Intelligibility of Music Playlists

Giovanni Gabbolini
MSC
119227536

**Thesis submitted for the degree of
Doctor of Philosophy**

NATIONAL UNIVERSITY OF IRELAND, CORK

COLLEGE OF SCIENCE, ENGINEERING AND FOOD SCIENCE

SCHOOL OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY
INSIGHT CENTRE FOR DATA ANALYTICS

March 2023

Head of School:	Professor Utz Roedig
Supervisor:	Dr. Derek G. Bridge

Contents

List of Figures

List of Tables

I, Giovanni Gabbolini, certify that this thesis is my own work and has not been submitted for another degree at University College Cork or elsewhere.

Giovanni Gabbolini

A Mamma e Babbo,
per essere continue ispirazioni.

Acknowledgements

I would like to thank the people that made my PhD journey special.

Thanks to my family: Rossella; Franco; Chiara; and Anna. My family is a safe space, a place to be during joy, sorrow, or whenever I want to. I'm extremely grateful to have them.

Thanks to my supervisor, Derek. Derek has many qualities: he is clear-thinking, knowledgeable and professional, which make him one of the top-researchers in our community. However, differently from many other top-researchers, Derek has an additional, special quality: he is a passionate teacher. In the current research world, Derek is an example in his passion for science and people.

Thanks to my friends in the Sad-Sepolcro group for supporting me in everything I do, everywhere I go, and in everything I feel.

Thanks to my colleagues and friends in University College Cork for our time together, both inside and outside the office. A special thanks to Vedat for the nights out, the fun, and for being a dear friend. Also, thanks to Insight staff Caitríona, Eleanor, Linda and Peter for their support and kindness.

Thanks to the Insight Centre for Data Analytics for financing my research, including my travels to several conferences around the globe.

Thanks to my housemates, Buvana, Marco and Roberta, for creating a nice environment in which to live a serene everyday life.

I'm grateful for the internship I did. Thanks to University College Cork for supporting the internship. Thanks to Deezer for hosting me, to Elena Epure and Romain Hennequin for arranging the internship, and for their supervision, and to the whole research team for being amazing colleagues. A special thanks to Bruno, for making my time in Paris so special.

Thanks to everybody I met during these three years, for your contribution to my professional and personal growth.

Derek and I thank Peter Knees of TU Wien for useful discussions about the design of the user trial of Chapter 3. We also thank Elena Epure & Romain Hennequin from Deezer, for sharing the dataset we used in the experiments of Chapter 6, and for assisting us in the process of replicating the baselines of Chapter 6. We also thank Jeong Choi from NAVER, for assisting us in the process of replicating the baselines of Chapter 6.

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.

Giovanni Gabbolini
Cork, March 2023

Abstract

A common strategy for organising music is by arranging songs in a playlist to obtain a continuous and thematic music flow. Playlists are popular in music streaming services, where 58% of the listeners construct their own playlists. The flip side of popularity is content-overload; streaming services currently host billions of playlists. The commercial value of playlists has attracted notable research efforts during the last two decades. Much of the research on playlists is concerned with automatically constructing playlists. This dissertation is on playlists, but on a topic complementary to constructing playlists. Our concern here is on describing playlists, so that playlists can be understood by a human audience, i.e. so that they become *intelligible*.

The way we achieve intelligibility is by developing algorithms that can generate textual annotations, both at playlist level and at song level. At playlist level, an annotation can be text (e.g. a tag or a caption) that describes the playlist as a whole; at song level, an annotation can be text that describes the transition between two consecutive songs in the playlist. The purpose of intelligibility is that of facilitating music organisation & access, as well as enhancing the listening experience of users, two goals particularly relevant in a content overload scenario.

We propose five algorithms for playlist-level intelligibility, and three algorithms for song-level intelligibility. We are particularly interested in the user experience, so we test the algorithms, in most cases, with both offline experiments and user trials. We find evidence that the algorithms can help accomplish the two goals of intelligibility, i.e. enhancing listening experiences, and facilitating organisation and access.

We pair the algorithms with a comprehensive survey of MIR research on music playlists, which provide a useful framework for understanding our contributions in the context of a broad selection of related research.

Chapter 1

Introduction

1.1 Background and context

The technological revolutions of the late 20^{th} 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. According to the global music report of the International Federation of the Phonographic Industry (IFPI)[2], streaming accounted for 16.9 billion dollars of revenue in 2021, which is 68% of the total global recorded music industry revenues for that year. In the context of this dissertation, we consider music streaming services as the default medium to interact with recorded music. And, we refer to recorded music simply as “music”.

In exchange for a monthly subscription fee of around ten euros, or for free in exchange for exposure to advertisements, music streaming services allow their subscribers to access an enormous catalogue of music, from different devices, such as smartphones and personal computers, at any time. In the context of this dissertation, we refer to a subscriber of a music streaming service as a “user”.

The abundance of available music raises the risk that users of streaming services will be overwhelmed [Hag15]. For example, the music streaming services Spotify[3] and Deezer[4] have catalogue sizes of respectively 50 and 90 million songs.

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

[2]`https://globalmusicreport.ifpi.org`.

[3]`https://spotify.com`.

[4]`https://deezer.com`.

$$\begin{tikzcd}
\boxed{\textit{Interstellar Love} \text{ by } \textit{Thundercat}} \arrow[d] \\
\boxed{\textit{Post Requisite} \text{ by } \textit{Flying Lotus}} \arrow[d] \\
\boxed{\textit{Wisdom Eye} \text{ by } \textit{Alice Coltrane}}
\end{tikzcd}$$

Figure 1.1: A playlist of three songs.

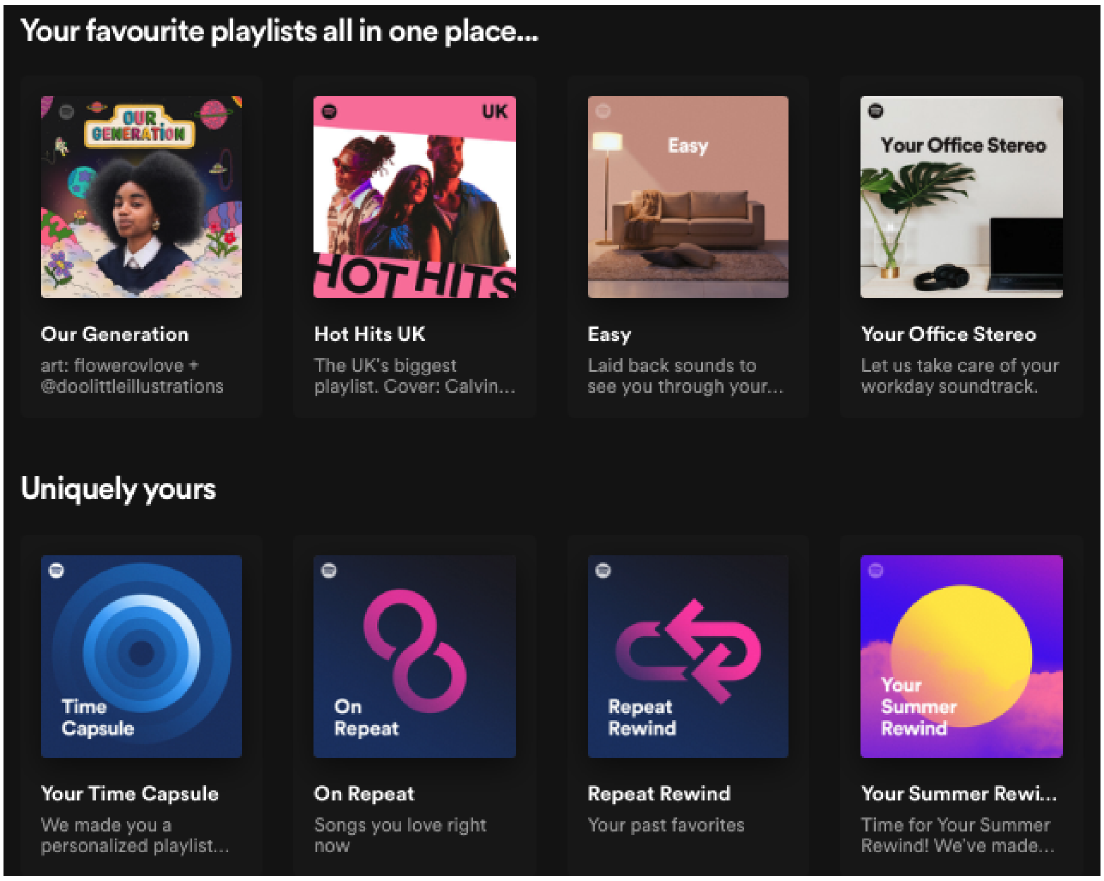

Your favourite playlists all in one place...
OUR GENERATION
UK
HOT HITS
Easy
Your Office Stereo
Our Generation
art: flowerovlove + @doolittleillustrations
Hot Hits UK
The UK's biggest playlist. Cover: Calvin...
Easy
Laid back sounds to see you through your...
Your Office Stereo
Let us take care of your workday soundtrack.
Uniquely yours
Time Capsule
On Repeat
Repeat Rewind
Your Summer Rewind
Your Time Capsule
We made you a personalized playlist...
On Repeat
Songs you love right now
Repeat Rewind
Your past favorites
Your Summer Rewi...
Time for Your Summer Rewind! We've made...

Figure 1.2: The home page of the music streaming service Spotify features playlists as a personalised and prominent element. Picture taken June 15th 2022.

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" [SZC+18], have become one of the preferred ways of accessing music. An example of a three-song playlist is in Figure 1.1. Listeners use playlists to structure their listening, efficiently accessing the right music at the right time [Fly16].

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 1.2. Notice that the playlists of Figure 1.2 are personalised, 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%) [SZC$^+$18].

Playlists are created for users by professional curators and algorithms, and by users for themselves and other users, for convenience and self-expression [Web20]. A study conducted in 2017 reveled that 58% of users in the USA create their own playlists, and that 32% of users share their playlists with other users [SZC$^+$18]. In total, the music streaming service Spotify was hosting more than four billion playlists in 2021[5].

The commercial value of playlists has attracted notable research efforts during the last two decades.

Much of the research on playlists is concerned with automatic generation of playlists, e.g. [FSGW08, VGV05, VRC$^+$18, GVJSM19]. Other work looks into how humans manually construct playlists, e.g. see [PZS16, Hag15]. We offer a complete review of the research on playlists in Chapter 2.

This dissertation is on playlists, but focuses on a complementary topic to the ones described in the previous paragraph. Our focus is on *intelligibility*, that is the degree to which playlists can be understood by a human audience. Given the sequential nature of playlists, we can distinguish two levels of intelligibility: song-level and playlist-level. Song-level intelligibility is the degree to which transitions between consecutive songs can be understood by a human audience. Playlist-level intelligibility is the degree to which the characteristics of a playlist can be understood by a human audience.

The way we achieve intelligibility in this dissertation is by developing algorithms that can generate textual annotations, both at song-level and playlist-level. In particular, we achieve song-level intelligibility by generating short song-to-song textual annotations, or *segues*. For example, given the playlist of Figure 1.1, a segue between the first two songs is: "*Interstellar Love* was produced by *Flying Lotus*, and *Flying Lotus* is the artist who made the next song". We achieve playlist-level intelligibility by generating playlist tags and captions. For example, the playlist of Figure 1.1 can be tagged as "jazz" & "alternative", and can be captioned as "Derivations of jazz, from the 70s to our time". Playlist tagging is the task of assigning to a playlist one or more tags, drawn from a

[5]`https://backlinko.com/spotify-users`.

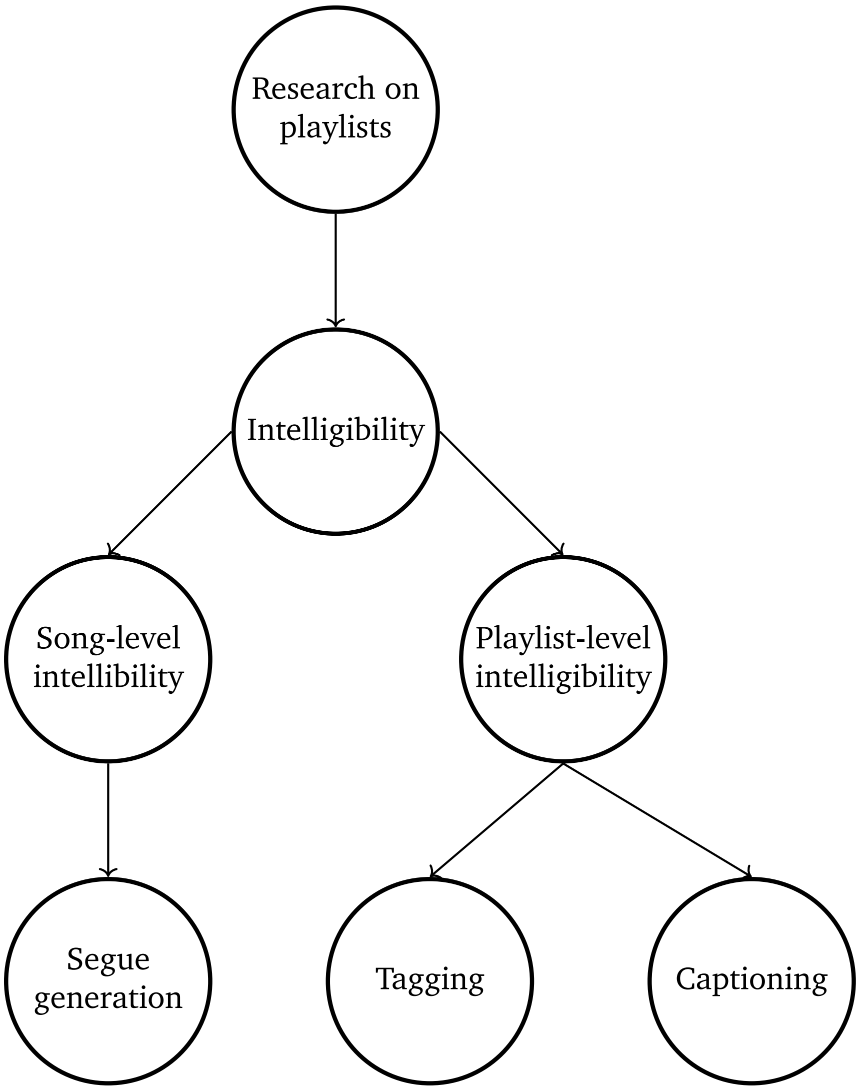


Figure 1.3: Hierarchical organisation of the research topics we tackle in this dissertation.

fixed vocabulary of tags. Playlist captioning is the task of describing a playlist using natural language [CFM+16]. In summary, the research topics we tackle can be organised hierarchically as in Figure 1.3.

1.2 Motivation

The research on intelligibility we present in this dissertation has two goals:

1. Enhance the playlist listening experience;
2. Facilitate playlist organisation and access.

In the following, we review these two goals separately, explaining why it is

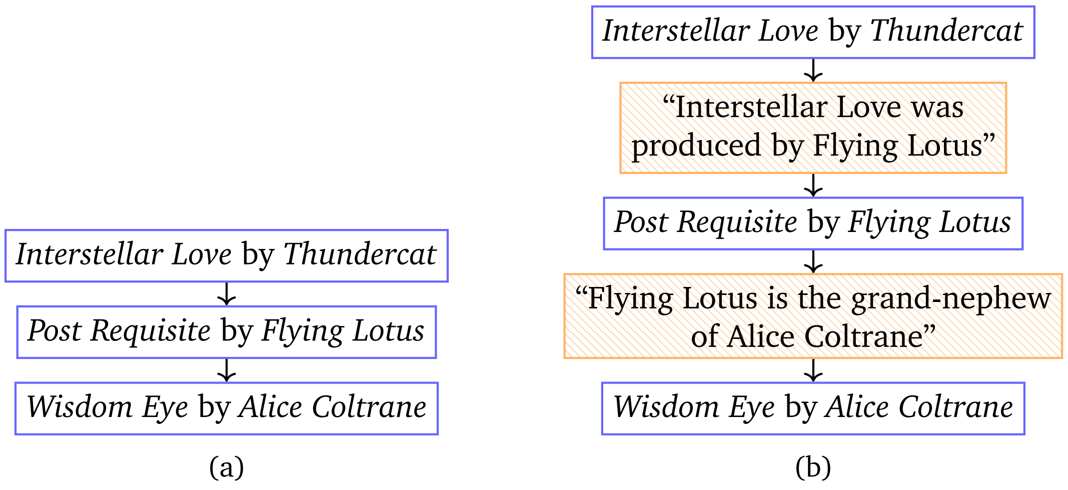

Interstellar Love by *Thundercat*
Post Requisite by *Flying Lotus*
Wisdom Eye by *Alice Coltrane*
(a)
Interstellar Love by *Thundercat*
“Interstellar Love was produced by Flying Lotus”
Post Requisite by *Flying Lotus*
“Flying Lotus is the grand-nephew of Alice Coltrane”
Wisdom Eye by *Alice Coltrane*
(b)

Figure 1.4: A playlist of three songs (a), and a tour of the same three songs (b).

important to attain them, and linking them to the two research topics of song- and playlist-level intelligibility.

1.2.1 Enhance the playlist listening experience

The literature on the psychology of music provides evidence that information-seeking is one of several motivations for why we, as people, listen to music. For example, [BH13] find that people sometimes listen to music to seek information about the music content itself; [LN11] find that people may listen to music in order to “learn about things”; and [LD04] find that a majority of listeners are likely, on occasion, to search for information such as lyrics and artist information when listening to music.

In a study comparing broadcast radio and streaming services as music listening mediums, [COWH20] find that the two mediums are complementary: while listening to music, some people switch from streaming services to radio when in need of information. Streaming services have adapted to the need that their users have for information when listening to single songs, e.g. by presenting the lyrics of the songs and stories associated with the songs [BMT+19]. However, how to address the information-seeking needs of users when listening to playlists is an under-researched topic. In fact, the way playlists are presented to date is list-wise: one song follows the other, from the first to the last. Presented in this way, no interaction with users is possible when going from one song to another.

