neural Networks Meet the Neighborhood for Session-Based Recommendation. In *Proceedings of the ACM Conference on Recommender Systems*, 306–310.
[173] Tristan Jehan and Ali Mazalek. 2000. Interacting with Music in a Social Setting. In *Proceedings of the ACM Conference on Human Interaction (CHI)*.
[174] Claus Aage Jensen, Ester M Mungure, Torben Bach Pedersen, and Kenneth Sorensen. 2007. A Data and Query Model for Dynamic Playlist Generation. In *Proceedings of the IEEE International Conference on Data Engineering Workshop*, 65–74.
[175] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. 2016. Fasttext.zip: Compressing Text Classification Models. arXiv:1612.03651. Retrieved from https://doi.org/10.48550/arXiv:1612.03651
[176] Carles Fernandes Julià and Sergi Jordà. 2009. SongExplorer: A Tabletop Application for Exploring Large Collections of Songs. In *Proceedings of the International Society for Music Information Retrieval Conference,* 675–680.
[177] Daniel Jurafsky and James H. Martin. 2009. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall.
[178] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A Convolutional Neural Network for Modelling Sentences. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 655–665.
[179] Mohsen Kamalzadeh, Dominikus Baur, and Torsten Möller. 2012. A Survey on Music Listening and Management Behaviours. In *Proceedings of the International Society for Music Information Retrieval Conference,* 373–378.
[180] Mohsen Kamalzadeh, Dominikus Baur, and Torsten Möller. 2016. Listen or Interact? A Large-scale Survey on Music Listening and Management Behaviours. *Journal of New Music Research* 45, 1 (2016), 42–67.
[181] Mohsen Kamalzadeh, Christoph Kralj, Torsten Möller, and Michael Sedlmair. 2016. TagFlip: Active Mobile Music Discovery with Social Tags. In *Proceedings of the International Conference on Intelligent User Interfaces*, 19–30.
[182] Iman Kamehkhosh, Geoffray Bonnin, and Dietmar Jannach. 2020. Effects of Recommendations on the Playlist Creation Behavior of Users. *User Modeling and User-Adapted Interaction* 30, 2 (2020), 285–322.
[183] Iman Kamehkhosh and Dietmar Jannach. 2017. User Perception of Next-track Music Recommendations. In *Proceedings of the Conference on User Modeling, Adaptation and Personalization*, 113–121.
[184] Iman Kamehkhosh, Dietmar Jannach, and Geoffray Bonnin. 2018. How Automated Recommendations Affect the Playlist Creation Behavior of Users. In *Workshops of the ACM Conference on Intelligent User Interfaces*.
[185] Iman Kamehkhosh, Dietmar Jannach, and Lukas Lerche. 2016. Personalized Next-Track Music Recommendation with Multi-dimensional Long-Term Preference Signals. In *Proceedings of the ACM International Conference on User Modelling, Adaptation and Personalization*.

[186] Marius Kaminskas and Francesco Ricci. 2012. Contextual Music Information Retrieval and Recommendation: State of the Art and Challenges. *Computer Science Review* 6, 2–3 (2012), 89–119.
[187] Mesut Kaya and Derek Bridge. 2018. Automatic Playlist Continuation using Subprofile-aware Diversification. In *Proceedings of the ACM Recommender Systems Challenge*, 1–6.
[188] Mesut Kaya and Derek Bridge. 2019. Subprofile-Aware Diversification of Recommendations. *User Modeling and User-Adapted Interaction* 29, 3 (2019), 661–700.
[189] Domokos M. Kelen, Dániel Berecz, Ferenc Béres, and András A Benczúr. 2018. Efficient K-NN for Playlist Continuation. In *Proceedings of the ACM Recommender Systems Challenge*, 1–4.
[190] Haven Kim, SeungHeon Doh, Junwon Lee, and Juhan Nam. 2023. Music Playlist Title Generation Using Artist Information. arXiv:2301.08145. Retrieved from https://doi.org/10.48550/arxiv:2301.08145
[191] Jaehun Kim, Minz Won, Cynthia C. S. Liem, and Alan Hanjalic. 2018. Towards Seed-free Music Playlist Generation: Enhancing Collaborative Filtering with Playlist Title Information. In *Proceedings of the ACM Recommender Systems Challenge*, 1–6.
[192] James King and Vaiva Imbrasaitė. 2015. Generating Music playlists with Hierarchical Clustering and Q-learning. In *Proceedings of the European Conference on Information Retrieval*. Springer, 315–326.
[193] Yngvar Kjus. 2016. Musical Exploration via Streaming Services: The Norwegian Experience. *Popular Communication* 14, 3 (2016), 127–136.
[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 *Proceedings of the ACM International Workshop on Multimedia Information Retrieval*, 147–154.
[195] Peter Knees and Markus Schedl. 2016. Introduction to Music Similarity and Retrieval. In *Music Similarity and Retrieval: An Introduction to Audio- and Web-based Strategies*. Springer, 1–30.
[196] Peter Knees, Markus Schedl, Tim Pohle, and Gerhard Widmer. 2006. An Innovative Three-dimensional User Interface for Exploring Music Collections Enriched. In *Proceedings of the ACM International Conference on Multimedia*, 17–24.
[197] Teuvo Kohonen. 1990. The Self-Organizing Map. *Proceedings of IEEE* 78, 9 (1990), 1464–1480.
[198] Joseph A. Konstan, Bradley N. Miller, David Maltz, Jonathan L. Herlocker, Lee R. Gordon, and John Riedl. 1997. Grouplens: Applying Collaborative Filtering to Usenet News. *Communicatons of ACM* 40, 3 (1997), 77–87.
[199] Burak Köse, Süleyman Eken, and Ahmet Sayar. 2016. Playlist Generation via Vector Representation of Songs. In *Proceedings of the INNS Conference on Big Data*, 179–185.
[200] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2017. Imagenet Classification with Deep Convolutional Neural Networks. *Communications of ACM* 60, 6 (2017), 84–90.
[201] Paul Lamere. 2008. Social Tagging and Music Information Retrieval. *Journal of New Music Research* 37, 2 (2008), 101–114.
[202] Paul Lamere. 2023. Smarter Playlists. Retrieved from http://smarterplaylists.playlistmachinery.com
[203] Paul Lamere and Douglas Eck. 2007. Using 3D Visualizations to Explore and Discover Music. In *Proceedings of the International Society for Music Information Retrieval Conference*, 173–174.
[204] David B. Leake. 1996. CBR In Context: The Present and Future. In *Case-Based Reasoning: Experiences, Lessons, & Future Directions*, David B. Leake (Ed.). MIT Press, 3–30.
[205] Jin Ha Lee, Bobby Bare, and Gary Meek. 2011. How Similar is too Similar?: Exploring Users' Perceptions of Similarity in Playlist Evaluation. In *Proceedings of the International Society for Music Information Retrieval Conference*, Vol. 11, 109–114.
[206] Jin Ha Lee and J. Stephen Downie. 2004. Survey of Music Information Needs, Uses, and Seeking Behaviours: Preliminary Findings. In *Proceedings of the International Society for Music Information Retrieval Conference*, Vol. 5.
[207] Arto Lehtiniemi and Jukka Holm. 2011. Easy Access to Recommendation Playlists: Selecting Music by Exploring Preview Clips in Album Cover Space. In *Proceedings of the International Conference on Mobile and Ubiquitous Multimedia*. 94–99.
[208] Arto Lehtiniemi and Jukka Holm. 2012. Using Animated Mood Pictures in Music Recommendation. In *Proceedings of the 16th International Conference on Information Visualisation*. IEEE, 143–150.
[209] Arto Lehtiniemi and Jarno Ojala. 2013. Evaluating MoodPic – A Concept for Collaborative Mood Music Playlist Creation. In *Proceedings of the International Conference on Information Visualisation*. IEEE, 86–95.
[210] Arto Lehtiniemi, Jarno Ojala, and Henri Toukomaa. 2016. Design and Evaluation of Mood Pictures in Social Music Discovery Service. *Personal and Ubiquitous Computing* 20, 1 (2016), 97–119.
[211] Arto Lehtiniemi, Jarno Ojala, and Kaisa Väänänen. 2017. Socially Augmented Music Discovery with Collaborative Playlists and Mood Pictures. *Interacting with Computers* 29, 3 (2017), 416–437.
[212] Arto Lehtiniemi and Jarno Seppänen. 2007. Evaluation of Automatic Mobile Playlist Generator. In *Proceedings of the 4th International Conference on Mobile Technology, Applications, and Systems and the 1st International Symposium on Computer Human Interaction in Mobile Technology*, 452–459.