We defined song-level intelligibility as the degree to which users can understand transitions between consecutive songs in playlists. If these transitions are intelligible, then this is one way of addressing the information-seeking needs of users while listening to playlists. In this dissertation, we achieve song-level intelligibility by generating segues, defined in Section 1.1 as short song-to-song textual connections. For example, given the playlist of Figure 1.1, a segue between the first two songs is: "*Interstellar Love* was produced by *Flying Lotus*, and *Flying Lotus* is the artist who made the next song.".

We present users with what we call *tours*, i.e. playlists where songs alternate with segues. For example, Figure 1.4 (a) is a playlist of three songs and Figure 1.4 (b) is a tour of the same three songs. Tours resemble the popular format of radio shows, where a presenter speaks and takes the audience from one song to another. As such, tours address the information-seeking need of users. However, notice that tours are different from radio shows, because the segues in tours are confined to explaining the connection between the two songs, limiting the topic of interaction. Some radio shows, instead, would also interact with listeners about topics not related to the songs, such as updates on traffic or world news. Nevertheless, the work of [BMT+19] hints that tours offer a better user experience than playlists, given the right listening context.

1.2.2 Facilitate playlist organisation and access

Music organisation and access are two of the challenges faced within Music Information Retrieval (MIR)[6], an active research field with a two-decades long history. In particular, playlist organisation is the process of arranging playlists systematically, while playlist access is the process of retrieving a particular playlist. Content overload is the main motivation for researching playlist organisation and access, as users of music streaming services need tools to find their way through the enormous abundance of available playlists. For instance, the popular music streaming service Spotify, was hosting more than four billion playlists in 2021[7].

Playlist-level intelligibility is the degree to which users can understand the characteristics of a playlist. In this dissertation, we achieve playlist-level intelligibility by playlist captioning and by playlist tagging. For example, the playlist of Figure 1.1 can be tagged as "jazz" & "alternative", and can be captioned as "Derivations of jazz, from the 70s to our time". Improvements to playlist-level

[6]`https://www.ismir.net.`
[7]`https://backlinko.com/spotify-users`

intelligibility allow for several applications under the scope of playlist organisation and access. For example, playlist captions and tags can enable: search and discovery of playlists through human-like queries [MBQF21]; explainability of playlist recommendations [AMS$^+$22]; and provision of assistance to playlist curators when they are finding an appropriate title and/or description for a playlist.

1.3 Contributions

The work in this dissertation is about the intelligibility of music playlists, at song and playlist-level. In the following, we list our main contributions to this topic and how they relate to the chapters of this dissertation. Notice how the contributions from two to five fit into the topic of song-level intelligibility, while the contributions six and seven fit into the topic of playlist-level intelligibility. We release the source code supporting all the contributions we make in this dissertation. And, in those cases where we are allowed to do so, we also release the datasets as a complement to the source code.

1. **A survey of MIR research on music playlists** We present a survey of MIR research on music playlists in Chapter 2, which helps to position intelligibility in the context of other research on playlists. Our survey is extensive in the sense that it comprises all the MIR research on music playlists, spanning more than 20 years, and including around 200 papers.

2. **DAVE, an algorithm for generating song-to-song segues** We propose DAVE, an algorithm for generating song-to-song segues. A distinguishing feature of DAVE is its ability to highlight interesting segues, according to a novel measure of interestingness. DAVE assumes a knowledge graph as an abstract representation for songs and information about songs. In this abstraction, segues are paths from one item to another, and $interestingness$ is a scoring function for paths. We 'get back' from the abstraction by mapping paths to natural language texts. DAVE can generate song-to-song segues of 1553 different types, ranging from factual to word-play. We evaluate DAVE qualitatively by means of a user trial, where we compare DAVE's segues against curated segues from a segment of the Radcliff & Maconie Show on BBC Radio 6 called THE CHAIN. In the case of factual segues, we find that DAVE can produce segues of the same quality, if not better, than those to be found in THE CHAIN. And, we find that our

measure of interestingness positively correlates with human perceptions of segue quality. This content is published in [GB21a], and can be found Chapter 3.

3. **Three algorithms for generating music tours** We propose three algorithms for generating music tours: GREEDY, HILL-CLIMBING and OPTIMAL. The three algorithms take as input a collection of songs, and output an arrangement of the songs, with segues in between the songs, i.e. a musical tour, such as the one in Figure 1.4 (b). We formalise the problem of finding a musical tour as an optimisation problem, where the objective is maximising the interestingness of the segues, according to the interestignness measure we propose in [GB21a]. We set-up an offline experimental protocol, where we compare the interestingness of the segues in the tours produced by the three algorithms. This content is published in [GB21c], and can be found Chapter 4.

4. **A user-centered investigation of music tours** We consider two of the algorithms that we propose in [GB21c], GREEDY and OPTIMAL, and we set up semi-structured interviews, where interviewees judge tours generated by the two algorithms. We do not include the HILL-CLIMBING algorithm because, as reported in [GB21c], we found that it produces tours mostly equivalent to OPTIMAL, especially for small inputs. In these semi-structured interviews we investigate: what is a good segue arrangement; what is a good song arrangement; what topics should segues cover; how do we select the right music for tours; what is the overall quality of the tours recommended by the two algorithms; and what is the interestingness of their segues, all from the user perspective. Finally, we are interested in assessing whether users value the concept of tours in general, a fundamental issue already addressed by [BMT$^+$19], that we investigate further. With this work, we add practical and actionable knowledge to the literature on music tours, which can inform functional algorithms to generate tours. This content is published in [GB22], and can be found Chapter 5.

5. **Four algorithms for playlist tagging** We propose four algorithms for tagging music with listening context tags. The playlist taggers we propose are: KG-AVG, KG-SEQ, HYBRID-AVG and HYBRID-SEQ. Examples of listening context tags are "workout" and "party", which characterise playlists as being suitable for listening to by users while working out, and while

having a party. To the best of our knowledge, there exists only one other attempt to predict the listening context of music playlists: [CKE20]. The authors of [CKE20] propose four playlist taggers, which are limited in that they do not incorporate song metadata, such as musical genres. The algorithms we propose can incorporate song metadata, by using a knowledge graph as data model. We set-up an offline experimental protocol, using a dataset of playlists annotated with their listening contexts. The algorithms we propose achieve approximately 10% higher accuracy than the existing playlist taggers. And, a sensitivity analysis reveals that the algorithms we propose can incorporate song metadata effectively. This content is published in [GB23], and can be found Chapter 6.

6. **PLAYNTELL, an algorithm for playlist captioning** We propose PLAYNTELL, an algorithm for playlist captioning. To the best of our knowledge, there exist only two works on playlist captioning: [CFM$^+$16] and [DLN21]. Although promising, these attempts are afflicted by two main problems: (1) The data they use is of poor quality: in particular, they use public, crowdsourced datasets, which are very sparse and noisy; (2) Their approach must bridge a semantic gap: they represent playlists using low-level information, such as song audio, while the target captions are at a high semantic level. The algorithm we propose, PLAYNTELL, narrows the semantic gap by considering several sources of musical knowledge, such as song audio, user tags, and other hand-crafted features. And, we assemble a new high-quality dataset of editorial playlists from two major music streaming services, that we use in our off-line experiments. We find that PLAYNTELL largely outperforms existing playlist and music captioning algorithms in accuracy. We also provide a qualitative analysis of PLAYNTELL, as well as an ablation study and sensitivity analysis to validate the contribution of the different sources of musical knowledge and of the different model components. This content is published in [GHE22], and can be found Chapter 7.

Contribution six on playlist captioning is the outcome of a six-month internship that Giovanni Gabbolini completed at Deezer Research[8]. The research was supervised by two scientists from Deezer Research: Elena Epure and Romain Hennequin. A brief summary of the division of labour is as follows. The idea of working on playlist captioning was from Epure and Hennequin. Epure,

[8]https://research.deezer.com.

Hennequin and Gabbolini collaborated on the design of the algorithms and the design of the evaluation. Gabbolini implemented the algorithms, and evaluated the algorithms offline. Epure and Hennequin conducted the user study.

1.4 Outcomes

The work described in this dissertation has resulted in several publications. One of these publications received two distinct research awards. The work featuring in this dissertation was also the subject of several outreach activities towards industry and the general public.

Publications

1. Giovanni Gabbolini and Derek Bridge. Generating Interesting Song-to-Song Segues With Dave. In Proceedings of the 29th ACM Conference on User Modeling, Adaptation and Personalization (UMAP '21), June 21–25, 2021, Utrecht, Netherlands. ACM, New York, NY, USA, 10 pages. `https://doi.org/10.1145/3450613.3456819`;
2. Giovanni Gabbolini and Derek Bridge. Play It Again, Sam! Recommending Familiar Music in Fresh Ways. In Fifteenth ACM Conference on Recommender Systems (RecSys '21), September 27-October 1, 2021, Amsterdam, Netherlands. ACM, New York, NY, USA, 5 pages. `https://doi.org/10.1145/ 3460231.3478866`;
3. Giovanni Gabbolini and Derek Bridge. A User-Centered Investigation of Personal Music Tours. In Sixteenth ACM Conference on Recommender Systems (RecSys '22), September 18–23, 2022, Seattle, WA, USA. ACM, New York, NY, USA, 10 pages. `https://doi.org/10.1145/3523227.3546776`;
4. Giovanni Gabbolini and Derek Bridge. Predicting the Listening Contexts of Music Playlists Using Knowledge Graphs. In Advances in Information Retrieval: 45th European Conference on IR Research (ECIR '23), April 2-6, 2023, Dublin, Ireland. Springer, 16 pages.
5. Giovanni Gabbolini, Romain Hennequin, and Elena Epure. Data-Efficient Playlist Captioning With Musical and Linguistic Knowledge. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP '22), 7-11 December 2022, Abu Dhabi, UAE. 15 pages. `https://aclanthology.org/2022.emnlp-main.784.pdf`

Publications one to five above map to the Chapters 3 to Chapter 7 of this dissertation. We have another publication, not directly related to playlist intelligibility, but still worth mentioning. In this additional publication, we propose IPSim, an interpretable music similarity measure. IPSim works in a way that is similar to DAVE. In fact, IPSim assumes a knowledge graph as an abstract representation for items and information about those items; it finds paths in the graph between a seed and a target item; it scores those paths using the interestingness measure we propose in Chapter 3; and it aggregates the scores to determine the similarity between the seed and the target. Items can be songs, music artists, or any other entities of interest. Hence, IPSim is a path-based similarity measure, as it uses paths and a scoring function for paths to compute similarity. A distinguishing feature of IPSim is its interpretability. Users can understand the causes of a similarity score by looking at the natural language segues that correspond to the paths that were used to compute the similarity score. We test IPSim in the case when the items are artists. We set-up an offline experimental protocol, using four different datasets, and several other path-based similarity measures as baselines. IPSim outperforms the baselines in accuracy, proving that *interestingness* is an appropriate scoring function for paths. This content is published as:

- Giovanni Gabbolini and Derek Bridge. An Interpretable Music Similarity Measure Based on Path Interestingness. In Proceedings of the 22nd International Society for Music Information Retrieval Conference (ISMIR '21), November 7-12, 2021, Online. 7 pages. `https://archives.ismir.net/ismir2021/paper/000026.pdf`;

Additionally, we plan to submit the content presented in Chapter 2 for publication, in the form of a survey paper.

Recognition

The paper "Generating Interesting Song-to-Song Segues With Dave", in reference [GB21a] received two research awards:

- James Chen best student paper award 2021[9] at the conference on User Modeling, Adaptation and Personalization (UMAP);
- Best paper award 2021 within the School of Computer Science & Information Technology in University College Cork.

[9]`https://www.um.org/index.php/awards/james-chen-best-student-paper-awards`

Outreach

- Derek Bridge spoke about the work presented in the paper "Generating Interesting Song-to-Song Segues With Dave", in reference [GB21a], on the Radcliffe & Maconie Show on BBC Radio Six[10];
- Giovanni Gabbolini joined Deezer research for a six-months internship. The internship opened the way to a collaboration between University College Cork and Deezer. The collaboration is developing at the time of writing;
- The paper "Generating Interesting Song-to-Song Segues With Dave", in reference [GB21a], originated a research collaboration between the V&A museum in London[11] and University College Cork, which is active at the time of writing.

1.5 Outline of the dissertation

The dissertation continues in the following Chapters with the various contributions we list in Section 1.3, exactly in the order that we list them. Chapter 8 concludes the dissertation with an overview of future work.

[10]https://www.bbc.co.uk/programmes/b0100rp6.
[11]https://www.vam.ac.uk.

Chapter 2

Related work

2.1 Introduction

In this Chapter, we present an extensive survey of MIR research on music playlists. We survey all the MIR research on music playlists, spanning more than 20 years, so as to include around 200 papers. Our survey is the first self-contained survey to include all the different MIR research on music playlists, and it is particularly useful to position intelligibility in the context of other research on playlists. The remainder of this Chapter is organised as follows: in Section 2.2, we formalise the concept of playlists, we divide the research on playlists into several research topics, and we compare the research on playlists with other related research; in Section 2.3, we summarise existing surveys on playlists, and we describe the relationships between these existing surveys and our survey; in Section 2.4, we describe how we select relevant papers for inclusion in this survey; and in the remaining Sections we survey those relevant papers, by dividing them into the research topics we identify in Section 2.2.

2.2 Definitions and research landscape

In this dissertation we define the concept of playlists as follows:

Definition 1. *A playlist is a sequence of songs, intended to be listened to together.*

Definition 1 is equivalent to what we find in notable references, e.g. [FL10, SZC$^+$18, ZSLC19].

Sometimes, a playlist is defined in simpler terms, for example [BJ14] write that

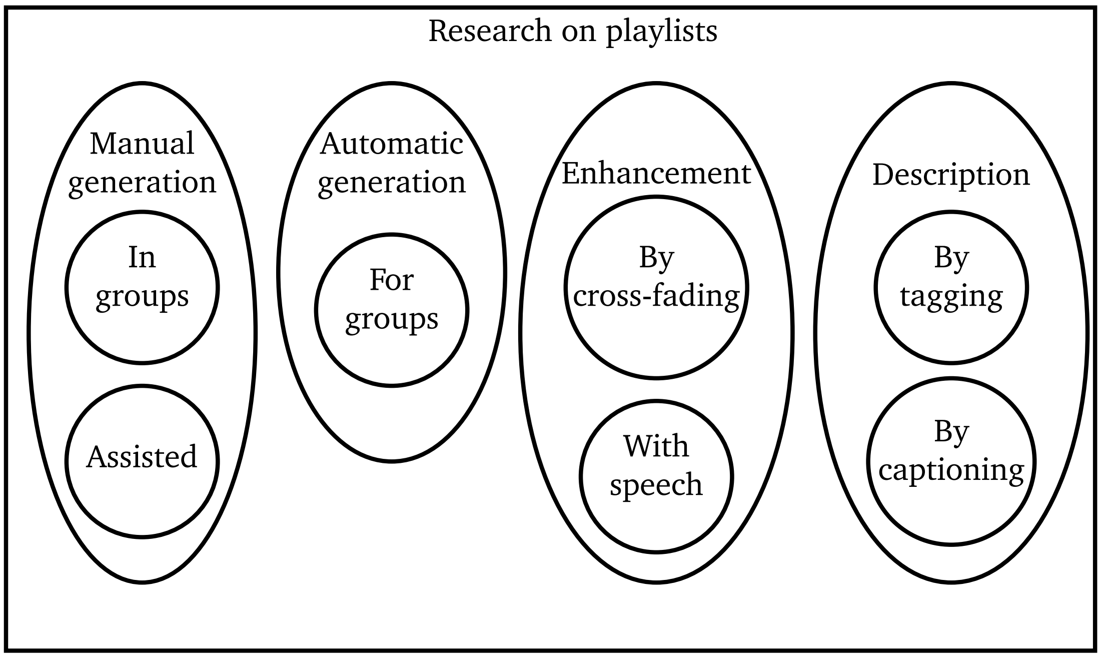


Figure 2.1: Venn diagram of the research topics we survey.

a playlist is “a sequence of songs”. 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 organised according to some principle [CBF06]. Definition 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, [KBJ20] find that the order does not matter, while [DMV97] find that the order does matter. We further discuss the issue in Section 2.6.2.2.

In the remainder of this Section, we present several research topics on playlists, and then we briefly relate the research on playlists to research on music recommender systems, and to the research on sequence-aware recommender systems in other domains, such as the tourism domain.

2.2.1 Playlist research topics

We divide the research on playlists that we survey into several topics. We organise those topics with a Venn diagram in Figure 2.1, and we briefly introduce these topics in the following Sections.

2.2.1.1 Playlist generation

Research on playlist generation is concerned with the construction of playlists. We adopt the definition of playlist generation presented in [BJ14]:

Definition 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 organisation principles which make a playlist a sequence of songs to be listened to together, which is consistent with Definition 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 manual and automatic:

Manual playlist generation (MPG) In MPG, the playlist is constructed manually by the user. MPG is an activity that dates before the digitisation of music. For example, in the 1980s it was common to compile mix-tapes, and to exchange those mix-tapes with other users [Fre08]. 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 [Fly16]. 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 [Web20]. In music streaming services, all users have access to the same catalogue of music, without actually owning any of it. But playlists allow uses to select and organise 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[1] music streaming service create their own playlists, and that 32%

[1]https://spotify.com.

of these users share their playlists with other users [SZC+18].

Automatic playlist generation (APG) In APG, the playlist is constructed automatically for the user by an algorithm. For example, [FSGW08] present an algorithm for creating a playlist that progresses smoothly from a specified start song to a specified end song. APG is a major research topic, counting hundreds of publications. The interest in APG is motivated by the fact that MPG can be experienced as a tedious, time-consuming activity, and it may require special background knowledge [BJ14].

The above characterisation of MPG and APG tacitly assumes that playlists are, respectively, constructed by and for a single user. However, there exists a special subset of work on MPG and APG which deals with the case in which playlists are, respectively, constructed by and for a group of users. We identify these topics as group MPG (G-MPG) and group APG (G-APG).