[213] Stefan Leitich and Martin Topf. 2007. Globe of Music-Music Library Visualization Using Geosom. In *Proceedings of the International Society for Music Information Retrieval Conference*, 167–170.
[214] Eva Lenz, Sarah Diefenbach, Marc Hassenzahl, and Sébastien Lienhard. 2012. Mo. Shared Music, Shared Moment. In *Proceedings of the Nordic Conference on Human-Computer Interaction: Making Sense Through Design*, 736–741.
[215] Tuck Leong, Steve Howard, and Frank Vetere. 2008. Choice: Abdicating or Exercising? In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 715–724.
[216] Tuck W Leong, Frank Vetere, and Steve Howard. 2005. The Serendipity Shuffle. In *Proceedings of the Australian Conference on Computer-Human Interaction*, 1–4.
[217] Tuck Wah Leong, Frank Vetere, and Steve Howard. 2006. Randomness as a Resource for Design. In *Proceedings of the Conference on Designing Interactive Systems*, 132–139.
[218] James R. Lewis. 1995. IBM Computer Usability Satisfaction Questionnaires: Psychometric Evaluation and Instructions for Use. *International Journal of Human–Computer Interaction* 7, 1 (1995), 57–78.
[219] Hsin-Wei Li, Sok-Ian Sou, and Hsun-Ping Hsieh. 2020. Room-based Playlist Arrangement System using Group Recommendation. In *Proceedings of the International Computer Symposium*, 50–54.
[220] Elad Liebman, Piyush Khandelwal, Maytal Saar-Tsechansky, and Peter Stone. 2017. Designing Better Playlists with Monte Carlo Tree Search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 4715–4720.
[221] Elad Liebman, Maytal Saar-Tsechansky, and Peter Stone. 2015. DJ-MC: A Reinforcement-Learning Agent for Music Playlist Recommendation. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, 591–599.
[222] Elad Liebman, Maytal Saar-Tsechansky, and Peter Stone. 2019. The Right Music at the Right Time: Adaptive Personalized Playlists based on Sequence Modeling. *MIS Quarterly* 43, 3 (2019).
[223] Anita Shen Lillie. 2008. *MusicBox: Navigating the Space of your Music*. Ph.D. Dissertation. Massachusetts Institute of Technology.
[224] Zachary C. Lipton, John Berkowitz, and Charles Elkan. 2015. A Critical Review of Recurrent Neural Networks for Sequence Learning. arXiv:1506.00019. Retrieved from https://doi.org/10.48550/arxiv:1506.00019
[225] KuanTing Liu and Roger Andersson Reimer. 2008. Social Playlist: Enabling Touch Points and Enriching Ongoing Relationships through Collaborative Mobile Music Listening. In *Proceedings of the International Conference on Human Computer Interaction with Mobile Devices and Services*, 403–406.
[226] Beth Logan. 2002. Content-Based Playlist Generation: Exploratory Experiments. In *Proceedings of the International Society for Music Information Retrieval Conference*, Vol. 2, 295–296.
[227] Beth Logan. 2004. Music Recommendation from Song Sets. In *Proceedings of the International Society for Music Information Retrieval Conference*, 425–428.
[228] Beth Logan and Ariel Salomon. 2001. A Music Similarity Function Based on Signal Analysis. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, 190–190.
[229] Malte Ludewig, Iman Kamehkhosh, Nick Landia, and Dietmar Jannach. 2018. Effective Nearest-neighbor Music Recommendations. In *Proceedings of the ACM Recommender Systems Challenge*, 1–6.
[230] Gergely Lukács and Mátyás Jani. 2016. Analyzing Speech and Music Blocks in Radio Channels: Lessons Learned for Playlist Generation. In *Proceedings of the International Conference on Digital Information Management*. IEEE, 179–184.
[231] Valentina Maccatrozzo, Tobias Kuhn, Davide Ceolin, and Jacco Van Ossenbruggen. 2023. The Role of Serendipity in User-Curated Music Playlists. In *Proceedings of the 12th Knowledge Capture Conference*. ACM, 140–147.
[232] Maake Benard Magara, Sunday Ojo, Seleman Ngwira, and Tranos Zuva. 2016. MPlist: Context Aware Music Playlist. In *Proceedings of the IEEE International Conference on Emerging Technologies and Innovative Business Practices for the Transformation of Societies (EmergiTech)*, 309–316.
[233] François Maillet, Douglas Eck, Guillaume Desjardins, and Paul Lamere. 2009. Steerable Playlist Generation by Learning Song Similarity from Radio Station Playlists. In *Proceedings of the International Society for Music Information Retrieval Conference*, 345–350.
[234] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. 2016. Adversarial Autoencoders. arxiv:1511.05644. Retrieved from https://doi.org/10.48550/arXiv.1511.05644
[235] Judith Masthoff and Amra Delić. 2022. Group Recommender Systems: Beyond Preference Aggregation. In *Recommender Systems Handbook*. Springer, 381–420.
[236] Joseph F McCarthy and Theodore D Anagnost. 1998. MusicFX: An Arbiter of Group Preferences for Computer Supported Collaborative Workouts. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, 363–372.
[237] Brian McFee and Gert R. G. Lanckriet. 2012. Hypergraph Models of Playlist Dialects. In *Proceedings of the International Society for Music Information Retrieval Conference*, Vol. 12, 343–348.
[238] Brian McFee and Gert R. G. Lanckriet. 2011. The Natural Language of Playlists. In *Proceedings of the International Society for Music Information Retrieval Conference*, Vol. 11, 537–541.

[239] Daniel Mélo. 2020. Investigating the Role of Personalization when Creating Relaxing Playlists. In *Proceedings of the Brazilian Symposium on Multimedia and the Web*, 213–216.
[240] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems*, Vol. 26.
[241] Martijn Millecamp, Nyi Nyi Htun, Cristina Conati, and Katrien Verbert. 2019. To Explain or Not to Explain: The Effects of Personal Characteristics when Explaining Music Recommendations. In *Proceedings of the International Conference on Intelligent User Interfaces*, 397–407.
[242] Scott Miller, Paul Reimer, Steven R. Ness, and George Tzanetakis. 2010. Geoshuffle: Location-Aware, Content-based Music Browsing Using Self-organizing Tag Clouds. In *Proceedings of the International Society for Music Information Retrieval Conference*, 237–242.
[243] Jose A. Mocholi, Victor Martinez, Javier Jaen, and Alejandro Catala. 2012. A Multicriteria Ant Colony Algorithm for Generating Music Playlists. *Expert Systems with Applications* 39, 3 (2012), 2270–2278.
[244] Bart Moens, Leon van Noorden, and Marc Leman. 2010. D-Jogger: Syncing Music with Walking. In *Proceedings of the Sound and Music Computing Conference*, 451–456.
[245] Diego Monti, Enrico Palumbo, Giuseppe Rizzo, Pasquale Lisena, Raphaël Troncy, Michael Fell, Elena Cabrio, and Maurizio Morisio. 2018. An Ensemble Approach of Recurrent Neural Networks using Pre-trained Embeddings for Playlist Completion. In *Proceedings of the ACM Recommender Systems Challenge*, 1–6.
[246] Joshua L. Moore, Shuo Chen, Thorsten Joachims, and Douglas Turnbull. 2012. Learning to Embed Songs and Tags for Playlist Prediction. In *Proceedings of the International Society for Music Information Retrieval Conference*, Vol. 12, 349–354.
[247] Fabian Mörchen, Alfred Ultsch, Mario Nöcker, and Christian Stamm. 2005. Databionic Visualization of Music Collections According to Perceptual Distance. In *Proceedings of the International Society for Music Information Retrieval Conference*, 396–403.
[248] Marta Moscati, Christian Wallmann, Markus Reiter-Haas, Dominik Kowald, Elisabeth Lex, and Markus Schedl. 2023. Integrating the ACT-R Framework with Collaborative Filtering for Explainable Sequential Music Recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems*. ACM, 840–847.
[249] YV Srinivasa Murthy and Shashidhar G Koolagudi. 2018. Content-based Music Information Retrieval (CB-MIR) and Its Applications Toward the Music Industry: A Review. *ACM Computing Surveys* 51, 3 (2018), 1–46.
[250] Cataldo Musto, Marco de Gemmis, Pasquale Lops, Fedelucio Narducci, and Giovanni Semeraro. 2022. Semantics and Content-based Recommendations. In *Recommender Systems Handbook*. Springer, 251–298.
[251] Cataldo Musto, Fedelucio Narducci, Giovanni Semeraro, Pasquale Lops, and Marco de Gemmis. 2013. Distributional Models vs. Linked Data: Exploiting Crowdsourcing to Personalize Music Playlists. In *Proceedings of the 4th Italian Information Retrieval Workshop*. CEUR-WS.org, 84–87.
[252] Cataldo Musto, Giovanni Semeraro, Pasquale Lops, Marco de Gemmis, and Fedelucio Narducci. 2012. Leveraging Social Media Sources to Generate Personalized Music Playlists. In *Proceedings of E-Commerce and Web Technologies*. Springer, Berlin, 112–123.
[253] Tomoyasu Nakano, Jun Kato, Masahiro Hamasaki, and Masataka Goto. 2016. PlaylistPlayer: An Interface Using Multiple Criteria to Change the Playback Order of a Music Playlist. In *Proceedings of the 21st International Conference on Intelligent User Interfaces*. ACM, 186–190.
[254] Robert Neumayer, Michael Dittenbach, and Andreas Rauber. 2005. Playsom and Pocketsomplayer, Alternative Interfaces to Large Music Collections. In *Proceedings of the International Society for Music Information Retrieval Conference*, 618–623.
[255] Athanasios N Nikolakopoulos, Xia Ning, Christian Desrosiers, and George Karypis. 2022. Trust your Neighbors: A Comprehensive Survey of Neighborhood-based Methods for Recommender Systems. *Recommender Systems Handbook*, 39–89.
[256] Kosuke Nonaka and Satoshi Nakamura. 2021. reco.mu: A Music Recommendation System depending on Listener's Preference by Creating a Branching Playlist. In *Proceedings of the International Conference on Entertainment Computing*. Springer, 252–263.
[257] Kenton O'Hara, Matthew Lipson, Marcel Jansen, Axel Unger, Huw Jeffries, and Peter Macer. 2004. Jukola: Democratic Music Choice in a Public Space. In *Proceedings of the Conference on Designing interactive Systems: Processes, Practices, Methods, and Techniques*, 145–154.
[258] Nuria Oliver and Fernando Flores-Mangas. 2006. MPTrain: A Mobile, Music and Physiology-based Personal Trainer. In *Proceedings of the Conference on Human-Computer Interaction with Mobile Devices and Services*, 21–28.
[259] Nuria Oliver and Lucas Kreger-Stickles. 2006. PAPA: Physiology and Purpose-Aware Automatic Playlist Generation. In *Proceedings of the 7th International Society for Music Information Retrieval Conference*, Vol. 2006.
[260] Melissa Onori, Alessandro Micarelli, and Giuseppe Sansonetti. 2016. A Comparative Analysis of Personality-Based Music Recommender Systems. In *Proceedings of the Empire Workshop at the ACM Conference on Recommender Systems*, 55–59.