Some research in MPG considers the scenario in which the user is assisted by an algorithm while manually constructing the playlist [DGF17]. For example, in [KBJ20] the user is assisted by an algorithm that recommends the next song to add as the user constructs the playlist, while [Voo06] propose an algorithm for organising songs in a colour 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.

2.2.1.2 Playlist enhancement

Research on playlist enhancement is concerned with automatically decorating existing playlists with additional content, so as to increase the enjoyability of the playlist. We identify two enhancements, one based on mixing consecutive songs, so as to obtain a continuous music flow, and the other based on interleaving pairs of consecutive songs with speech, similar to what happens on a radio show. We note that enhanced playlists generalise the definition of playlists that we gave in Definition 1, since a ‘regular’ playlist can be seen as an enhanced playlist with a “null” enhancement.

2.2.1.3 Playlist description

Research on playlist description looks 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. In fact, music streaming services feature billions of playlists created by users, professional editors or algorithms [Dea21]. Playlist description allows for effective and automated organisation and access to playlists [CFM$^+$16]. We identify two ways of describing playlists, one is by using tags, which are closed-vocabulary short textual descriptions, naturally limited in expressiveness, and the other is by captioning with well-formed natural language, which is expressive but more complex to generate.

2.2.2 Related research topics

The music information retrieval research on playlists that we describe in this survey has relationships with other research, most notably with research into recommender systems (RS), and especially with the topics of sequence-aware recommender systems (SARSs) and music recommender systems (MRSs).

2.2.2.1 Sequence-aware recommender systems

Sequence-aware recommender systems (SARSs) are algorithms that are able to predict the next item given a sequence of user interactions. For example, in an e-commerce scenario, a SARS would analyse a user interaction log, which may consist of product impressions seen by the user, and of the products bought by the user, and would recommend another product for the user to buy. SARSs can also be used to recommend a sequence of items, 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.

Research in APG is very much related to the research in SARSs, since many algorithms for APG can be seen as applications of SARSs in the music domain, which recommend the next song to add to the playlist based on the songs already in the playlist, see Section 2.5.1.2. However, the music domain is different from other domains, for a number of reasons, as argued by Schedl *et al.* in their survey on music recommender systems [SKMB22]. 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 SARSs explicitly for the music domain, which are of interest for this survey.

2.2.2.2 Music recommender systems

Music recommender systems (MRSs) are recommender systems that work in the music domain, tackling tasks such as the recommendation of a personalised selection of songs, artists or albums [SKMB22]. Algorithms for APG are also tasked with recommending a selection of songs, i.e. the playlist, so 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 2.6.2.

2.3 Related surveys and contributions

Music information retrieval research on playlists is a two-decades-old research field, the oldest reference in our survey here dating back to 1997 [DMV97]. We are not the first authors to survey the literature. To the best of our knowledge, there are two surveys related to ours:

1. Bonnin & Jannach's "Automated Generation of Music Playlists: Survey and Experiments", published in 2014; and
2. Dias *et al.*'s "From Manual to Assisted Playlist Creation: a Survey", published in 2017.

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 & expert annotations, social web data, and usage data. We give more details in Section 2.5.1. In our survey here, we borrow some ideas from survey (1), as we also characterise 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 exclusively survey. For example, we survey APG algorithms based on reinforcement learning, which are relevant to our survey here. Inevitably, those published since 2014 did not make it into survey (1); but there were also papers on this topic that were published prior to 2014 that did not make it into survey (1) since reinforcement learning was not one of the algorithm types considered in that survey.

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 survey (2) on APG is a subset of what is included in survey (1), while the part of survey (2) on MPG briefly discusses characteristics of manually created playlists, such as recurrent playlist themes, and notable manual construction styles. Nevertheless, the main contribution of survey (2) is a review of A-MPG algorithms. It presents several algorithms for A-MPG, all based on visualisations, i.e. that assist the user in the manual construction of a playlist by using visualisations of the song catalogue. Survey (2) identifies several categories of A-MPG algorithm, based on the kind of visualisation that is employed: maps, graphs, dots and radar. In our survey here, we include the A-MPG algorithms that were covered by survey (2), as well as other work published from 2017 onwards, which, inevitably, did not make it into survey (2). However, the recent work does not belong to any of the categories proposed in survey (2). For example, recent work provides lists of song recommendations for addition to the playlist, which is not a catalogue visualisation of any kind. Hence, we propose a novel categorisation of A-MPG algorithms, to cover both recent and non-recent work. Specifically, we divide algorithms into two categories: visualisation, and recommendation.

Despite some overlap, our survey here is substantially different from surveys (1) and (2), as we give a fresh look at APG and A-MPG research, by including relevant work published after the publication of surveys (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 International Federation of the Phonographic Industry [IFP22], the share of revenue coming from music streaming was 10% of the total revenue for recorded music in 2013, 29% in 2016 and 65% in 2021.[2] 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 created playlists, as well as listening logs where the streaming service records the actions of its users when they are listening to music. Recently, large datasets of usage data became available for researchers to use. For example, the Million

[2]Data for 2022 was not available at the time of writing.

Playlists Dataset (MPD), released in 2018, contains manually created playlists, and is one order of magnitude larger than the datasets commonly employed in the research included in survey (1) [BJ14]. The availability of usage data has dramatically changed research in APG. The recent APG algorithms that we exclusively survey here rely mainly on usage data, while few of them employ content-based data, metadata & expert annotations, which is the predominant background knowledge used in APG algorithms included in survey (1) [BJ14]. See Table 2.1 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 [KSH17]. Many recent APG algorithms that we exclusively survey here use deep learning and reinforcement learning, while few of them rely on music similarity, which was the predominant approach of APG algorithms included in survey (1) [BJ14]. See Table 2.3 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 optimisation problems with non-linear complexity. As a result, we do not encounter any discrete optimisation APG algorithms in the recent research that we survey exclusively, see Table 2.3. The shift from small to large song catalogues has also impacted research on A-MPG, as the goal of the recent A-MPG algorithms that we exclusively survey 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 visualisations that assist the user in the manual construction of playlists by visualising the song catalogue in a map or in a graph, and which do not scale to catalogues of millions of songs.

Finally, our survey includes a substantial amount of work that was not included in survey (1) and (2), such as the research on group automatic playlist generation (G-APG), on group manual playlist generation (G-MPG), as well as the research on playlist enhancement and playlist description. Our survey also includes an extensive account of research on MPG. Survey (2) also includes research on MPG, but it gives only a partial account, from just five papers. In our survey here, instead, we review 38 papers on MPG, and we cover MPG topics not covered in survey (2), such as a detailed discussion of the role of song diversity and homogeneity when manually selecting songs.

2.4 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, recursively. 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 recommender system 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) [3], which is the premiere venue for research in music information retrieval. In particular, based on papers published in ISMIR, we crafted a search string that includes keywords from the titles of those papers.[4] We use the string to search the academic aggregator dbpl[5].

In total, our survey reviews around 200 relevant papers, and it also cites over 100 additional supporting sources.

[3] `https://ismir.net/`.

[4] 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|

[5] `https://dblp.org/`.

2.5 Automatic playlist generation

Automatic playlist generation (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 the algorithms in Section 2.5.1, and the strategies for evaluating those algorithms in Section 2.5.2. Additionally, we present in Section 2.5.3 a special subset of APG algorithms, group APG (G-APG) algorithms, which are concerned with generating playlists to be listened to by a groups of users.

2.5.1 Algorithms

Bonnin & Jannach [BJ14] surveyed the literature on APG in 2014, organising APG 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, *etc*. According to Definition 2, three inputs are required for an APG algorithm: a catalogue of songs, background knowledge, and some target characteristics of the desired playlist. The first two of Bonnin & Jannach's attributes correspond with two of the inputs that are mentioned in the definition of APG. In this Section, we adopt Bonnin & Jannach's organisation: we also characterise APG algorithms based on the same three attributes, but we complement their work, as we incorporate the newer literature up to the date of writing.

2.5.1.1 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, Bonnin & Jannach [BJ14] identify several categories of background knowledge. In this Section, we review those categories, while adding fresh details from the recent work that we exclusively survey. We organise APG algorithms based on their background knowledge in Table 2.1. We highlight differences from the research surveyed by Bonnin & Jannach [BJ14] to the research we exclusively survey by dividing algorithms published up to 2013 from

Table 2.1: Organisation of APG algorithms, based on the category (and sub-category) of background knowledge they use, and dividing those published up to 2013 from those published from 2014 onwards. Those published up to 2013 are also reviewed by Bonnin & Jannach in their 2014 survey [BJ14]. The categories are the ones used by Bonnin & Jannach.

Category	Up to 2013	From 2014
Content-based-data	[Cha02, PPW05, KPSW06, PKS$^+$07, SPKW07, DSPK08, ET06, MvNL10, OFM06, OKS06, JMPS07, LH11, HLE10, RBBC07, GCW13, AP02b, AP02a, BvdH10, BSDK12, CZZM07, CZS11, CDR13, FRCJ08, FSGW08, GKS07, Log02, Log04, LS01, MEDL09, VP05, XLSZ09, BGH$^+$17, Cli00, ML12, SC12]	[BELK$^+$17, LSTS15, KI15, GCW16, BCR$^+$22, IOK18, HSL17, VDSW18, VEZD$^+$17, JLK15, LSTS19, SVC$^+$18, CZHY22, VRC$^+$18, FBY$^+$18, ABC$^+$18, PBdGS19, IBZ$^+$19, CFS16, FMM19, STOH20, STOH21, IEPR22, PKW$^+$20, FD21, FM22]
Metadata and expert annotations	[Cha02, BAB06, JMPS07, LH11, HLE10, RBBC07, CTLH10, CCG08, AT00, AT01, CA09, CDR13, HY13, HC11, HO11, PRC00, PE02, PvdW05, PVV06, PVV08, PBS$^+$01, VP05, ZGMRMF10, BJC11, ML12]	[HSL17, GSM17a, GSM17b, GSJ18, VSC16, JLK15, KJL16, VRC$^+$18, VGMS18, RKV$^+$18, ABC$^+$18, PBdGS19, STOH22]
Social web data	[KPSW06, PKS$^+$07, SPKW07, DSPK08, HMB12, CA09, CDR13, FRCJ08, GC13, MCJT12, RM06, SACH06, SW07, ML12]	[BELK$^+$17, SBI14, VDSW18, VEZD$^+$17, JLK15, JKL17, KJL16, KKMS16, OKT10, PBdGS19]
Usage data	[HMB12, BJ13, AKS12, BP06, CMTJ12, GC13, HO11, MEDL09, MCJT12, RBH05, ZGMRMF10, ML12]	[BELK$^+$17, VQSW18, VQS$^+$17, VQSW19, Jan15, KES16, BCR$^+$22, CCC17, HSL17, SC18, VSC16, TCC$^+$15, VDSW18, VEZD$^+$17, JL17, JLK15, GSM17b, GSM17a, GSJ18, JKL17, KJL16, PBS$^+$01, UKM18, dONLF19, ZHJ$^+$18, ZSCH18, YJCL18, VRC$^+$18, VGMS18, RKV$^+$18, vNdV18, KWLH18, KBBB18, KB18, FPA18, ABC$^+$18, LKLJ18, MPR$^+$18, TSL19, IBZ$^+$19, CFS16, STOH22, STOH20, STOH21, YBK21, YKB21, IEPR22]

those published from 2014 onwards.

2.5.1.1.1 Content-based data Researchers in the field of music information retrieval (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 [MK18].[6] These content-based features can be high-level, such as the emotions evoked by a musical piece [ZML16, SVC+18], its genre [COKG11], timbre [PGS+11], chords [Ell06], pitch [ZK06], and beats per minute (BPM) [Sch98], or low-level, such as representations of the audio signal, for example mel frequency cepstrum coefficients (MFCC) [ZZS01], or such as learned embedding representations, as extracted, for example, by convolutional neural networks (CNNs) [IBZ+19, PKW+20]. Often, low-level features are used for extracting high-level features, e.g. see [PNP+18, WCNS20, CFSC17]. We refer the reader to [MK18] for a survey of content-based MIR.

Some APG algorithms rely on high-level content-based features. For example, [GCW13] extract the emotion evoked by songs, and construct a playlist that matches the emotion of the user, which is extracted by using several sensors. The same approach is taken in [GCW16, SVC+18]. The work [BCR+22] is similar to the above, except that users manually input their current emotion, e.g. melancholy. Liebman *et al.* [LSTS15, LSTS19] propose a reinforcement learning (RL) algorithm for APG 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, [PPW05] compute the similarity between songs in the catalogue based on a MFCC representation, to create a playlist in which consecutive songs are as similar as possible, so as to guarantee a coherent listening experience. A similar approach is followed by [BvdH10, FSGW08, Log02, Log04, LS01]. In [IBZ+19, CFS16], 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.

2.5.1.1.2 Metadata and expert annotations Following [BJ14], we use the word metadata to refer to any information describing the songs that is not de-

[6]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 [MGL+22]. We position our survey more in line with the MIR community, in which content-based features typically refers to those features which are extracted from the music audio signal [MK18]

rived from the audio signal. Examples of song metadata are the year of release, the record label, the lyrics, and the genre[7], among others [GB21a]. Usually, experts manually annotate songs with their metadata. A notable example is the Music Genome Project [Cas06], a database of songs and their metadata created and maintained by experts employed by the Pandora music streaming service.[8].

Different types of metadata are sometimes represented in a single structure, for example in a knowledge graph, where nodes can represent songs but also heterogeneous types of metadata [FŞA$^+$20], and edges express the relationship between the songs and the metadata, and the metadata with other metadata. For example, song names, album names and years can be represented as nodes, and a “belongs to” edge can link a specific song name to its album name, and a “released in” edge can link the album name to its year of release [GB21a]. Knowledge graphs are also used in [ML12, UKM18, dONLF19]. Edges in a knowledge graph might also represent relationships between classes, subclasses and instances, e.g. between genres and subgenres. Indeed, ontologies and taxonomies offer an alternative to knowledge graphs, placing the focus on classes, subclasses, instances and their properties. For example, [BELK$^+$17] use a taxonomy of musical genres.

One way to use metadata in APG is by allowing users to specify constraints on the metadata, and then generate a playlist that satisfies those constraints. For example, in [PVV08] users can input constraints on the song genre, release year and length (in seconds), and an optimisation algorithm is used to generate a playlist which satisfies these constraints. The same strategy is followed in [Cha02, PvdW05, PVV06, PRC00, HC11, AP02a, AT00].

2.5.1.1.3 Social web data Social web data are data shared online by internet users. Following [BJ14], we list three types of social web data:

User tags A user tag[9] is a free text annotation that a user applies to a musical

[7]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 [PRC00]; in recent work, accurate content-based MIR algorithms are often employed to extract the musical genres of large scale song catalogues [PNP$^+$18].

[8]`https://pandora.com`.

[9]The word “tag” is ambiguous. In some work, it is used to indicate free text, e.g. [Lam08]; in other work, it is used to indicate a text drawn from a fixed vocabulary, e.g. [CKE20]. This dissertation needs to use the word “tag” in both of its meanings. For example, in this Section and in Chapter 7, tags are free text, while in Chapter 6 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,

item, e.g. a song or an artist [Lam08]. 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, usually expressed in a 1-to-5 rating scale or as a “like” or “dislike” statement. 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 [SKMB22]. 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[10]. Social graphs are sometimes used as background knowledge, under the assumption that people who are closely connected in the graph have similar musical tastes [GC13]. 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.

2.5.1.1.4 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 to interpret a skip as an indication of a song that the user does not like [ET06, KI15, CTLH10, HSL17, BELK+17]. Listening logs can be used to compute embedding representations. One strategy is to rely on the word2vec algorithm [MSC+13], by treating songs as words, and listening logs as phrases, by analogy with natural language.

Popularity Usage data give 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

and we use simply “tag” where there is a fixed vocabulary.

[10] `https://facebook.com`.

APG. For example, [BJ13] 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. Moreover, popularity can be employed as a fallback strategy for estimating the musical tastes of new users, i.e. those that just signed up to the streaming service.

Manually created playlists Users frequently create playlists for convenience [Fly16] and self-expression [Web20]. These user playlists can be used as background knowledge for creating new playlists. For example, [ML11] learn song-to-song transition probabilities based on a database of user playlists, and they use these probabilities to generate new playlists. Manually created playlists can also be used to compute embedding representations. One strategy is to rely on the word2vec algorithm [MSC+13], by treating songs as words, and playlists as phrases, by analogy with natural language.

2.5.1.1.5 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 created playlists. The same goes for "long-tail" songs [KS16], i.e. those songs which are rarely listened to, that constitute the large majority of the catalogue [Cel10]. The unavailability of background knowledge for such portions of the catalogue leads to biases against new songs (cold-start problem [DDC+19]); and against long-tail songs (popularity bias [JKB16]). 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 is available for every song in the catalogue, as content-based data is extracted from the song audio itself. As such, content-based data allow 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, metadata are affected by a low level of noise as they are most often annotations made manually by domain experts. Nevertheless, inconsistencies in metadata may exist, especially because some metadata are not objective. For example, [FLR21] find that different annotators may disagree on the musical genre of songs. Content-based data are also effected by a low level of noise, as they are extracted by automatic procedures. Nevertheless, inconsistencies in content-based data may also exist because those automatic procedures are never 100% accurate. For example, predicting the tempo of a song is challenging and the results can be inaccurate [CLW09, PFC12].

Usage and social web data are typically much noisier than content-based and metadata, as they record the unpredictable behaviour of internet users. For example, Lamere [Lam08] analyses 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". Similarly, [Hag15] find that manually created playlists often do not have clearly defined target characteristics, but may be used as a randomly arranged container of the user's favourite music.