[261] Sergio Oramas, Vito Claudio Ostuni, Tommaso Di Noia, Xavier Serra, and Eugenio Di Sciascio. 2016. Sound and Music Recommendation with Knowledge Graphs. *ACM Transactions on Intelligent Systems and Technology* 8, 2 (2016), 1–21.

[262] Ashley M. Oudenne, Youngmoo E. Kim, and Douglas S. Turnbull. 2010. Meerkat: Exploring Semantic Music Discovery Using Personalized Radio. In *Proceedings of the International Conference on Multimedia Information Retrieval*, 429–432.

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

[264] Elias Pampalk and Martin Gasser. 2006. Dynamic Playlist Generation Based on Skipping Behavior. In *Proceedings of the 7th International Society for Music Information Retrieval Conference*, 389–390.

[265] Elias Pampalk, Tim Pohle, and Gerhard Widmer. 2005. Dynamic Playlist Generation Based on Skipping Behavior. In *Proceedings of the International Society for Music Information Retrieval Conference*, 634–637.

[266] Elias Pampalk, Andreas Rauber, and Dieter Merkl. 2002. Content-Based Organization and Visualization of Music Archives. In *Proceedings of the ACM International Conference on Multimedia*, 570–579.

[267] Piyush Papreja, Hemanth Venkateswara, and Sethuraman Panchanathan. 2020. Representation, Exploration and Recommendation of Playlists. In *Machine Learning and Knowledge Discovery in Databases*, Peggy Cellier and Kurt Driessens (Eds.). Springer International Publishing, 543–550.

[268] So Yeon Park and Blair Kaneshiro. 2021. Social Music Curation that Works: Insights from Successful Collaborative Playlists. *Proceedings of the ACM on Human-Computer Interaction* 5 (2021), 1–27.

[269] So Yeon Park and Blair Kaneshiro. 2022. User Perspectives on Critical Factors for Collaborative Playlists. *PloS One* 17, 1 (2022), e0260750.

[270] So Yeon Park, Audrey Laplante, Jin Ha Lee, and Blair Kaneshiro. 2019. Tunes Together: Perception and Experience of Collaborative Playlists. In *Proceedings of the International Society for Music Information Retrieval Conference*, 723–730.

[271] So Yeon Park and Sang Won Lee. 2021. Lost in Co-curation: Uncomfortable Interactions and the Role of Communication in Collaborative Music Playlists. *Proceedings of the ACM Conference on Human-Computer Interaction* 5 (2021), 1–24.

[272] So Yeon Park, Emily Redmond, Jonathan Berger, and Blair Kaneshiro. 2022. Hitting Pause: How User Perceptions of Collaborative Playlists Evolved in the United States During the COVID-19 Pandemic. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 1–16.

[273] Ayush Patwari, Nicholas Kong, Jun Wang, Ullas Gargi, Michele Covell, and Aren Jansen. 2020. Semantically Meaningful Attributes from Co-Listen Embeddings for Playlist Exploration and Expansion. In *Proceedings of the International Society for Music Information Retrieval Conference*, 527–533.

[274] Steffen Pauws and Berry Eggen. 2002. PATS: Realization and User Evaluation of an Automatic Playlist Generator. In *Proceedings of the International Society for Music Information Retrieval Conference*, Vol. 2, 222–230.

[275] Steffen Pauws and Sander van de Wijdeven. 2005. User Evaluation of a New Interactive Playlist Generation Concept. In *Proceedings of the International Society for Music Information Retrieval Conference*, 638–643.

[276] Steffen Pauws, Wim Verhaegh, and Mark Vossen. 2006. Fast Generation of Optimal Music Playlists using Local Search. In *Proceedings of the International Society for Music Information Retrieval Conference*, 138–143.

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

[278] Geoffroy Peeters and Joachim Flocon-Cholet. 2012. Perceptual Tempo Estimation Using GMM-regression. In *Proceedings of the Second International ACM Workshop on Music Information Retrieval With User-Centered and Multimodal Strategies*, 45–50.

[279] Geoffroy Peeters, Bruno L. Giordano, Patrick Susini, Nicolas Misdariis, and Stephen McAdams. 2011. The Timbre Toolbox: Extracting Audio Descriptors from Musical Signals. *The Journal of the Acoustical Society of America* 130, 5 (2011), 2902–2916.

[280] Martin Pichl, Eva Zangerle, and Günther Specht. 2015. Towards a Context-Aware Music Recommendation Approach: What Is Hidden in the Playlist Name?. In *Proceedings of the 2015 IEEE International Conference on Data Mining Workshop (ICDMW)*, 1360–1365.

[281] Martin Pichl, Eva Zangerle, and Günther Specht. 2016. Understanding Playlist Creation on Music Streaming Platforms. In *Proceedings of the IEEE International Symposium on Multimedia*. IEEE, 475–480.

[282] Martin Pichl, Eva Zangerle, and Günther Specht. 2017. Understanding User-Curated Playlists on Spotify: A Machine Learning Approach. *International Journal of Multimedia Data Engineering and Management* 8, 4 (2017), 44–59.

[283] John Platt, Christopher J. Burges, Steven Swenson, Christopher Weare, and Alice Zheng. 2001. Learning a Gaussian Process Prior for Automatically Generating Music Playlists. In *Advances in Neural Information Processing Systems*, Vol. 14.

[284] Tim Pohle, Peter Knees, Markus Schedl, Elias Pampalk, and Gerhard Widmer. 2007. "Reinventing the wheel": A Novel Approach to Music Player Interfaces. *IEEE Transactions on Multimedia* 9, 3 (2007), 567–575.

[285] Tim Pohle, Elias Pampalk, and Gerhard Widmer. 2005. Generating Similarity-Based Playlists using Traveling Salesman Algorithms. In *Proceedings of the International Conference on Digital Audio Effects*, 220–225.

[286] Marco Polignano, Pierpaolo Basile, Marco de Gemmis, and Giovanni Semeraro. 2019. Social Tags and Emotions as Main Features for the Next Song to Play in Automatic Playlist Continuation. In *Adjunct Proceedings of the Conference on User Modeling, Adaptation and Personalization*, 235–239.

[287] Jordi Pons, Oriol Nieto, Matthew Prockup, Erik Schmidt, Andreas Ehmann, and Xavier Serra. 2018. End-To-End Learning for Music Audio Tagging at Scale. In *Proceedings of the International Society for Music Information Retrieval Conference*.

[288] Luciana Fujii Pontello, Pedro H. F. Holanda, Bruno Guilherme, João Paulo V. Cardoso, Olga Goussevskaia, and Ana Paula Couto Da Silva. 2017. Mixtape: Using Real-time User Feedback to Navigate Large Media Collections. *ACM Transactions on Multimedia Computing, Communications, and Applications* 13, 4 (2017), 1–22.

[289] George Popescu and Pearl Pu. 2012. What's the Best Music You Have? Designing Music Recommendation for Group Enjoyment in GroupFun. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, 1673–1678.

[290] Lorenzo Porcaro and Emilia Gómez Gutiérrez. 2019. 20 Years of Playlists: A Statistical Analysis on Popularity and Diversity. In *Proceedings of the International Society for Music Information Retrieval Conference,* 130–136.

[291] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. 2018. Sequence-Aware Recommender Systems. *ACM Computing Surveys* 51, 4 (2018).

[292] Robert Ragno, Christopher J. C. Burges, and Cormac Herley. 2005. Inferring Similarity between Music Objects with Application to Playlist Generation. In *Proceedings of the ACM SIGMM International Workshop on Multimedia Information Retrieval*, 73–80.

[293] Sasank Reddy and Jeff Mascia. 2006. Lifetrak: Music in Tune with Your Life. In *Proceedings of the ACM International Workshop on Human-centered Multimedia*, 25–34.

[294] Gordon Reynolds, Dan Barry, Ted Burke, and Eugene Coyle. 2007. Towards a Personal Automatic Music Playlist Generation Algorithm: The Need for Contextual Information. In *Proceedings of the International Audio Mostly Conference*. 84–89.

[295] Gordon Reynolds, Dan Barry, Ted Burke, and Eugene Coyle. 2008. Interacting with Large Music Collections: Towards the Use of Environmental Metadata. In *Proceedings of the IEEE International Conference on Multimedia and Expo.* IEEE, 989–992.

[296] Francesco Ricci, Lior Rokach, and Bracha Shapira. 2022. Recommender Systems: Techniques, Applications, and Challenges. In *Recommender Systems Handbook*. Springer, 1–35.

[297] Richard A. Roberts and Clifford T. Mullis. 1987. *Digital Signal Processing*. Addison-Wesley.

[298] Vasiliy Rubtsov, Mikhail Kamenshchikov, Ilya Valyaev, Vasiliy Leksin, and Dmitry I Ignatov. 2018. A Hybrid Two-Stage Recommender System for Automatic Playlist Continuation. In *Proceedings of the ACM Recommender Systems Challenge*, 1–4.