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 [SB13] 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 [SK05] or metadata [VEzDS16]. 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 personalised playlists. Hence, it is common to combine different categories of background knowledge. For example, [BELK$^+$17, JLK15, ML12] show how the quality of generated playlists is enhanced by making effective use of more than one category of background knowledge. One way to readily include different sources of background knowledge is to organise them in a unifying structure, for example a knowledge graph (described earlier), where nodes can represent songs but also heterogeneous types of background knowledge [FŞA$^+$20]. In a knowledge graph, listening logs, for example, can be represented alongside metadata. Nodes can represent users, songs, and metadata, and edges can link: users with the songs that they listened to; songs with their metadata; and metadata with other metadata [OON$^+$16].

If we refer back to Table 2.1, we can see the differences between the research up to 2013 (the *first period*), which was already surveyed by Bonnin & Jannach [BJ14], with respect to the research from 2014 onwards (the *second period*), that we exclusively survey. 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 [IFP22], the emphasis shifted to usage data, mainly due to the availability of that type of background knowledge, easily recorded by the music streaming service.

2.5.1.2 Target characteristics

The target characteristics of a playlist are the organisation principles which make a playlist 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, Bonnin & Jannach [BJ14] identify several categories of target characteristics. In this Section, we review those categories, and make a small update to better accommodate the recent work that we exclusively survey.[11]. We organise APG algorithms based on the target characteristics in Table 2.2. We highlight differences between the research surveyed by Bonnin & Jannach [BJ14] and the research we exclusively survey by dividing algorithms published up to 2013 from those published from 2014 onwards.

2.5.1.2.1 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 [BELK$^+$17], or the first and last song [AT00], or a list of

[11] We update the category "free-form keywords" proposed in the survey by Bonnin & Jannach [BJ14] to the more general "free-form text".

Table 2.2: Organisation of APG 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. Those published up to 2013 are also reviewed by Bonnin & Jannach in their 2014 survey [BJ14]. The categories and sub-categories are the same ones used by Bonnin & Jannach, except for a minor change we make so as to better accommodate the recent work we exclusively survey: we rename their category "free-form keywords" to "free-form text".

Category	Sub-category	Up to 2013	From 2014
Explicit preferences and constraints	Seed songs	[PPW05, KPSW06, PKS$^+$07, SPKW07, DSPK08, LH11, HMB12, BJ13, AKS12, PvdW05, BP06, BvdH10, BSDK12, CZZM07, CMTJ12, CZS11, FRCJ08, FSGW08, GKS07, GC13, Log02, Log04, LS01, MEDL09, MCJT12, PE02, PBS$^+$01, RBH05, SACH06, SW07, VP05, XLSZ09, ZGMRMF10, ML12, SC12]	[BELK$^+$17, LSTS15, LSTS19, BGH$^+$17, VQSW18, VQS$^+$17, VQSW19, SBI14, Jan15, KES16, IOK18, GSM17a, GSM17b, GSJ18, SC18, VSC16, TCC$^+$15, VDSW18, VEZD$^+$17, JL17, JLK15, JKL17, KJL16, UKM18, dONLF19, CZHY22, ZHJ$^+$18, ZSCH18, YJCL18, VRC$^+$18, VGMS18, RKV$^+$18, vNdV18, KBBB18, KB18, FBY$^+$18, FPA18, ABC$^+$18, LKLJ18, MPR$^+$18, PBdGS19, TSL19, IBZ$^+$19, CFS16, PKW$^+$20, FD21, FM22]
	Free-form text	[CDR13, MCJT12, RM06]	[CCC17, SVC$^+$18, ZHJ$^+$18, ZSCH18, YJCL18, VRC$^+$18, VGMS18, RKV$^+$18, vNdV18, KWLH18, KBBB18, KB18, FBY$^+$18, FPA18, ABC$^+$18, LKLJ18, MPR$^+$18, YBK21, YKB21]
	Explicit and pre-defined constraints	[Cha02, JMPS07, HLE10, AT01, FL10, AP02a, HY13, HC11, MEDL09, PRC00, PvdW05, PVV06, PVV08, RM06, Cli00]	[GCW16, BCR$^+$22]
	Real-time feedback	[CTLH10, CA09, GKS07, HO11, RM06]	[LSTS15, LSTS19, KI15, BCR$^+$22, HSL17, KKMS16, OKT10]
Past user preferences		[CBH02, CA09, FRCJ08, GC13, HO11, ZGMRMF10]	[BELK$^+$17, BCR$^+$22, HSL17, TCC$^+$15, JLK15, JKL17, KJL16]
Contextual and sensor information		[ET06, MvNL10, OKS06, OFM06, BAB06, RBBC07, CCG08, GCW13, RM06, HO11, AKS12, BJC11]	[SBI14, GSM17b, GSJ18, SVC$^+$18, FMM19, IEPR22]

songs already contained in the playlist [TCC+15], or the set of all the songs to include in the playlist[12] [KPSW06]. Some algorithms allow users to specify seed artists, instead of seed songs [BELK+17, SBI14].

Free-form text In this case, users specify constraints on the songs by providing free-form text, which is used to select relevant songs [CCC17]. The free-form text can be single keywords, such as artist names, musical genres, or moods [YBK21, YKB21]. But free-form text can also be well-formed natural language phrases. For example, the APG algorithm proposed in [SVC+18] 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 predefined and users choose among them, e.g. the user chooses a desired mood from six categories [BCR+22], or the user chooses a tag from a tag-cloud [MEDL09].

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, or implicit by listening to a song to completion or by skipping a song. The playlist can be modified in real time according to the feedback [KI15]. As well as giving feedback on songs, users can give feedback about metadata associated with the songs. For example, in [KKMS16, OKT10] 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.

2.5.1.2.2 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 [AH06]. For example, [LBM11] 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. This means that music in a playlist should be highly personalised. The musical preferences of a user are usually estimated by considering usage data, such as listening logs and manually created playlists [BJ13, RBH05] (see Section 2.5.1.1).

[12]In this last case, algorithms are tasked with arranging the provided set of songs, without applying any song selection. These special APG algorithms are sometimes called sequencing algorithms [BGH+17].

2.5.1.2.3 Contextual and sensor information The listening context influences the musical choices of users [ABTU22]. For example, [DWLX15] and [CS14] find that the user's mood and location influence their musical choices. Therefore, context-awareness is an important target characteristic. Mood and location are only two example of listening context, which is a broad concept. Indeed, Kaminskas & Ricci define the listening context as "any contextual conditions that might influence the user's perception of music" [KR12]. Other examples of listening contexts are: user activities e.g. "party" [CKE20]; the time of day [HRS10]; 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 [SBI14].

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. For example, [OMS16] build a model to infer a user's mood from audio signals detected by microphones, [SVC$^+$18] infer the user's mood from free-form text, and [IEPR22] infer what activity the user is engaging in from listening logs.

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, [BAB06, ET06, MvNL10, OKS06, OFM06] extract a model of user's pace by means of an accelerometer, and construct a playlist where the BPMs of songs depend on the user's pace.

2.5.1.2.4 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 [JLK15].

Nevertheless, the most common way of specifying the target characteristics is via seed songs, as Table 2.2 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 [ZSLC19], which is a task known as automatic playlist continuation (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 [SZC+18].

APC was the focus of the ACM RecSys Challenge 2018[13], in which participants were asked to add more songs to user-created playlists taken from the Spotify music streaming service [ZSLC19]. In total, 113 teams participated in the challenge, which represents a landmark in APC research. APC is the dominant research trend in APG: we count that 46% of the works in APG from 2014 onwards focus on APC.

2.5.1.3 Algorithm type

We review algorithms for APG based on their type. In their 2014 survey on APG, Bonnin & Jannach [BJ14] identify several types of algorithms. In this Section, we review those types, while adding three types that emerge from the recent work that we exclusively survey: deep learning; reinforcement learning; and learning to rank. We organise APG algorithms based on their type in Table 2.3. We highlight differences between the research surveyed by Bonnin & Jannach [BJ14] and the research we exclusively survey by dividing algorithms published up to 2013 from those published from 2014 onwards.

2.5.1.3.1 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 [PPW05, IOK18, AP02b, CZZM07, BvdH10], metadata [PE02, PBS+01], tags [SBI14, PKS+07], manually created playlists [BJ13, RBH05, MEDL09], listening logs [VSC16], ratings [SW07, CA09], or any combination of the above [JMPS07]. For example, [PPW05, CZZM07, BvdH10] compute the similarity between songs based on a MFCC representation; [PE02, PBdGS19] count the values of metadata features that two songs have in common; [BJ13] consider how often two songs co-occour in manually created playlists; and [VSC16, KES16, DBAH+18] rely on a song embedding representation learned using the word2vec algorithm [MSC+13], by treating

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

Table 2.3: Organisation of APG algorithms based on their type, and dividing algorithms published up to 2013 from those published from 2014 onwards. Those published up to 2013 are also reviewed by Bonnin & Jannach in their 2014 survey [BJ14], 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 & Jannach, except for some changes we make so as to better accommodate the recent work that we exclusively survey. Specifically, we include three additional algorithm types: deep learning; reinforcement learning and learning to rank. We exclude one of their algorithm types, frequent pattern mining, merging the work on frequent pattern mining into the work on statistical models.

Algorithm type	Up to 2013	From 2014
Similarity	[PPW05, KPSW06, PKS$^+$07, SPKW07, DSPK08, JMPS07, LH11, HLE10, BJ13, FL10, BvdH10, BSDK12, CZZM07, CA09, Cli00, CZS11, CDR13, FRCJ08, FSGW08, GKS07, GC13, Log02, LS01, Log04, MEDL09, PE02, PBS$^+$01, RBH05, SACH06, SW07, VP05, XLSZ09, SC12]	[SBI14, BCR$^+$22, IOK18, VSC16, BGH$^+$17, CZHY22, ABC$^+$18, PBdGS19, FMM19, PKW$^+$20, FD21, FM22]
Collaborative filtering	[AKS12]	[Jan15, TCC$^+$15, JL17, JLK15, JKL17, KJL16, ZHJ$^+$18, ZSCH18, VRC$^+$18, RKV$^+$18, KWLH18, KBBB18, KB18, FBY$^+$18, FBY$^+$18, FPA18, ABC$^+$18, LKLJ18, YBK21, YKB21]
Case-based reasoning	[BP06]	[GSM17a, GSM17b, GVJSM19, GSJ18]
Discrete optimisation	[PPW05, KPSW06, PKS$^+$07, SPKW07, DSPK08, AT00, AT01, AP02a, HY13, HC11, PRC00, PvdW05, PVV06, PVV08]	—
Deep learning	—	[VQSW18, VQS$^+$17, VQSW19, SC18, VDSW18, VEZD$^+$17, JL17, YJCL18, VRC$^+$18, VGMS18, KWLH18, MPR$^+$18, IBZ$^+$19, CFS16, STOH22]
Reinforcement learning	[CTLH10, HO11]	[LSTS15, KI15, LSTS19, STOH20, STOH21, STOH22]
Statistical models	[CMTJ12, MCJT12, ZGMRMF10, ML12, HMB12, CMTJ12, BJ13]	[BELK$^+$17, SBI14, KES16, CCC17, UKM18, dONLF19, vNdV18, TSL19]
Learning to rank	—	[ZHJ$^+$18, VRC$^+$18, RKV$^+$18]

songs as words, and manually created 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 [FMM19].

Integrating multiple sources of background knowledge is beneficial when computing similarity. For example, [BOL09, SM11] find that users consider content-based features, such as energy and tempo, as well as musical styles and lyrical content when judging the similarity of a song that could be added to a playlist with the songs already in the playlist.

The perception of song similarity is subjective. Even expert listeners are found to disagree when asked to rate the similarity between songs [Fle14]. For example, some people consider content-based data more than metadata while judging similarity, and some other users may do the opposite [LBM11]. Some work in APG integrates personalisation in the similarity computation. For example, [SLT07] 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. And, [SACH06] 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 personalised based on the user's tagging style.

Once similarities are computed, songs can be chosen for their similarity with the seed songs [Log02, BJ13, AP02b], or for their similarity with other songs liked by the user [HO11]. Another possibility is to create playlists so as to maximise the similarity between songs [KPSW06].

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, [PGG19]. Also, a risk with optimising for similarity is that the playlist may become monotonous [LBM11], e.g. containing songs from the same album. See Section 2.6.2.1 for a discussion of coherence and diversity.

One use-case for similarity algorithms is that of playlist sequencing, the special case of APG 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 that music is coherent, from one song to the next song [BGH+17]. For example, [BGH+17, Cli00] use a similarity algorithm working on content-based similarity. It compares the distance between songs based on several features, and then arranges the songs in such a way that those distances are minimised. Sarroff & Casey [SC12] 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, [FD21, FM22] go in the direction of personalised sequencing. They analyse song-to-song transitions in a user's playlists so that they can sequence playlists in a personalised way.

2.5.1.3.2 Collaborative filtering Collaborative Filtering (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 [RRS22]. Hence, the use of CF algorithms is facilitated by the existence of usage data, recording the preferences of other users.

The most common way of employing CF for APG is by applying the playlists-as-users analogy [BJ13, RKV+18], in which a user is a playlist, and the songs in the playlist are the songs that the user likes. Another common analogy is the titles-as-users analogy [ZSCH18, VRC+18, RKV+18], in which a user is a playlist title, and the songs in playlists with that title are the songs that the user likes. 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, the rest of this Section uses only the playlist-as-user analogy. The playlists-songs 'preference' matrix can be unary [JL17], i.e. recording a 1 if a playlist contains the song. Or, it can be non-unary; for example [TCC+15, ABC+18] assign a value to a song according to its position in the playlist, giving more weight to the last songs.

A popular option is to employ nearest neighbors CF algorithms [NNDK22][14]. For example, the system in [VQSW19] computes the similarity between the ac-

[14]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 categorised as Collaborative Filtering, especially in the RSs community [RRS22].

tive playlist and the other playlists in the dataset as the cosine similarity of their row-vector representations in the playlists-songs preference 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 contain 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 [KMM+97]. The user-based k-nearest neighbors algorithm is also employed in [Jan15, JL17].

Another option is to use the item-based k-nearest neighbors algorithm [DK04]. For example, the system in [VQS+17, VQSW18] computes the similarity between songs as the cosine similarity of their column-vector representations in the playlists-songs preference 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, where k is a positive integer hyper-parameter.

Some work uses similarity functions other than cosine; for example [TCC+15] use Jaccard. Additionally, the similarity functions can be augmented with heuristics, for example by giving higher weight to unpopular items [KBBB18].

There exist other CF algorithms for constructing playlists. For example, [AKS12] use a matrix factorization (MF)-based approach, in which the playlists-songs preference matrix is factorised 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 embedding. Finally, the highest-scoring song is selected to be added to the playlist. In [ZSCH18], 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.

The playlists-as-users analogy has two main limitations: (1) the constructed playlists are not personalised; (2) the performance depends on the number of songs in the playlist. 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 personalised. One approach, for example, is to modify the active playlist by adding songs from other playlists created by the same listener [JKL17, KJL16, AKS12]. Concerning limitation (2), CF algorithms are affected by the cold-start problem, which manifests with small or newly created playlists. 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 [ZSLC19]. 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 titles-as-users analogy shares the same two 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 [ZSCH18]. 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 [ZSCH18, YKB21, YBK21]. Another way to cluster titles is by employing text-embedding procedures, such as FastText [JGB$^+$16], and by running a clustering algorithm on those embeddings [MPR$^+$18].

One last drawback of CF approaches is that they are not designed for the specific challenges of APG, and aspects such as songs coherence have to be addressed separately [BJ14].

2.5.1.3.3 Case-based reasoning Case-Based Reasoning (CBR) is an approach to problem-solving that involves reasoning with prior experiences. CBR can be effective when two tenets hold [Lea96]: similar problems have similar solutions; and the types of problems an agent encounters tend to recur. Case-based APG assumes that existing playlists encode the results of prior reasoning, and that it is therefore worthwhile to reuse existing playlists when creating new playlists.

Given a dataset or case base of existing playlists and an initial seed playlist, [GSM17a] uses CBR to recommend a set of songs for playlist continuation (APC). The 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[15]. The approach is extended to include time-of-creation pre-filtering

[15]This CBR system could alternatively be classified as a similarity algorithm, but we review

[GSM17b] and shared latent topic pre-filtering [GVJSM19, GSJ18].

By contrast, [BP06] deals with case-based playlist generation from a seed song (APG), rather than case-based playlist continuation, and treats the order of the songs in the playlists in the case base as significant. In this 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. The idea is that recurring subsequences are meaningful: they capture songs that go well together and the ordering that makes them go well together.

Both [GVJSM19] and [BP06] have additional scoring mechanisms that try to take coherence and diversity into account, these being concepts that we discuss further in Section 2.6.2.1.

2.5.1.3.4 Discrete optimisation 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 maximising some utility function [BJ14].

Discrete optimisation approaches to APG differ in their constraints. [AT00, PRC00] 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 [KS16]. And, [AT01, AP02a, HY13, HC11, PRC00, PvdW05] impose constraints on metadata or content-based data, for example by requiring that at least n songs in the playlist should have a specific musical genre [AP02a].

Some approaches define a utility function to be maximised during the optimisation process. For example, [HY13, PKS$^+$07, KPSW06] seek to maximise the similarity of consecutive songs in the playlist, as measured by a song similarity measure [KS16].

Also, different approaches use different strategies to solve the optimisation problem. For example, [AT00, AT01] use linear programming, [AP02a, PRC00, PvdW05] use constraint satisfaction, [HY13, PVV08] use simulated annealing, and [HC11] use genetic algorithms.

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.

2.5.1.3.5 Deep learning Deep learning is a form of machine learning that is based on the usage of many-layered artificial neural networks. Deep learning has led to advances in different application fields of AI, such as natural language modeling [BMR+20] and object classification in images [KSH17]. Given those promising results, deep learning has recently been applied to the task of APG. The survey on APG by Bonnin & Jannach dates to 2014, which is prior to the application of deep learning to APG. Our survey here therefore contains an exclusive review of deep learning algorithms for APG.