[299] Keigo Sakurai, Ren Togo, Takahiro Ogawa, and Miki Haseyama. 2020. Music Playlist Generation Based on Reinforcement Learning Using Acoustic Feature Map. In *Proceedings of the Global Conference on Consumer Electronics.* IEEE, 942–943.

[300] Keigo Sakurai, Ren Togo, Takahiro Ogawa, and Miki Haseyama. 2021. Music Playlist Generation based on Graph Exploration Using Reinforcement Learning. In *Proceedings of the Global Conference on Life Sciences and Technologies.* IEEE, 53–54.

[301] Keigo Sakurai, Ren Togo, Takahiro Ogawa, and Miki Haseyama. 2022. Controllable Music Playlist Generation Based on Knowledge Graph and Reinforcement Learning. *Sensors* 22, 10 (2022), 3722.

[302] Vegard Sandvold, Thomas Aussenac, Oscar Celma, and Perfecto Herrera. 2006. Good Vibrations: Music Discovery through Personal Musical Concepts. In *Proceedings of the International Society for Music Information Retrieval Conference*, 322–323.

[303] Andy M. Sarroff and Michael Casey. 2012. Modeling and Predicting Song Adjacencies in Commercial Albums. In *Proceedings of the Sound and Music Computing Conference,* 364–371.

[304] Markus Schedl, Georg Breitschopf, and Bogdan Ionescu. 2014. Mobile Music Genius: Reggae at the Beach, Metal on a Friday Night? In *Proceedings of International Conference on Multimedia Retrieval*, 507–510.

[305] Markus Schedl, Peter Knees, Brian McFee, and Dmitry Bogdanov. 2022. Music Recommendation Systems: Techniques, Use Cases, and Challenges. In *Recommender Systems Handbook*. Springer, 927–971.

[306] Markus Schedl, Hamed Zamani, Ching-Wei Chen, Yashar Deldjoo, and Mehdi Elahi. 2018. Current Challenges and Visions in Music Recommender Systems Research. *International Journal of Multimedia Information Retrieval* 7, 2 (2018), 95–116.

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

[308] Ingo Schmädecke and Holger Blume. 2013. High Performance Hardware Architectures for Automated Music Classification. In *Algorithms from and for Nature and Life*. Springer, 539–547.
[309] Dominik Schnitzer, Tim Pohle, Peter Knees, and Gerhard Widmer. 2007. One-Touch Access to Music on Mobile Devices. In *Proceedings of the International Conference on Mobile and Ubiquitous Multimedia*, 103–109.
[310] Harald Victor Schweiger, Emilia Parada-Cabaleiro, and Markus Schedl. 2021. Does Track Sequence in User-Generated Playlists Matter? In *Proceedings of the International Society for Music Information Retrieval Conference*, 618–625.
[311] Pavan Seshadri and Peter Knees. 2023. Leveraging Negative Signals with Self-Attention for Sequential Music Recommendation. In *Proceedings of MuRS: Music Recommender Systems Workshop*.
[312] Shun-Yao Shih and Heng-Yu Chi. 2018. Automatic, Personalized, and Flexible Playlist Generation Using Reinforcement Learning. In *Proceedings of the International Society for Music Information Retrieval Conference,* 168–174.
[313] Ben Shneiderman. 1992. Tree Visualization with Tree-maps: 2-d Space-Filling Approach. *ACM Transactions on Graphics* 11, 1 (1992), 92–99.
[314] Malcolm Slaney and William White. 2006. Measuring Playlist Diversity for Recommendation Systems. In *Proceedings of the ACM Workshop on Audio and Music Computing Multimedia*, 77–82.
[315] Malcolm Slaney and William White. 2007. Similarity Based on Rating Data. In *Proceedings of the International Society for Music Information Retrieval Conference*, 479–484.
[316] Dionisios N. Sotiropoulos, Aristomenis S. Lampropoulos, and George A. Tsihrintzis. 2007. Evaluation of Modeling Music Similarity Perception via Feature Subset Selection. In *International Conference on User Modeling*. Springer, 288–297.
[317] Louis Spinelli, Josephine Lau, Liz Pritchard, and Jin Ha Lee. 2018. Influences on the Social Practices Surrounding Commercial Music Services: A Model for Rich Interactions. In *Proceedings of the International Society for Music Information Retrieval Conference*, 671–677.
[318] Spotify. 2024. Explore Spotify Catalogue. Retrieved from https://explore.spotify.com/uk/pages/Mobile-feature-discover-catalog
[319] David Sprague, Fuqu Wu, and Melanie Tory. 2008. Music Selection Using the Partyvote Democratic Jukebox. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, 433–436.
[320] Harald Steck, Linas Baltrunas, Ehtsham Elahi, Dawen Liang, Yves Raimond, and Justin Basilico. 2021. Deep Learning for Recommender Systems: A Netflix Case Study. *AI Magazine* 42 (2021), 7–18.
[321] Richard Stenzel and Thomas Kamps. 2005. Improving Content-Based Similarity Measures by Training a Collaborative Model. In *Proceedings of the International Society for Music Information Retrieval Conference*, 264–271.
[322] Rebecca Stewart and Mark Sandler. 2011. An Auditory Display in Playlist Generation. *IEEE Signal Processing Magazine* 28, 4 (2011), 14–23.
[323] Simone Stumpf and Sam Muscroft. 2011. When Users Generate Music Playlists: When Words Leave Off, Music Begins? In *Proceedings of the IEEE International Conference on Multimedia and Expo.* IEEE, 1–6.
[324] Ganeshsiva Subramaniam, Janhavi Verma, Nikhil Chandrasekhar, K. C. Narendra, and Koshy George. 2018. Generating Playlists on the Basis of Emotion. In *IEEE Symposium Series on Computational Intelligence.* IEEE, 366–373.
[325] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction.* MIT Press.
[326] Irene Teinemaa, Niek Tax, and Carlos Bentes. 2018. Automatic Playlist Continuation through a Composition of Collaborative Filters. arXiv:1808.04288. Retrieved from https://doi.org/10.48550/arxiv:1808.04288
[327] Nava Tintarev, Christoph Lofi, and C. S. Liem. 2017. Sequences of Diverse Song Recommendations. *User Modelling, Adaptation and Personalization,* 391–392.
[328] Federico Tomasi, Joseph Cauteruccio, Surya Kanoria, Kamil Ciosek, Matteo Rinaldi, and Zhenwen Dai. 2023. Automatic Music Playlist Generation via Simulation-Based Reinforcement Learning. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining.* ACM, 4948–4957.
[329] Marc Torrens, Patrick Hertzog, and Josep Lluis Arcos. 2004. Visualizing and Exploring Personal Music Libraries. In *Proceedings of the International Society for Music Information Retrieval Conference.*
[330] Thanh Tran, Renee Sweeney, and Kyumin Lee. 2019. Adversarial Mahalanobis Distance-Based Attentive Song Recommender for Automatic Playlist Continuation. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, 245–254.
[331] Roberto Turrin, Andrea Condorelli, Paolo Cremonesi, Roberto Pagano, and Massimo Quadrana. 2015. Large Scale Music Recommendation. In *Proceedings of the Workshop on Large-Scale Recommender Systems at the ACM Recommender Systems Conference*.
[332] Roberto Turrin, Massimo Quadrana, Andrea Condorelli, Roberto Pagano, and Paolo Cremonesi. 2015. 30Music Listening and Playlists Dataset. In *Poster Proceedings of the 9th ACM Conference on Recommender Systems.* CEUR-WS.org.

[333] Seiji Ueda, Atsushi Keyaki, and Jun Miyazaki. 2018. A Contextual Random Walk Model for Automated Playlist Generation. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*. IEEE, 367–374.
[334] Tomomi Uota and Takayuki Itoh. 2014. GRAPE: A Gradation Based Portable Visual Playlist. In *Proceedings of the 2014 18th International Conference on Information Visualisation*, 361–365.
[335] Iacopo Vagliano, Lukas Galke, Florian Mai, and Ansgar Scherp. 2018. Using Adversarial Autoencoders for Multi-Modal Automatic Playlist Continuation. In *Proceedings of the ACM Recommender Systems Challenge*, 1–6.
[336] Andreu Vall, Matthias Dorfer, Hamid Eghbal-Zadeh, Markus Schedl, Keki Burjorjee, and Gerhard Widmer. 2019. Feature-combination Hybrid Recommender Systems for Automated Music Playlist Continuation. *User Modeling and User-Adapted Interaction* 29, 2 (2019), 527–572.
[337] Andreu Vall, Matthias Dorfer, Markus Schedl, and Gerhard Widmer. 2018. A Hybrid Approach to Music Playlist Continuation Based on Playlist-Song Membership. In *Proceedings of the Annual ACM Symposium on Applied Computing*, 1374–1382.
[338] Andreu Vall, Hamid Eghbal-zadeh, Matthias Dorfer, and Markus Schedl. 2016. Timbral and Semantic Features for Music Playlists. In *Proceedings of the Machine Learning for Music Discovery Workshop at the Internation Conference on Machine Learning*.
[339] Andreu Vall, Hamid Eghbal-Zadeh, Matthias Dorfer, Markus Schedl, and Gerhard Widmer. 2017. Music Playlist Continuation by Learning from Hand-curated Examples and Song Features: Alleviating the Cold-start Problem for Rare and Out-of-set Songs. In *Proceedings of the Workshop on Deep Learning for Recommender Systems*, 46–54.
[340] Andreu Vall, Massimo Quadrana, Markus Schedl, and Gerhard Widmer. 2018. The Importance of Song Context and Song Order in Automated Music Playlist Generation. arXiv:1807.04690. Retrieved from https://doi.org/10.48550/arxiv:1807.04690
[341] Andreu Vall, Massimo Quadrana, Markus Schedl, and Gerhard Widmer. 2019. Order, Context and Popularity Bias in Next-song Recommendations. *International Journal of Multimedia Information Retrieval* 8, 2 (2019), 101–113.
[342] Andreu Vall, Massimo Quadrana, Markus Schedl, Gerhard Widmer, and Paolo Cremonesi. 2017. The Importance of Song Context in Music Playlists. In *Proceedings of the ACM Conference on Recommender Systems*.
[343] Rob Van Gulik and Fabio Vignoli. 2005. Visual Playlist Generation on the Artist Map. In *Proceedings of the International Society for Music Information Retrieval Conference*, Vol. 5, 520–523.
[344] Rob van Gulik, Fabio Vignoli, and Huub van de Wetering. 2004. Mapping Music in the Palm of Your Hand, Explore and Discover Your Collection. In *Proceedings of the International Society for Music Information Retrieval Conference*.
[345] Timo van Niedek and Arjen P. de Vries. 2018. Random Walk with Restart for Automatic Playlist Continuation and Query-specific Adaptations. In *Proceedings of the ACM Recommender Systems Challenge*, 1–6.
[346] Flavian Vasile, Elena Smirnova, and Alexis Conneau. 2016. Meta-prod2vec: Product Embeddings using Side-information for Recommendation. In *Proceedings of the ACM Conference on Recommender Systems*, 225–232.
[347] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Advances in Neural Information Processing Systems*, Vol. 30, 6000–6010.
[348] Felipe Vieira and Nazareno Andrade. 2015. Evaluating Conflict Management Mechanisms for Online Social Jukeboxes. In *Proceedings of the International Society for Music Information Retrieval Conference*, 190–196.
[349] Fabio Vignoli and Steffen Pauws. 2005. A Music Retrieval System Based on User Driven Similarity and Its Evaluation. In *Proceedings of the International Society for Music Information Retrieval Conference*, 272–279.
[350] Brett Vintch. 2020. Quick Lists: Enriched Playlist Embeddings for Future Playlist Recommendation. In *Proceedings of the Future Technologies Conference*. Springer, 583–593.
[351] Maksims Volkovs, Himanshu Rai, Zhaoyue Cheng, Ga Wu, Yichao Lu, and Scott Sanner. 2018. Two-Stage Model for Automatic Playlist Continuation at Scale. In *Proceedings of the ACM Recommender Systems Challenge*, 1–6.
[352] Michael Voong. 2006. *Generating Music playlists Using Colour*. Bachelor's thesis. University of Birmingham.
[353] Guan-Hua Wang, Chia-Hao Chung, Yian Chen, and Homer Chen. 2018. Multi-Label Playlist Classification Using Convolutional Neural Network. In *Proceedings of the 2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 1957–1962.
[354] Shoujin Wang, Longbing Cao, Yan Wang, Quan Z. Sheng, Mehmet A. Orgun, and Defu Lian. 2021. A Survey on Session-based Recommender Systems. *ACM Computing Surveys* 54, 7 (2021), 154:1–154:38.
[355] Morgan Ward, Joseph Goodman, and Julie Irwin. 2006. I Want It Even Though I Do Not Like It: Preference for Familiar But Less Liked Music. *Advances in Consumer Research* 33 (2006).
[356] Morgan K. Ward, Joseph K. Goodman, and Julie R. Irwin. 2014. The Same Old Song: The Power of Familiarity in Music Choice. *Marketing Letters* 25, 1 (2014), 1–11.
[357] Christopher John Cornish Hellaby Watkins. 1989. *Learning from Delayed Rewards*. Ph.D. Dissertation. University of Cambridge.