One famous family of deep learning models are recurrent neural networks (RNNs) [LBE15]. 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, 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. [VQSW19, VQSW18, VQS+17, JL17] employ RNNs for APG by resorting to a particular RNN model, called GRU4Rec [HKBT15]. They train the RNN on a dataset of manually created 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 optimised so that it correctly predicts the song in position $n + 1$. Related work makes use of other RNN models, such as LSTMs [IBZ+19, CFS16]. [SC18] 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 reinforcement learning algorithm [HSSL19]. 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 [KWLH18].

Another famous family of deep learning models are the autoencoders. In the simplest case, an autoencoder consists of two components: encoder and decoder. The input data is first projected into a hidden representation by the encoder, and the decoder is tasked with reconstructing the input from the hidden representation. The hidden representation is set to be a low-dimensional vector. The output of the decoder is used to decide which songs to add to the playlist. A more sophisticated autoencoder is the adversarial autoencoder, in which the hidden vector distribution is regularised so as to match a gaussian prior distribution [MSJ+15]. Autoencoders of both kinds are used for APG by setting the input to be a binary vector indicating which songs are in the playlist [YJCL18, VGMS18]. The output is a vector approximating the input vector, that can be used for selecting other songs for addition to the playlist. The encoder and the decoder are neural networks. For example, [YJCL18] use simple feed-forward neural networks. [VGMS18] successfully integrate additional background knowledge in an adversarial autoencoder, by concatenating embeddings of textual data, such as the playlist title, to the hidden representation.

The last family of deep learning models we consider are the convolutional neural networks (CNNs). The use of CNNs was popularised in computer vision, where they yield state-of-the-art accuracy in tasks such as image recognition [KSH17]. More recently, CNNs have been adapted to do language modeling [KGB14] and APG. For example, by applying the songs-as-words analogy, it is possible to use a CNN to predict the next song in the playlist [VRC+18]. Moreover, by treating a playlist title as a sequence of characters, it is possible to use CNNs to process the characters, obtaining an embedding vector, that can be used to predict the songs in the playlist, given the title [YJCL18].

The transformer is a recently proposed deep learning model for modeling sequences [VSP+17]. Transformers have been applied in a number of fields, including natural language processing [VSP+17], recommender systems [GDW+20] and computer vision [KNH+22]. Transformers excel in modeling sequences, and especially long sequences, yielding increases in accuracy against other state-of-the-art models in a number of different tasks, such as machine translation [VSP+17]. However, to the best of our knowledge, transformers have not yet been applied to the task of APG.

Deep learning is often used as a powerful tool to combine heterogeneous features and information sources [SBE+21]. For example, [VDSW18, VEZD+17] 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.

2.5.1.3.6 Reinforcement learning Reinforcement learning (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 maximise 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 observes 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 [SB18].

Our survey here contains an exclusive review of RL algorithms for APG. The 2014 survey on APG by Bonnin & Jannach does not contain any RL algorithm for APG, either because those RL algorithms were not published in 2014, or because they were published but did not make it into their survey.

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, and the reward to be some notion of playlist quality.

RL is usually formalised as a Markov decision process (MDP), i.e. a tuple (S, A, P, R) such that:

1. S is the set of states of the environment;
2. A is the set of the agent's possible actions;
3. P is the transition function, which models the state s_2 that the environment transitions to after the agent takes an action a in a state s_1, i.e. $P(s_1, a) = s_2$ where $s_1, s_2 \in S$ and $a \in A$;[16]
4. R is the reward function, which models the reward r for taking an action a in a state s, i.e. $R(s, a) = r$, where $s \in S$, $a \in A$ and $r \in \mathbb{R}$.

The goal of a RL algorithm is to learn a policy, which determines what action a the agent should take in each state s so as to maximise the cumulative reward, where $s \in S$ and $a \in A$.

The APG work that is based on RL shares a similar MDP formalisation to the one above: a state $s \in S$ is the list of songs in the playlist. For example, if the playlist has been constructed up to the tenth song, and the agent is tasked with choosing the eleventh song, then s is the list of those ten songs; an action $a \in A$ is the action of adding a specific song to the playlist; and the transition function P is known, and defined by appending the new song to the list of songs in the playlist. The reward function R quantifies the playlist quality, and it is unknown. Some APG work based on RL estimates a return function from observed rewards, and use the return function to determine a policy, solving

[16]The definition of transition function we adopt is deterministic, which is consistent with the APG literature. In general, the transition function in MDPs is a stochastic function.

the APG problem. For example, [LSTS15, LSTS19] parameterise the estimated return function as a linear transformation of the newly added song's content-based data. In other APG work based on RL, the agent learns a policy from observed rewards directly. For example, [KI15, CTLH10, HSL17] use Q-learning [Wat89] to learn a policy from observed rewards.

One apparent issue with the MDP formalisation above is that of combinatorial explosion. For example, if the catalogue size is ten million songs, as it is in some music streaming services [Spo22], then there are 10^{70} possible states just for playlists that contain ten songs, and 10^7 possible actions. One way to tackle the state explosion problem is by factorising the song representation in terms of their features, such as content-based features [LSTS15, KI15, LSTS19], metadata [HSL17] or mood [CTLH10], and/or by applying windowing, e.g. by representing just the last three songs of a playlist in a state [CTLH10, HSL17].

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 [KI15]. In other work, rewards are observed explicitly, for example by asking the user to rate the newly added song on a numerical scale [LSTS15].

2.5.1.3.7 Statistical models Statistical models work by modeling the probability of adding a song to the playlist[17]. 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 research cited below, the state is defined as the last song in the playlist. As such, the core component of Markov models is the estimation of the song-to-song transition probabilities[18]. [ML11] 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 created playlists, while others rely on content-based or metadata similarity. Other Markov models are proposed in [CMTJ12, MCJT12, ML12, UKM18, dONLF19, vNdV18]. Markov models may lead to the construction of problematic playlists [VQS$^+$17]. For example, adding a song based only on the

[17]Some of the other types of algorithm above are also statistical models. In particular: reinforcement learning, deep learning and collaborative filtering are statistical models. Also, similarity algorithms can be used to build statistical models. However, we devote a separate Section to each of them as they are notable and recognisable algorithm types for RSs in general and for automatic playlist generation in particular.

[18]The difference between the Markov models of this Section and the MDPs of the previous Section is that the MDPs explicitly model the actions that cause the state transitions.

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, [HO11] consider a playlist as a time series and use an autoregressive integrated moving average (ARIMA) model [Har90] to predict the next song.

Frequent pattern mining approaches also do not make the Markov assumption, and work by mining sequential patterns from a dataset of manually created playlists. A sequential 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. Sequential 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. Sequential patterns are not often applied to APG. However, [BJ14] show that they can achieve comparable performance to other types of algorithms. For example, [CMTJ12] 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 [JM09]. A similar strategy is followed in [BJ14, BJ13]. In [HMB12], the PrefixSpan algorithm [HPMA$^+$01] 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 [BNJ03].

The most sophisticated statistical models are also personalised, 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, [BELK$^+$17] 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 [ZGMRMF10, TSL19].

Other notable statistical models are proposed for the scenario in which the target characteristics are specified using natural language. For example, [CCC17] propose a statistical model for linking a word to a song. It is trained on a dataset of manually created 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.

2.5.1.3.8 Learning to rank Learning to rank (LTR) is a form of machine learning that constructs ranking models for information retrieval systems [Liu09]. In practice, given a query and a list of items, the LTR algorithm is trained to re-rank the list of items, so that the most relevant items are at the top of the list. For example, in the context of a search engine, the query is provided by the user, who might search for "the counties of Ireland", and the list of items is a list of websites. The model learned by the LTR algorithm would rank the most relevant items at the top. In the search engine example, a website listing all the counties of Ireland would be ranked before a website listing all the counties in the state of Washington, US. For training, a LTR algorithm is given queries, the list of items to re-rank, along with the relevance of the items to each of the queries, and a set of features about the queries and about the list of items. In the search engine example, the relevance labels between the websites to the queries are usually given manually, and the set of features of the queries and websites are usually text features extracted automatically, for example TF-IDF features. LTR has recently been applied to the task of APG. The survey by Bonnin & Jannach dates to 2014, prior to the application of LTR to APG. Our survey here contains an exclusive review of LTR algorithms for APG.

In the context of APG, the query captures the target characteristics of a playlist; the items are the songs in the song catalogue, and their relevance to the target characteristics is usually binary, i.e. 1 if the song is in the playlist and 0 otherwise; the set of features distill sources of background knowledge about songs and about target characteristics, such as those we reviewed in Section 2.5.1.1. Examples of song features are: popularity [VRC$^+$18], song homogeneity and diversity [RKV$^+$18], as well as song embedding representations, computed, for example, from usage data [ZSCH18] or content-based data [VRC$^+$18].

In practice, LTR algorithms for APG are not used to rank the full song catalogue, which is computationally unfeasible. Instead, a two-stage architecture is usually employed [ZSLC19]. In the *first stage*, an APG algorithm is used to retrieve a list of candidate songs. In the *second stage*, a LTR algorithm is used to re-rank the candidate songs so as to select songs for addition to the playlist. The *first stage* algorithm can be any APG algorithm, and it is used to retrieve a relatively small list of songs, usually composed of thousands of songs. The *second stage* algorithm can be any learning to rank algorithm. We refer the reader to [Liu09] for a review of learning to rank algorithms. Typically, the *first stage* algorithm is chosen to be computationally inexpensive, so that it can retrieve the list of candidate songs in a short time. A typical choice for the *first stage* algorithm is a collaborative filtering algorithm [ZSLC19], such as matrix factorisation. The *second stage* algorithm, instead, is chosen to be a state-of-the-art learning to rank algorithm, typically a list-wise algorithm, such as XGBoost [CG16].

2.5.1.3.9 Discussion The types of algorithms we review differ in their performance. There is, for example, some evidence that CF and deep learning algorithms excel in generating high-quality playlists, especially in the case where the target characteristics are specified as a list of seed songs [BJ14, ZSLC19]. 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 algorithms of different types, or to combine algorithms of different types, in different circumstances, especially depending on the background knowledge available, and on the way the target characteristics are specified. For example, [SBI14] 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 [JL17, ZSLC19]. One common way of combining algorithms is to compute a weighted average of their predictions [VRC$^+$18, LKLJ18]. We refer the reader to [Bur02] for an understanding of the possible ways in which RS algorithms can be combined, many of which can be adapted to APG.

Table 2.3 reveals the differences between the research up to 2013 (the *first period*), which was already surveyed by Bonnin & Jannach [BJ14], and the research from 2014 onwards (the *second period*), that we exclusively survey. The majority of algorithms from the *first period* are similarity and discrete optimisation algorithms. In the *second period*, the emphasis has shifted to CF and deep learning algorithms. We believe this paradigm shift to be linked to the paradigm shift in music consumption, which is now dominated by streaming services [IFP22]. The song catalogues available for APG algorithms shifted from small personal collections to millions of songs [Spo22], which makes it impossible to utilise discrete optimisation algorithms, as their worst-case computational complexity is exponential. Music streaming services also made available large quantities of usage data, which explains the surge of work in CF.

2.5.2 Evaluation

Up to now, we have referred to playlist quality as the way we would measure the performance of APG 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 [WGI14], and on the listening context [ABTU22]. 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, Bonnin & Jannach [BJ14] 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 created playlists or listening logs, under the assumption that the manually created playlists or listening logs reflect a gold standard.

These three categories are still valid today, and cover the evaluation protocols used in the recent work on APG that we exclusively survey, i.e. in papers published from 2014 onwards. In the following, we focus on how APG algorithms are evaluated in the papers that we exclusively survey, i.e. in papers published from 2014 onwards.

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 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 algorithms, [BJ14, BJ13] set N to 1 and allowed K to range from 1 to 1000. They used hit-rate as M, which, for a ground truth playlist, measures whether the K recommended songs contains the withheld song. They reported the percentage of ground truth playlists for which there was a hit. In the RecSys Challenge 2018, instead, N is set as a fixed percentage of the playlist songs, K is set to 500, and M is set to a number of different metrics related to hit-rate, including nDCG and R-precision [ZSLC19].

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, [BJ13, BJ14] show that using average log-likelihood as M, which was used in some other work [ML11], leads to inconsistent conclusions with hit-rate related metrics, and recommend to avoid its use. Kamehkhosh & Jannach [KJ17] 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. They find that users are likely to select the withheld song as a favourite 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, [VQSW19] criticise 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.

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 2.2. However, evaluation protocol (3) can be adapted to work when the target characteristics are specified in different ways. For example, [CCC17] propose an APG 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.

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, [Hag15] find that manually created playlists often do not have clearly defined target characteristics. For example, they may sometimes be used as a randomly arranged container of the user's favourite 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 [BPK09]. 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 created playlists for evaluation is offered in [CBF06], as they find that playlists manually created 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 the popularity bias [BCC17]. Since most of the ground truth playlists tend to contain popular songs [Cel10], an APG strategy that constructs playlists in a popularity-driven fashion will usually yield good performance [BJ14, BJ13]. 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 [CDW$^+$22]. A simple strategy is used in [VQSW19], 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 algorithms. It works by computing statistics on the constructed playlists, under the assumption that those statistics reflect aspects of playlist quality. For example, [CCC17, JLK15, JKL17, KJL16] measure song diversity and coherence, by considering song tags or musical genres. Additionally, [CCC17] measure song popularity, while [SC18] 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. 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 [KBJ20, DMV97]. See Section 2.6.2.1 for a discussion of coherence and diversity.

Evaluation protocol (1) consists of user studies. For example, reinforcement learning 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 [KI15], or by explicitly asking the user if they like the music or not [LSTS15, LSTS19]. As another example, [IOK18] employ a user study to evaluate the smoothness of song-to-song transitions in playlists. 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, e.g. by monitoring the play counts [BCR$^+$22], which is an indicator of playlist quality. Unfortunately, A/B tests require resources not accessible to most researchers, such as the availability of a music streaming service platform in which the A/B test can be conducted. A middle ground between an A/B test and evaluation protocol (3) is counterfactual evaluation, which allows estimation of the performance of a candidate algorithm, as if it were in production, by relying on listening logs extracted from whatever algorithm is currently in production. Researchers at Spotify share their recipe for counterfactual evaluation in [GCC$^+$19], showing that they can rely on high correlation with actual A/B tests.

2.5.3 Group automatic playlist generation

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 [LR08, HRH07]. 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,

Table 2.4: Classification of the G-APG 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. 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 (Similarity) is applied or not.

Reference	Preference acquisition					Preferences aggregation		Playlist construction	
	Mechanics		Type		Inference	Strategy	Requests	Strategy	Similarity correction
	Stat.	Dyn.	Impl.	Expl.					
[BP07]	✓	✓	✓	✓	✓	Avg wo misery		Deterministic	✓
[CBH02]	✓		✓		✓	Avg		Stochastic	✓
[MA98]	✓			✓		Avg wo misery		Stochastic	
[HF01]		✓		✓		Avg	✓	Deterministic	✓
[AMNS02]		✓		✓		Avg		Deterministic	
[CBF05]		✓		✓		Avg wo misery		Stochastic	
[DP02]		✓		✓		Avg	✓	Stochastic	
[SWT08]		✓		✓	✓	Avg	✓	Deterministic	
[LSH20]		✓	✓		✓	Avg		Deterministic	

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 group automatic playlist generation (G-APG) algorithms.

G-APG 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 & Delic [MD22] describe the typical design of a group RS as a three step procedure, that we borrow and adapt for our discussion of G-APG 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 G-APG algorithms in terms of their preference acquisition, preference aggregation, and playlist construction mechanisms. Also, we describe strategies for evaluating G-APG algorithms. We conclude the Section by outlining the links that there are between G-APG and "standard" APG algorithms.

2.5.3.1 Preference acquisition

G-APG 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' behaviour, without any explicit actions required by the users. Examples of implicit data include listening counts, which can be acquired by inspecting log data [CBH02, LSH20]. Explicit data, instead, are filled in by the user manually. Examples of explicit data are ratings, such as a one to five rating [BP07], like/dislike [CBF05], or a vote [OLJ$^+$04] for a musical item, as well as song requests [DP02, SWT08, VA15], and pairwise ratings, indicating the preference of one song over another song [AMNS02].

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 towards a musical item if they never interacted with it. Some G-APG work uses explicit and/or implicit data to infer unknown preferences. For example, the Poolcasting system [BP07] infers unknown song preferences by taking the average of the known preferences for songs composed by the same artist. And, the Flytrap system [CBH02] 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 [SWT08] system uses song similarity based on content-based data for preference inference. Finally, the Flycasting system

[HF01] uses collaborative filtering 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 [MA98], 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 [BP07] 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 [CBF05]. 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 [BP07] 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 [BP07], and use those private preferences in a group setting. This work 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 [CNBA14].

Table 2.4 classifies the G-APG 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.

2.5.3.2 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

G-APG 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 [CBH02] 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 [MA98] avoids musical genres that have been given extremely low ratings by at least one user in the group. And, the Adaptive-radio [CBF05] 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 aggregations mechanisms, average and average without misery, are two of the many mechanisms used in group recommender systems research. For example, [MD22] show that there exist 11 preference aggregation mechanisms in group recommender systems research.

Along with the mentioned preference aggregation mechanisms, some of the G-APG 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 FIFO 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 [SWT08].

Table 2.4 classifies the G-APG work that we survey for how the preference are aggregated, and indicating whether song requests are handled or not.

2.5.3.3 Playlist construction

Once the group preferences are aggregated, the last step is to construct the playlist. The G-APG 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 [AMNS02] 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 [Cel10]. 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 [CBH02], 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 2.4 classifies the G-APG work that we survey on whether the music selection strategy is deterministic or stochastic, and indicating whether similarity correction is applied.

2.5.3.4 Evaluation

One strategy for evaluating G-APG 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, the authors of [CBF05] install the system Adaptive-radio 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. Similarly, [MA98] install the MusicFX system in a gym and poll subscribers to the system on whether the music playing in the gym has improved since the installation of MusicFX. A similar experiment is reported in [HF01], and in [PP12], where the authors compare several preference aggregation and playlist construction strategies.

Another strategy for evaluating G-APG algorithms is to use synthetic data. The use of synthetic data is common in the literature on group RS, e.g. see [MD22], but not common in the literature in G-APG. The only work that uses synthetic data is [LSH20], which gathers check-in data in a coffee-shop from foursquare[19], as well as listening histories from Last.fm[20], and analyses the accuracy of algorithms in predicting songs in the listening histories of users currently in the coffee-shop.

Finally, a conspicuous share of G-APG work that we survey (five over eight papers) does not offer any evaluation.

2.5.3.5 Relationship between G-APG and APG

The work in G-APG and APG that we survey is clearly related, not least because the two research topics share the end goal of generating a playlist automatically. We define the task of playlist generation in Definition 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. The fundamental difference between APG and G-APG resides in the target characteristics. APG algorithms handle target characteristics coming from a single user, while G-APG algorithms handle target characteristics coming from multiple users, i.e. the musical preferences of the group members, which are then aggregated to construct the playlist. The aggregation step is not present in APG algorithms, as the target characteristics are coming from a single user. Also, some G-APG algorithms support a way of giving target characteristics that is peculiar to the group setting: song requests. 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 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.

2.6 Manual playlist generation

Manual playlist generation (MPG) is a major topic within research on playlists. Research in MPG looks into how people manually construct playlists. Research in MPG is important as understanding how people manually construct playlists 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 as the problem of se-

[19]`https://foursquare.com/`.
[20]`https://last.fm`.

lecting 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 2.6.1) 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 [CBF06] 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 [BJK18, SM11]. 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 [DGF17].

In this Section, we survey the literature on MPG, presenting common target characteristics for playlists in Section 2.6.1, and criteria for selecting the songs, with a special focus on the issue of song ordering, in Section 2.6.2. Finally, in Section 2.6.3 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.

2.6.1 Target characteristics

The target characteristics of a playlist are the organisation principles which make the playlist 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 [Hag15] analyses the themes of playlists created by users of music streaming services, and provides a categorisation of those themes, employing four categories:

Standardised Themes concerning standard music organisation 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 organisation 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 being “any contextual condition that might influence the user’s perception of music” [KR12]. 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: message and puzzles, e.g. a playlists 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 example include playlists containing favourite songs, e.g. “all time favourites” or “latest favourites”.

The categorisation above is consistent with other categorisations proposed in related work, e.g. see [CJJ04, CBF06, HG09].

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 standardised 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, [CBF06] 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. And [KBJ20] suggests that playlists sometimes have more than one theme.

Few studies investigate the popularity of different themes, i.e. how frequently users create playlists of specific themes. For example, [CBF06] analyse playlist requests posted by users in blog posts and find that the majority of requests are for standardised and contextual themes. And [PZS17, PZS16] carry out a data-driven analysis of manually created playlists in Spotify, by clustering playlists on the basis of acoustic features, and analysing those clusters based on their semantics, as extracted from the most relevant song tags in the cluster. 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.

2.6.2 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 [DMV97]. Interestingly, [HG09] find that songs liked the most by users are usually picked first. Two other guidelines are song coherence/diversity and song ordering.

2.6.2.1 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, [LBM11, KBJ20] 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. Music similarity is a multi-faceted and subjective concept that we further discuss in Section 2.5.1.3. The work we review measures coherence using metadata; for example, [JKB14, SW06] use musical genres, while [PGG19, CFS15] use tags.

As well as being coherent, songs should also be diverse through the playlist [HG09, TLL17]. We use the concept of song diversity as a contrast to the concept of song coherence. For example, [LBM11] 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.

Striking the right coherence/diversity balance is key to successful song selection in playlists [KBJ20, HLW19], almost as important as matching the listener's musical taste [DMV97]. Some work tries to measure coherence/diversity in manually created playlists, with the goal of characterising the ideal trade-off between the two quantities. For example, [JKB14] utilise 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. Porcaro *et al.* [PGG19] 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 [AWMC17]. Slaney *et al.* [SW06] 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.* [CFS15] 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 trade-off 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, [LBM11] find that the appropriate trade-off between diversity and coherence depends on the preferences of the individual user [LBM11].

2.6.2.2 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 [DMV97].

There is work that investigates the relevance of song ordering via user studies, finding that song ordering is relevant. For example, [CBF06] observe the playlist creation behaviour of users, reporting that they actively arrange songs so to achieve a music flow, especially when they are creating a playlist to be shared with other listeners. And, [DMV97] explicitly ask about the importance of song ordering, and report a result of zero, on a scale from minus four to four, which we interpret as an indication that song ordering has some relevance. The authors also ask participants in their study about the importance of other factors, such as the start and end songs, rated as minus two, choosing songs according to taste, rated as two, and balancing coherence and diversity, also rated as two.

However, other work that also employs user studies finds that song ordering is not relevant. For example, [AH05] ask the 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. They 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, [TLL17] find that users do not expect playlists to be ordered in the first place. Finally, [KBJ20] ask about the importance of song ordering and six other factors, finding that song ordering is the second least relevant factor.

Some work in APG also indirectly studies the relevance of song ordering. For example, [ABC+18] 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. This is an indication that song ordering is relevant. But, [VQSW19] instead find that song ordering is not relevant. 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 [VQSW19] is presented in [ML12].

Some work investigates the importance of song ordering via statistical analysis. For example, [SPCS21] consider a set of manually constructed playlists, and extract a variety of song features, including content-based data and metadata. The content-based data are: acousticness, danceability, energy, instrumentalness, liveness, loudness, speechiness, tempo, valence, key, and mode. The metadata are: 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 standardised experimental procedure. For example, [SPCS21] 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. 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 [CBF06]. Looking at the studies above, those that deal with playlists to be shared find that song ordering is important, e.g. [SPCS21, ABC$^+$18, DMV97], while those that deal with playlists for private use find that song ordering is not important, e.g. [TLL17, KBJ20, AH05]. 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 [CBF06]. The research of Leong *et al.* [LVH05, LVH06, LHV08] 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.

Songs 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.* [DGF17] 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, e.g. two parts of the same continuous recording [CBF06, CBB17]; Rule (2): place songs with complementary sounds and styles together consecutively, so as to avoid the “clash of one song against the other”.

2.6.3 Assisted manual playlist generation

Assisted manual playlist generation (A-MPG) is a special case of MPG, concerned with the development and evaluation of algorithms for assisting users

Table 2.5: Organisation of work on A-MPG based on how the user is assisted during the playlist construction process. We distinguish work that assists users with visualisations and with recommendations. Additionally, we divide work published up to 2016 from work published from 2016 onwards.

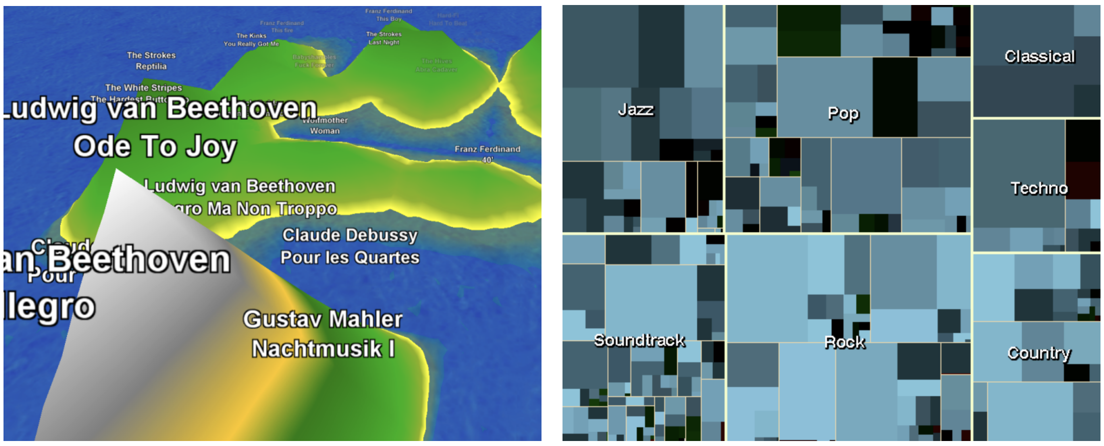
Category	Up to 2016	From 2017
Visualisation	[PRM02, THA04, MUNS05, NDR05, KSPW06, LT07, Lil08, GKW08, CB09, DF10, MRNT10, CVER07, GCS11, vGVvdW04, JJ09, VGV05, CPP11, Voo06, LE07, HAS09]	—
Recommendation	[GG05, BBB10, BHBB11, DGF16, HHKB06]	[KBJ20, KJB18, PHG$^+$17, MHCV19]

in the process of manual playlist construction. An example of work in A-MPG is [KBJ20]. The authors utilise 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. The authors also analyse the impact of these song recommendations in the playlist construction behaviour of users.

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 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 [Web20]. 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 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[21]. A-MPG unites the two paradigms, leaving users in control of the playlist construction process,

[21]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)

Figure 2.2: Examples of visualisation used in A-MPG to assist users in manually creating playlists. Figure (a) is from [KSPW06] and Figure (b) is from [THA04].

while alleviating users of the burden of manually searching enormous song catalogues. A-MPG thus strives to achieve a trade-off 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 fast, but may lead to low user satisfaction with the song choices [Kju16]; 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 [BBB10].

In their 2017 survey, Dias *et al.* [DGF17] review the research on A-MPG to that date. They present several categories of algorithms, all based on visualisations: maps, graphs, dots and radar A-MPG algorithms. In our survey here we include the A-MPG algorithms that were covered by survey (2), as well as more recent work, not covered in survey (2). However, the recent work does not belong to the categories proposed in survey (2). Hence, we propose a novel categorisation of A-MPG algorithms, to cover both recent and non-recent work. Specifically, we divide algorithms in two categories: visualisation, and recommendation. In Table 2.5 we divide the work on A-MPG based on the categorisation, and we distinguish the research up to 2016, which was already surveyed by Dias *et. al*, from the research from 2017 onwards, that we exclusively survey.

2.6.3.1 Visualisation

Some work in A-MPG use visualisations for assisting users in the manual construction of playlists. An example of visualisations 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 other. Maps can either be 2D, e.g. [PRM02, JJ09], or 3D, e.g. [KSPW06, LE07]. In Figure 2.2, we show examples of maps from two A-MPG algorithms.

Figure 2.2(a) is a map proposed in [KSPW06]. The authors represent the songs in a multi-dimensional space made of multiple types of content-based data. Then, they use self-organising maps (SOMs) [Koh90] to project the songs to a two-dimensional 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 [PRM02]. Similar work that uses SOMs includes [NDR05, LT07, MUNS05, MRNT10, vGVvdW04, Voo06, JJ09].

Figure 2.2(b) is a map proposed in [THA04]. 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 [Shn92] 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 2.2(b) shows a map drawn at the level of genres. Another work that uses treemaps is [DF10].

Visualisations can also consist of graphs. For example, [GCS11] create a graph of artists, using similarity between artists, as retrieved from Last.fm, which is known to rely on usage data [CKN$^+$11]. Artist-to-artist similarity is symbolised as edges, with longer edges joining less similar artists, and shorter edges joining more similar artists. A similar work using graphs is [CVER07].

Visualisations can be drawn using simpler strategies, such as associating songs to points in the valence-arousal space [Gre16], without applying other transformations [CPP11], or by associating songs to colours [HAS09], and drawing a colour-map representing a collection of songs.

The visualisations 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, [GKW08] relies on usage data. Inside Section 2.5.1.3, we describes other ways in which music similarity can be measured. Some visualisations leave the user in control to decide their preferred background knowledge for computing some similarity. For example, [vGVvdW04, VGV05] 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 with the control of deciding which sources of background knowledge to use.

Once the visualisation is created, the user can interact with the visualisation while creating a playlist. For example, in the case of maps, different types of interactions are possible, including manual song picking in the map [THA04], as well as drawing paths [Lil08, JJ09] or shapes [NDR05] in order to create a playlist of the songs included on the path or shape.

2.6.3.2 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 2.5.1 can be used for providing those recommendations. For example, [KBJ20, KJB18] use the APG feature that is part of the Spotify API and an heuristic algorithm based on popularity, while [GG05] uses a similarity-based algorithm.

The recommendations are usually displayed as elements of the user interface. Standard approaches include [KBJ20, KJB18], which present recommendations as a list of songs that can be clicked on. A more creative approach is in [GG05], 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* [BBB10]. *Rush* asks the user to select a seed song, and it then presents the user with a carousel of five songs that it recommends 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* [BHBB11], 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 customise *Rush 2*'s recommendations by setting filters on BPMs and genre. *AudiRadar* is a system that is similar to *Rush 2* [HHKB06]. *MixTape* [PHG$^+$17] 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 recommendations, and it exploits similarity when the user accepts a song recommendations. *PlaylistCreator* [DGF16] displays recommendations using a standard list strategy, but the interface allows users to customise 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, [MHCV19] provide explanations based on the visualisation of content-based features, and find that the presence of explanations increases the satisfaction of users with the process of playlist construction.

Kamehkhosh *et al.* [KBJ20, KJB18] 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 ten 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.

2.6.3.3 Discussion

Table 2.5 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 onwards (the *second period*), that we exclusively survey. Research from the *first period* focuses mainly on visualising 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 [Spo22], which are difficult to visualise on a map or on a graph. Recommendations, on the other hand, can help users by providing a small selection of relevant songs.

Evaluation of A-MPG studies is primarily done through user studies. One fundamental issue is that of evaluating the usability of the A-MPG algorithm, which can be done using standard questionnaires, common in the HCI field, e.g. see [Lew95], or through semi-structured interviews. Other work investigates domain-specific aspects, for example satisfaction with constructed playlists when assisted by the A-MPG algorithm [BBB10]. Finally, some works in A-MPG do not carry out any experimental evaluation e.g. [CPP11, GKW08].

2.6.4 Group manual playlist generation

Research in group manual playlist generation (G-MPG) 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. In Section 2.6, 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. G-MPG 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, [PK21] report that the majority of CPs have a maximum of six collaborators. In the same study, [PK21] report 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.* [SLPL18]. CPs can be listened to in a group, which can be composed of the people that contributed to the playlist, or may include other people, such as other people invited to a party [CN09]. In a large proportion of cases, CPs are also listened to by individuals on their own, for example to accompany everyday activities [PK21] CPs are very dynamic as it is common for the collaborators to frequently update a CP [LO13]. For example, during a party, a playlist could be modified so as to accommodate song requests from the guests [SLPL18].

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 a recent study, Park & Kaneshiro report that more than half of Spotify users in the US collaborate on CPs [PK22].

CPs serve a number of important purposes. Park *et al.* [PLLK19] 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. Some of the purpose of CPs align with the purposes of PPs, which are convenience and self-expression (see Section 2.2.1). In particular, the practical and social purposes of CPs align with the convenience and self-expression purposes of PPs. Several other work corroborate the three categories of CP purposes proposed in [PLLK19]. For example, [PK21] find that users use CPs as a means of discovering new music added by the other collaborators, and that these social recommendations are welcome by users because of the social link with the collaborator that recommended them. In a similar vein, [LOV17, LOT16, SLPL18] report that social recommendations are more appreciated than algorithmic recommendations. And, [PK21, PRBK22] find that CPs facilitate the reinforcement of social connections, because music is often used as a medium for discovering the personality of other individuals [LDHL12], and can spark conversations among collaborators [BMA06]. Similarly, [LDHL12] find that CPs can be used to keep in touch, for example using songs that evoke shared memories, acting a little like souvenirs.

In the rest of this Section, we continue our characterisation of CPs. In Sections 2.6.4.1 and 2.6.4.2 we talk, respectively, about target characteristics and song selection strategies in CPs. In Section 2.6.4.3, we talk about the group dynamics that arise in the construction of CPs.

2.6.4.1 Target characteristics

The target characteristics of a playlist are the organisation principles which make the playlist a sequence of songs to be listened to together. In Section 2.6.1, we make use of the word “theme” to refer to the target characteristics of a manually constructed playlist, and we analysed the themes of PPs. In this Section, we make the same use of the word “theme”, as we analyse the themes of CPs.

The themes of CPs mostly overlap with the themes of PPs, which we covered in Section 2.6.1. 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 [BJK18], or a road trip [CNBA14, AÖ02]. 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 [BJK18]. 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 [PK21, PK22], and whether the playlist matches a desired listening context [LR08]. The active engagement of the collaborators with a CP determines its success [PK21], 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 2.6.1, 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.

2.6.4.2 Song selection

In Section 2.6.2, we analyse 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 [PK21]. 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 group of 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 group of 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 [SLPL18], 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 [CN09], and/or by sorting songs, for example by musical genre or by artist name [PK22].

2.6.4.3 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.

2.6.4.3.1 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 [CN09]. 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 [CN09].

Since musical taste is linked to the personality of an individual, sharing music involves some revelation of the personality of an individual [LDHL12], 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 [CNBA14].

When collaborators do add songs to a playlist, they do so consciously, trying to fit the theme of the playlist and the taste of the other collaborators and any other intended listeners [LOV17, LOT16]. Bauer & Ferwerda [BF20] 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, [PL21] measure the comfort of a user in performing different tasks, namely adding, deleting, and re-ordering songs in a CP. 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 [PK21], 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.

2.6.4.3.2 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 subscribers 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 authorised users. Some works propose going a step forward, by developing a range of social features in the context of CPs: [LR08, PK22, LO13] propose to log modifications to the playlists, such as song additions, deletions and re-orderings, as well as logging the author of the modification; [PL21, PK22, LO13] propose 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 socialise; and [PK22] propose 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.

Some work reports that users are in favour of including these social functionalities in streaming services. For example, [PL21] report that the chat functionality would increase the comfort of users in adding, deleting and re-ordering songs. But, other work reports that these social functionalities should be mindful of user privacy. For example, [Anb18, BMA06] report that 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, 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 [PK22]. Some more fine-grained levels of access control are also discussed; for example, [LOV17, LOT16] propose that the playlist creator should have the opportunity to approve any changes to the playlist. 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 manual construction and automatic construction. In the remainder of the survey, we focus on other topics, starting with the construction of enhanced playlists, i.e. objects that generalise playlists, by including special features.