[358] Jack Webster. 2020. Taste in the Platform Age: Music Streaming Services and New Forms of Class Distinction. *Information, Communication & Society* 23, 13 (2020), 1909–1924.
[359] Minz Won, Sanghyuk Chun, Oriol Nieto, and Xavier Serra. 2020. Data-Driven Harmonic Filters for Audio Representation Learning. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, 536–540.
[360] Linxing Xiao, Lie Lu, Frank Seide, and Jie Zhou. 2009. Learning a Music Similarity Measure on Automatic Annotations with Application to Playlist Generation. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 1885–1888.
[361] Hojin Yang, Yoonki Jeong, Minjin Choi, and Jongwuk Lee. 2018. MMCF: Multimodal Collaborative Filtering for Automatic Playlist Continuation. In *Proceedings of the ACM Recommender Systems Challenge*, 1–6.
[362] Ali Yürekli, Alper Bilge, and Cihan Kaleli. 2021. Exploring Playlist Titles for Cold-start Music Recommendation: An Effectiveness Analysis. *Journal of Ambient Intelligence and Humanized Computing* 12, 11 (2021), 10125–10144.
[363] Ali Yürekli, Cihan Kaleli, and Alper Bilge. 2021. Alleviating the Cold-Start Playlist Continuation in Music Recommendation using Latent Semantic Indexing. *International Journal of Multimedia Information Retrieval* 10, 3 (2021), 185–198.
[364] Hamed Zamani, Markus Schedl, Paul Lamere, and Ching-Wei Chen. 2019. An Analysis of Approaches Taken in the ACM RecSys Challenge 2018 for Automatic Music Playlist Continuation. *ACM Transactions on Intelligent Systems and Technology* 10, 5 (2019), 1–21.
[365] Eva Zangerle, Michael Tschuggnall, Stefan Wurzinger, and Günther Specht. 2018. ALF-200k: Towards Extensive Multimodal Analyses of Music Tracks and Playlists. In *Proceedings of the 39th European Conference on IR Research*. Springer, 584–590.
[366] Fan Zhang, Hongying Meng, and Maozhen Li. 2016. Emotion Extraction and Recognition from Music. In *Proceedings of the International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery*. 1728–1733.
[367] Zhiqiang Zhang, Changshui Zhang, and Shifeng Weng. 2013. Music Playlist Prediction via Detecting Song Moods. In *Proceedings of the 2013 IEEE China Summit and International Conference on Signal and Information Processing*, 174–178.
[368] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A Survey of Large Language Models. arXiv:2303.18223. Retrived from https://doi.org/10.48550/arxiv:2303.18223
[369] Xing Zhao, Qingquan Song, James Caverlee, and Xia Hu. 2018. Trailmix: An Ensemble Recommender System for Playlist Curation and Continuation. In *Proceedings of the ACM Recommender Systems Challenge*, 1–6.
[370] Elena Zheleva, John Guiver, Eduarda Mendes Rodrigues, and Nataša Milić-Frayling. 2010. Statistical Models of Music-Listening Sessions in Social Media. In *Proceedings of the International Conference on World Wide Web*, 1019–1028.
[371] Fang Zheng, Guoliang Zhang, and Zhanjiang Song. 2001. Comparison of Different Implementations of MFCC. *Journal of Computer Science and Technology* 16, 6 (2001), 582–589.
[372] Lin Zhu, Bowen He, Mengxin Ji, Cheng Ju, and Yihong Chen. 2018. Automatic Music Playlist Continuation via Neighbor-Based Collaborative Filtering and Discriminative Reweighting/Reranking. In *Proceedings of the ACM Recommender Systems Challenge*, 1–6.
[373] Yongwei Zhu and Mohan S. Kankanhalli. 2006. Precise Pitch Profile Feature Extraction from Musical Audio for Key Detection. *IEEE Transactions on Multimedia* 8, 3 (2006), 575–584.

Received 20 October 2023; revised 21 May 2024; accepted 23 July 2024