2.7 Enhanced playlists

In Section 2.2, we define 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 two types of enhancements: song mixing and interleaving songs with speech.

2.7.1 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, [SC12] propose a method to classify "good" song-to-song transitions from "bad" song-to-song transitions. 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 the 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 [RM87] and on psychoacoustic studies [GDS16]. Ishizaki *et al.* [IHT09] 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 minimised. 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 [Cli00] and [BGH$^+$17]. The latter work proposes a sophisticated algorithm for determining the transition points, i.e. the points that mark where the two tracks 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 penalised; 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 [GMS10]. Basu [Bas04] 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 [Bas04, BGH$^+$17, IHT09].

2.7.2 Interleaving songs with speech

In playlists, one song follows another, from the first song to the last song. Playlists therefore do not facilitate access to music contextual information while listening[22]. However, studies report that information seeking is one of the most important motivations for listening to music [LD04], 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 [COWH20]. 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 analysing real radio programs, and developing tools and knowledge that can help to replicate those radio programs. For example, [JLT14] 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 recognise 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.* [LJ16] 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.* [BMT$^+$19] propose an algorithm for generating small pieces of text, called *segues*, for connecting one song to the next one in a playlist.

[22]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[23]. 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.

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. Our work on song-level intelligibility, which we describe in this dissertation, is related to that of Behrooz *et al.* [BMT+19]. For example, in Chapter 3 we define a scoring function able to determine the interestingness of any segue. In the same Chapter, we validate our measure of interestingness, finding that it positively correlates with human perceptions of segue quality. In Chapter 4, we provide several algorithms to generate music tours, with the objective being to maximise the interestingness of the segues in the tour [GB21c, GB22], and we evaluate those algorithms with offline experiments. In Chapter 5, we provide a user-centered investigation of tours.

2.8 Playlist tagging & captioning

Music streaming services feature billions of playlists created by users, professional editors and algorithms [Dea21]. In this content overload scenario, it is crucial to characterise playlists, so that music can be effectively organised and accessed [CFM+16]. Two common ways of characterising playlists are via automatic playlist tagging and automatic playlist captioning. We review both of these below.

2.8.1 Playlist tagging

A common approach for characterising playlists is tagging, which is the task of assigning to a playlist one or more tags[24]. For example, [FKL+21] describe a dataset of playlists annotated with a variety of different tags, such as musical genres and decades. Similarly, [CKE20] describe a dataset of playlists annotated with listening context tags. Examples of listening context tags are “workout” and “party”, which characterise 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.* [CKE20] set up a multi-label classification problem, in which playlists are classified for their listening contexts, proposing four classifiers: two matrix factorisation (MF)-based classifiers, that work by counting how many times a song is associated with each listening context, and two convolutional neural network

[24]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.

(CNN)-based classifiers, that work with song audio. Our work on playlist level intelligibility, which we describe in this dissertation, is related to that of Choi *et al.* [CKE20]. In Chapter 6, in fact, we propose four other classifiers, that integrate metadata in the form of a knowledge graph, reporting state-of-the-art accuracy.

2.8.2 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 2.6.1, 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 characterisation of playlists at a high semantic level.

Playlist captioning is introduced in [CFM$^+$16] as the task of automatically describing a playlist using natural language. In the same paper, [CFM$^+$16] propose to use a sequence-to-sequence (seq2seq) model based on a RNN, similar to those common in machine translation [BCB15], adapted to translate a playlist, as represented by song embeddings, to a caption. The song embeddings are extracted from song audio using a CNN. They benchmark their model on a small dataset of captioned playlists, reporting that their model fails to generalise to new playlists. Similarly, Doh *et al.* [DLN21] use a RNN and a transformer model [VSP$^+$17] to translate a playlist, as represented by song embeddings, to a caption. In their case, they rely on learned embeddings, and they use a larger dataset. Our work on playlist level intelligibility, which we describe in this dissertation, is related to the above work on playlist captioning. In Chapter 7, in fact, we also use a transformer architecture for playlist captioning. In our case, the song embeddings capture musical knowledge from song audio and tags. Additionally, we utilise linguistic knowledge held by the GPT-2 language model [BMR$^+$20], reporting state-of-the-art performance in the captioning task.

2.9 Conclusion

In this Chapter, we provided a comprehensive survey of MIR research on music playlists. The following Chapters are largely about song-level intelligibility, strongly related to Section 2.7.2 of the survey, and on playlist-level intelligibility, strongly related to Section 2.8 of the survey. More generally, with this Chapter we provide a useful framework for understanding this dissertation in the context of a broad selection of related research.

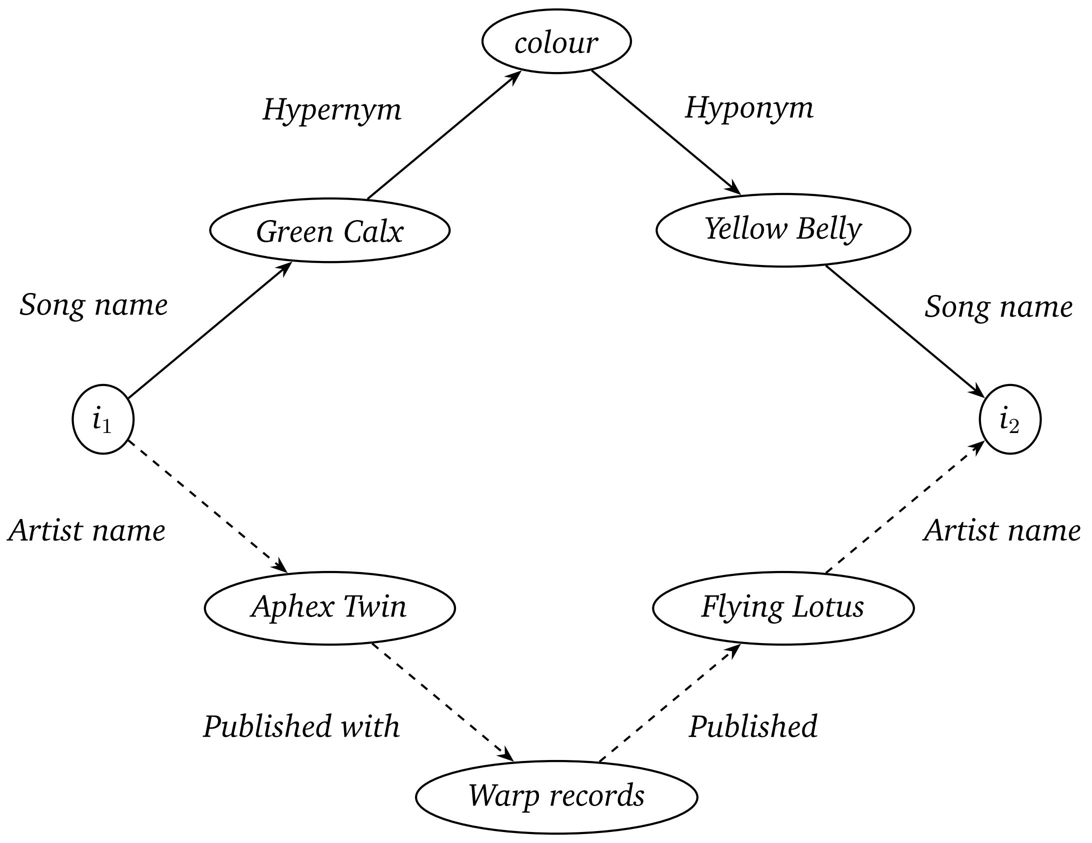
Chapter 3

Generating song-to-song segues

3.1 Introduction

The focus of this dissertation is on playlist intelligibility, that is the degree to which playlists can be understood by a human audience. As explained in Chapter 1, we distinguish two levels of intelligibility: song-level and playlist-level. In this chapter, and in the next two Chapters, we focus on song-level intelligibility, that is the degree to which transitions between consecutive songs can be understood by a human audience. In particular, we achieve song-level intelligibility by developing algorithms that can generate song-level textual annotations, or segues. In this Chapter, we propose and evaluate an algorithm, called DAVE, that, given two songs, can generate segues between the two songs. A distinguishing feature of DAVE is its ability to highlight interesting segues, according to a novel theory of interestingness. DAVE assumes a knowledge graph as an abstract representation for songs and information about songs. In our abstraction, segues are paths from one item to another, and *interestingness* is a scoring function for paths. We 'get back' from the abstraction by mapping paths to texts. DAVE can generate song-to-song segues of 1553 different types. Segue types range from factual to word-play. See Figure 3.1 for examples.

We evaluate DAVE qualitatively by means of a user trial, where we compare DAVE's segues against curated segues from a segment of the Radcliff & Maconie Show on BBC Radio 6 called THE CHAIN. In the case of factual segues, we find that DAVE can produce segues of the same quality, if not better, than those to be found in THE CHAIN. The results highlight the validity of DAVE as a method for generating song-to-song segues. We release the source code and the dataset,



Solid path: "From green to yellow".

Dashed path: "Aphex Twin recorded for Warp, and so did Flying Lotus".

Figure 3.1: Examples of two segues from the song "Green Clax" by "Aphex Twin" to the song "Yellow Belly" by "Flying Lotus". The two segues are: (1): "From green to yellow"; and (2)"Aphex twin recorded for Warp, and so did Flying Lotus". We also report a representation of segues in the form of paths in a knowledge graph. Segue (1) is represented by the solid path, and segue (2) is represented by the dashed path.

consisting of the answers gathered during the user trial, to facilitate future research on the subject[1].

Segues were first introduced in [BMT$^+$19]. There, the authors developed a simple prototype able to generate segues for consecutive songs in playlists. They score individual segues simply with static segue preference weights but they also take into account, for example, the position of the segue in the playlist and its proximity to segues of the same type, in order to obtain a degree of segue diversity. Their experimental procedure consisted of unstructured interviews, aimed at exploring the potential of the idea. Our contribution fills a number of

[1]`https://github.com/GiovanniGabbolini/dave`

Procedure 1: $find\_segue(G, s_1, s_2)$

```
input : knowledge graph $G$, songs $s_1$ and $s_2$
output: interesting segue
1 paths = $find\_paths(G, s_1, s_2)$
2 for path in paths do
3     path.score = $interestingness$(path)
4 end
5 best_path = path with highest score
6 best_segue = $path\_to\_text$(path)
7 return best_segue
```

gaps in this previous work. We develop *interestingness*, a scoring mechanism for individual segues. And, we evaluate our system with a larger user trial, comparing it with a curated baseline.

The remainder of this Chapter is organised as follows: Section 3.2 explains how DAVE works. Section 3.3 details the experimental procedure and analyses the results. Section 3.4 presents some conclusions.

3.2 Method

Our goal is to generate interesting song-to-song segues. That is, we aim for interestingness, and we discourage trivial and boring segues. We are interested in generating factual connections, but also amusing ones, based on simple word-play.

3.2.1 Working mechanism of DAVE

DAVE uses a knowledge graph G as an abstract representation for items and information about those items. In our abstraction, segues are paths from an item to another item, and interestingness is a scoring function from paths to numbers. We 'get back' from the abstraction through another function, which maps paths to texts. In particular, the procedure for finding a segue from a song s_1 to another song s_2 works as follows: we find the paths in G that connect s_1 to s_2, we score them based on their interestingness, keeping only the best one, which is translated to text, and finally returned. This approach allows us to exploit heterogeneous data [FŞA+20]. The procedure is also summarized as Procedure 1.

In the following, we will provide more details about Procedure 1. We start by presenting some preliminary concepts:

A **knowledge graph** is a set of triples $G = \{(e, r, e') \,|\, e, e' \in E, r \in R\}$, where E and R denote, respectively, the sets of entities and relationships. A special subset of entities $S \subseteq E$, are the songs. Every entity has a *type* and a *value*. For example, an entity that represents a song has *type* equal to "song" and a *value* equal to the song URI. Every relationship has a *type*.

A **path** p in G is an ordered list of entities and relationships in G, $p = [e_1, r_1, ..., r_{n-1}, e_n]$ where each triple in p must be in G. The *type* of p is the ordered concatenation of the entity and relationship types in p.

We turn to the components of Procedure 1, namely *find_paths*, *path_to_text* and *interestingness*. *find_paths* is a path-finding procedure that returns all the simple paths, i.e. those without cycles, starting from the item s_1 and reaching the item s_2 in G . However, we allow for the possibility of constraining the paths that *find_paths* can find; see Section 3.2.2.2. *path_to_text* works by template filling, with canned templates indexed by the path type, and completed with information on the entities and relationships that constitute it. Our focus for this work was mainly on *interestingness* and so we devote the remainder of this section to explaining our definition.

Defining a scoring function for segues based on interestingness is not a trivial task. To be concrete, we refer to Figure 3.1: which one of the two segues is more interesting? There is no correct answer to this question, as interestingness can depend upon personal relatedness [Sch79] and background knowledge [Kin80]. In this context, we develop a simple theory of interestingness according to which a ranking can be determined. Our theory builds upon the concepts of infrequency and conciseness. We believe that infrequent segues are more interesting, as pointed out by Schank [Sch79] and Kintsch [Kin80] when discussing interestingess of general statements. We also believe that concise segues are more interesting, as supported by Geng *et al.* [GH06] in the context of interestingness in data mining. Our definition of interestingness applies to paths in knowledge graphs and consists of three heuristics. The heuristics rely only on statistical information and simple content descriptors that do not depend on the semantics of segues.

Rarity We define the rarity of a path p using the proportion of paths in G that have the same type as p. To formalize this, let T be the set of all path types in G; and let $f(t)$ be the number of paths in G that are of type t. Then,

$$rarity(p) = 1 - \frac{f(type(p))}{\max_{t \in T} f(t)}$$

Unpopularity We define the unpopularity of a path p using the notion of centrality of an entity e. An entity is central if it has a high number of incoming and outgoing edges, compared with other entities of the same type. A path that visits central entities is more popular than one that does not. To formalize this, let $edgeset(e)$ be the set of incoming and outgoing edges to and from an entity $e \in E$ in G. We define the centrality of an entity e as:

$$centrality(e) = \min\left(1, \frac{|edgeset(e)|}{\underset{e' \in E}{median}|edgeset(e')|}\right), \; type(e') = type(e)$$

Then, we define the *unpopularity* of a path p as:

$$unpopularity(p) = 1 - \min_{e \in p \cap E}(centrality(e))$$

Shortness Let the *length* of a path p in G be:

$$length(p) = |p \cap R|$$

We define the *shortness* of a path p in G as done in [AMHWA$^+$05]:

$$shortness(p) = \frac{1}{length(p)}$$

The heuristics *rarity* and *unpopularity* both implement the idea of favouring infrequent segues, but in a different fashion: *rarity* has high values for infrequent path types, while *unpopularity* has high values for paths that include infrequent entities. For example, a path that connects people who share a birthday is likely to have a fairly high value for *rarity*, but not necessarily for *unpopularity*, e.g. if both people are very famous. Or, paths that connect people who have the same hair colour will have a low value for *rarity*, but can have a high value for *unpopularity*, e.g. if the shared hair colour is something unusual such as "green". The *shortness* heuristic implements the concept of conciseness.

We define the *interestingness* of a path p in G as the convex combination of the three heuristics:

$$interestingness(p) = w_1 rarity(p) + w_2 unpopularity(p) + w_3 shortness(p)$$

It ranges from zero to one. w_1, w_2, w_3 are parameters to be tuned, subject to $w_1 + w_2 + w_3 = 1$.

3.2.2 Implementation

We implement Procedure 1 as described in Section 3.2.1. The implementation of Procedure 1 is the algorithm we propose: DAVE.

3.2.2.1 Knowledge graph

We represent a song with three fields: *song name*, *artist name* and *album name*. The representation allows missing values, e.g. a song that is not part of an album, and can be easily changed with only minor modifications to the rest of our implementation, e.g. if songs were instead represented by their Spotify URIs.

DAVE uses a knowledge graph with 40 distinct node types and 230 distinct edge types, and can provide segues of 1153 different path types. We build the knowledge graph with data that we harvest from three main resources:

MusicBrainz We use MusicBrainz[2] as the main source of factual data. We exploit the MusicBrainz APIs[3]. They allow us to navigate the MusicBrainz database, and offer entity-linking functionalities. In a first step, we link the actual song, its album and its artist to their respective MusicBrainz URIs. Then, we mine different sorts of factual data, ranging from the genres of the song to the birth places of the artists.

Wikidata We use Wikidata[4] as an additional source of factual data. There exists a mapping from MusicBrainz URIs to Wikidata URIs, making it easy to use both resources. From Wikidata, we mine biographical data about artists that is not available in MusicBrainz, e.g. the awards that an artist has won.

WordNet We use WordNet [Mil98] to mine lexical data about the words in song, artist and album names, e.g. hypernyms and hyponyms.

[2]`https://musicbrainz.org/`
[3]`https://python-musicbrainzngs.readthedocs.io/en/v0.7.1/`
[4]`https://wikidata.org/`

In addition, we gather some further lexical data through a number of different resources. For example, we link words in song, artist and album names to their stems using the Porter stemmer [Por80], and to their phonetics with the NRL algorithm [EJMS76]. We provide a complete description of entities and relationships that build up the knowledge graph in the additional materials[5].

The factual data that we obtain from MusicBrainz and Wikidata allows for conventional, informative segues. On the other hand, the lexical data from WordNet and other resources yields less conventional and perhaps amusing connections based on word-play. We show two examples in Figure 3.1.

3.2.2.2 Procedure

As mentioned in Section 3.2.1, we constrain the $find\_paths$ component of Procedure 1; specifically, we constrain it to find only paths that go from a start song to another song without visiting another song. This constraint limits the number of paths to be scored, at the price of losing some indirect paths. We believe that some of these indirect paths would be filtered out by $interestingness$ anyway, since they would have low values for $shortness$. We set the weights to be used in the $interestingness$ score to $w_1 = 0.4, w_2 = 0.2, w_3 = 0.4$. The weights were set after empirical experimentation, using songs that were not used in the user trial. We discuss other ways to set the weights in Chapter 8.

3.3 Experiments

We evaluate DAVE by means of a user trial, where we compare DAVE's segues against a curated source of segues called THE CHAIN. THE CHAIN is a segment of the Radcliffe & Maconie Show on BBC Radio 6. In this segment of the show, listeners call in and propose the next song, always making sure that there is a connection (sometimes informative, sometimes funny) between the previous song and the next one. THE CHAIN is made for entertainment and therefore offers very interesting segues, with strong creative traits. A database with all the segues that have appeared in THE CHAIN is available on the internet[6]. At the time of writing, it comprises more than 8000 segues. A direct comparison with curated segues is our means of assessing the quality of segues from DAVE. We do not include any other algorithmic baseline, as we are not aware of any

[5]`https://doi.org/10.5281/zenodo.4619395`
[6]`https://www.thechain.uk`

other similar systems in the literature. The only work that deals with segues is [BMT+19]. We cannot compare with their work because their scoring mechanism is presented in insufficient detail to be reproducible.

Our user trial is a within-subject trial with two treatments: DAVE and THE CHAIN. We generate segues to show in each treatment using the following procedure. One segue is sampled at random from a fixed random sub-sample of THE CHAIN, of cardinality equal to 200 segues. The corresponding segue is generated by DAVE by picking the segue that maximizes the $interestingness$, when starting from the same song as THE CHAIN, and going towards a song from a fixed sample of songs, of cardinality equal to 496 songs. These 496 songs are a random sub-sample of 20000 songs from the RecSys Challenge 2018 Dataset [CLSZ18]. We use MusicBrainz as a source of artist data to filter out artists based on the country in which they were born or in which they are based, and on the musical genres with which they are associated. Specifically, we keep only artists born or based in the UK, and who are associated with at least one genre from the following: blues, blues-rock, pop, pop-rock, rock, soft-rock, funk, jazz, r&b and soul. This country and these genres are ones that predominate in THE CHAIN. This filtering ensure that the songs that DAVE can choose from are ones that match the style of songs found in THE CHAIN, thus mitigating one confounder from the user trial.

Each participant is asked to evaluate six segues: she undergoes both treatments three times. The order of treatments in each pair is randomized. We make sure that the same segues are not shown multiple times to a participant.

In the following, we provide details on the user trial design, we present statistics on the answers, and, finally, we analyze the results in depth.

3.3.1 User trial design

The user trial begins with an instructions page. It continues with a three-part survey, whose parts we will refer to as intro, segue evaluation and outro. The trial concludes with a final page that offers an optional comments box. In the following, we describe the three parts of the survey. We provide screenshots of the user trial text and its workflow in the additional materials[7].

[7]`https://doi.org/10.5281/zenodo.4619395`

3.3.1.1 Intro

In the intro, we ask each participant some questions to identify how much she engages with music. A previous study has highlighted that segues might be especially suited for music "nerds" [BMT+19]. By asking these questions in our trial, we can see whether music engagement correlates with segue appreciation, for example. The source of the questions that we ask is the Goldsmiths Musical Sophistication Index (Gold-MSI) [MGMS14]. The index comprises five aspects. Four of the aspects are concerned with music skills, such as musical training and singing abilities. We restrict ourselves to the remaining aspect, the one called Active Engagement (AE). AE covers "a range of active musical engagement behaviours (e.g. I often read or search the internet for things related to music) as well as the deliberate allocation of time and money on musical activities (e.g. I listen attentively to music for n hours per day)" [MGMS14]. We follow [MGMS14], and we measure AE by asking ten questions, with answers on a seven point scale.

We are also interested in English proficiency, as we want to make sure that participants can properly understand the segues. So in the intro we also ask the participant to identify her level of proficiency from among four options: "Low", "Mid", "High" and "Mother Tongue". We do not collect demographic information, as we considered it not essential for the scope of the study.

3.3.1.2 Segue evaluation

In the segue evaluation phase, each participant is asked to evaluate six segues, as described in Section 3.3. We show the segues, as well as the titles and artists of the songs that they connect. We do not provide any means for the user to listen to the songs. For every segue, we ask questions to measure a variety of quality metrics. First, we ask whether the segue is *likeable*, of *high-quality*, and whether it *sparked-interest* in the next song. Second, we asked how the participant perceived its content, on three dimensions: *informative*, *funny* and *creative*. Lastly, we wanted to make sure that the connection between the two songs is expressed in an *understandable* way by the segue, and that the segue is *well-written*. We call these three groups of dependent variables respectively: *valence*, *content* and *text* quality metrics. The dependent variables introduced in this part of the survey are summarized in Table 3.1. We measure all of them using five point Likert scales.

Table 3.1: Dependent variables measured during the segue evaluation part of the user trial.

name	dependent variables
valence quality metrics	*likeable*, *high-quality*, *sparked-interest*
content quality metrics	*funny*, *informative*, *creative*
text quality metrics	*understandable*, *well-written*

3.3.1.3 Outro

In the outro part of the trial, we measure the familiarity of the participant with the songs and artists involved in the segues that she has evaluated. Familiarity with songs and artists might be a confounder for the quality metrics, and we want to address these effects, if any. We consider familiarity because it has been shown to be an accurate predictor of musical choice, at least as good as liking [WGI14], and we assume there may be an extension of the results in [WGI14] to segues. We measure familiarity with the songs and artists of both the first song and second song in each segue. Familiarity is measured with a simple two point scale: “Familiar” and “Not familiar”. We decided to ask about familiarity in the outro, after the segue evaluation, so that we avoid accentuating any confounding effects.

3.3.2 Answer statistics

In total, 158 people completed the trial. They are undergraduate Computer Science students recruited in a university in Ireland. The median completion time for the survey is 7 minutes, with a maximum of 68 minutes and minimum of 2 minutes. We discard answers from people who took less than 3 minutes, as we are worried about their reliability. No participant declared their English proficiency to be “low” but we further filter out participants with “mid” English proficiency, since they also might not be able to properly evaluate segues. We are left with people with “high” proficiency and those for whom English is their mother tongue (90% of the total). After the filtering, we end up with 151 people. We convert Active Engagement answers to numbers from one to seven, and segue evaluation Likert scale answers to numbers from one to five. We also analyze the comments left by the participants, 35 comments in total, but we do not discuss them in this dissertation because those comments do not add much to our analysis.

3.3.2.1 User categories

In some of our analysis, we partition participants based on their level of Active Engagement with music (AE). We summed the answers to the AE questions given by each participant, obtaining a distribution of total AE scores. We divided participants according to the quartiles of this distribution, giving four categories: "low-AE", "mid-low-AE", "mid-high-AE" and "high-AE". We validated this partition by considering confidence intervals of segue evaluation answers: the quartiles show good internal cohesion.

3.3.2.2 Segue categories

We believe that segues can be divided into two categories: those that are intended to amuse ("funny") and those that are intended to impart information ("informative"). We decided to create a ground truth that assigns each segue to one of the two categories. Giovanni Gabbolini and Derek Bridge separately labeled every segue manually, guided by the following criterion: a segue is funny if it is written with the goal of making the listener smile, and a segue is informative if it is written with the goal of giving information to the listener. A segue can have both goals, e.g. if it presents information in a funny way. In such borderline cases, we assigned the goal that seemed dominant. We disagreed upon the labelling in 13 out of 400 cases (Cohen's $k = 0.92$). We solved divergences as follows: given a segue s_1 where there was disagreement, we found a second segue s_2 whose category was not in dispute, and that both of Gabbolini and Bridge considered to be similar to s_1. We then assigned to s_1 the category of s_2. We validate the ground truth by double-checking it against the answers to the survey. In particular, participants were asked to express how much segues were *informative* and *funny*. We computed mean values of their answers, considering separately segues labelled as informative and funny in the ground truth. We carry out a *t*-test for assessing the significance of differences in the mean values. We found that the mean for *informative* is statistically significantly higher than the mean for *funny* for segues labelled as informative (3.64 vs 2.81, $p{<}0.001$), and the opposite for segues labelled as funny (2.47 vs 2.82, $p{<}0.001$). We conclude that participants in the survey agree with our manual labeling, thus providing some support for the reliability of the ground truth.

The ground truth reveals that THE CHAIN is biased towards funny segues: roughly three out of every four of its segues are funny. DAVE is approximately balanced. We show some examples of informative and funny segues in Tables

Table 3.2: Examples of informative segues. Not only were these labeled informative in the ground truth but also at least two user trial participants gave them a maximum rating on the *informative* quality metric.

treatment	first song	segue	second song
DAVE	*Weather To Fly* by *Elbow*	And now Guy Garvey, who was a member of Elbow...	*Belly Of The Whale* by *Guy Garvey*
THE CHAIN	*505* by *Arctic Monkeys*	Bill Ryder-Jones of The Coral joined Arctic Monkeys on their last tour...	*Pass It On* by *The Coral*

Table 3.3: Examples of funny segues. Not only were these labeled funny in the ground truth but also at least two user trial participants gave them a maximum rating on the *funny* quality metric.

treatment	first song	segue	second song
DAVE	*Fleety Foot* by *Black Uhuru*	From foot to faces . . .	*Faces* by *Ed Sheeran*
THE CHAIN	*Tumbling Dice* by *The Rolling Stones*	You need a dice to play snakes and ladders...	*Rattlesnakes* by *Lloyd Cole and the Commotions*

3.2 and 3.3.

3.3.3 Results

In this section, we analyze the answers to the segue evaluation part of the survey, dividing by treatments, segue category and user category. We also investigate the effect of familiarity. And, finally, we evaluate the effectiveness of $interestingness$ and the correlation between quality metrics.

3.3.3.1 Performance in the quality metrics

We compute the average for each quality metric given in Table 3.1 within treatment (DAVE and THE CHAIN), presenting separately the results for informative segues (Table 3.4) and funny segues (Table 3.5). We conduct a *t*-test for assessing the significance of differences between the two treatments. We discuss these results below.

Table 3.4: Informative segues. Average value of quality metrics achieved in the two treatments, on a scale from one to five. Stars (*) indicate statistically significant different values of metrics between the two treatments. *: $p<0.05$; **: $p<0.01$.

	DAVE	THE CHAIN
likeable	3.26	3.20
high-quality	3.22	3.20
sparked-interest	3.06	3.13
funny	2.39	2.64*
informative	3.73**	3.43
creative	3.33	3.59*
understandable	3.84	3.69
well-written	3.46	3.33

3.3.3.1.1 Informative segues For informative segues (Table 3.4), DAVE outperforms the human-curated segues of THE CHAIN for two of the three *valence* quality metrics but the differences are not statistically significant. There are statistically significant differences on the *content* quality metrics: THE CHAIN is perceived as more *funny* and *creative* ($p<0.05$), while DAVE is more *informative* ($p<0.01$). Finally, turning to the *text* quality metrics, DAVE's segues turn out to be better written and more *understandable* than THE CHAIN's but again without statistical significance.

3.3.3.1.2 Funny segues When it comes to funny segues (Table 3.5), human-curated segues from THE CHAIN outperform DAVE's segues across all the quality metrics, with statistically significant differences. We notice low values for *funny* in both treatments. We might expect it to be higher in the category of segues we are considering. This may be due to the medium of presentation of the segues, i.e. read on a screen. We might expect better results if, for example, segues were spoken. It may also just be that the word-play humour of these segues does not appeal to the sense-of-humour of the trial participants.

3.3.3.2 Correlation of quality metrics

We compute pairwise correlations of the eight quality metrics and this is shown for all segues in Figure 3.2. There is high correlation (0.64, $p<0.001$) between both *high-quality* and *likeable* with *sparked-interest*: good segues can spark interest in the next song. We notice that *informative* has higher correlation than *funny* with all the *valence* quality metrics: a segue perceived as very informa-

Table 3.5: Funny segues. Average value of quality metrics achieved in the two treatments, on a scale from one to five. Stars (*) indicate statistically significant different values of metrics between the two treatments. *: $p<0.05$; **: $p<0.01$.

	DAVE	THE CHAIN
likeable	2.94	3.28***
high-quality	2.77	3.14***
sparked-interest	2.76	2.99*
funny	2.71	2.89*
informative	2.66	2.90**
creative	3.22	3.57***
understandable	3.58	3.78*
well-written	3.13	3.35*

tive is likely to be also perceived as very likeable, high-quality, and is more likely to spark interest in the next song. This happens to a lesser extent for segues perceived as very funny. Therefore, from a recommender systems point-of-view, where sparking interest is important, it might be more fruitful to address efforts into generating informative segues, as opposed to funny segues. However, this observation might be just due to the medium of presentation of the segues i.e. read. We do not know whether the result would generalize to other mediums, e.g. spoken.

The same happens with *understandable*, which is statistically significantly correlated with *informative* but not with *funny*. Further, *creative* has good correlation with all the *valence* quality metrics: creativity is a good asset for segues. Finally we report high correlation of *well-written* with all *valence* quality metrics (ranging from 0.52 to 0.62, $p<0.001$). This is expected, since segues are consumed in textual form. How they are written is very important, as important as the content itself.

We have also looked at these correlations in various subdivisions of the data: within treatment (DAVE or THE CHAIN), within segue category (funny or informative) and within user category (low-AE, etc.). The narrative that would accompany each of these correlation matrices is not appreciably different from the one we have given above. Hence, to save space, we do not show these more specific correlation matrices in this dissertation.

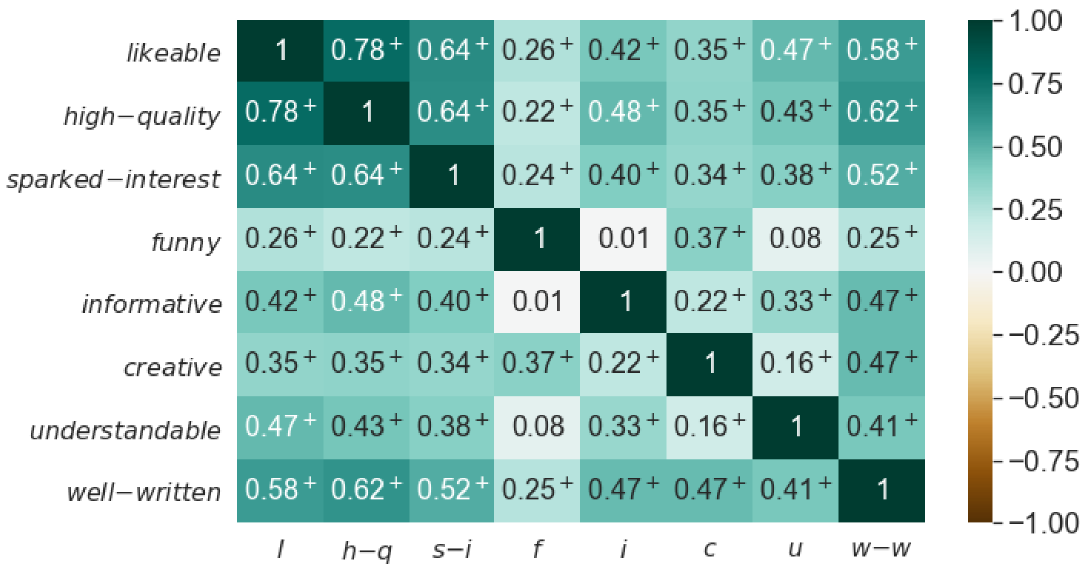


Figure 3.2: All segues. Quality metrics correlation matrix. $^{+}$: $p<0.001$, otherwise $p>0.05$

3.3.3.3 Performance in quality metrics and user category

We divide users into four groups, as explained in Section 3.3.2, and we compute the average of the answers to the quality metric questions within each user category. We conduct a statistical test for assessing the significance of differences in performance, comparing the lowest level of AE with the other three. We show the results in Table 3.6. Only one metric changes statistically significantly: *sparked-interest*. This is reasonable: higher active engagement with music is somehow correlated with a propensity for music discovery. The other main quality metrics slightly increase with AE, but the differences are not statistically significant. The results we obtain if we further divide, e.g. by treatment or by segue category, confirm those that we have presented in Table 3.6. The results contradict the intuition that segues are especially suited to "nerds" [BMT+19]. The result might change if segues were to include more musicological detail, for example, the synthesizer brand used by two artists during a recording. At present, DAVE's knowledge graph does not contain these kinds of details and so does not allow DAVE to produce segues such as these.

3.3.3.4 Performance in quality metrics and familiarity

We consider whether quality metrics are related or not to familiarity with the artists and songs involved in the segues. We first focus our attention on familiarity with songs. Then, we comment on the results we have for artists. We

Table 3.6: All segues. Average value of quality metrics, divided by level of Active Engagement (AE) with music. Values range from one to five. We conduct a significance test, comparing the lowest value of AE with the other three. Stars (*) indicate statistically significant different values of metrics between low-AE, and the other three level of AE. *: $p<0.05$; **: $p<0.01$.

	low-AE	mid-low-AE	mid-high-AE	high-AE
likeable	3.08	3.20	3.19	3.25
high-quality	3.00	3.08	3.09	3.16
sparked-interest	2.64	2.97**	3.16***	3.09***
funny	2.56	2.72	2.80*	2.66
informative	3.07	3.01	3.26	3.19
creative	3.28	3.34	3.54**	3.53**
understandable	3.63	3.71	3.72	3.86*
well-written	3.18	3.23	3.46**	3.42*

divide answers into four groups, based on whether participants are familiar or not with each of the two songs connected by the segue, and we compute the averages of each group. We conduct a statistical test for assessing the significance of differences in performance, comparing the first group (familiar with neither song) with the other three. We do not further partition by treatment, segue category or user category, as the cardinality of some of the groups is already small. We show the results in Table 3.7.

We observe that familiarity with songs, in general, leads to higher appreciation of segues. Segues are more *likeable* when connecting two familiar songs than when connecting two unfamiliar songs ($p<0.01$). And, segues are able to spark interest more when the songs are already familiar, with respect to when they are not ($p<0.001$). Moreover, they are perceived as better written ($p<0.01$), and more *understandable* ($p<0.05$).

When repeating the analysis but considering familiarity with the artists, we observe the same phenomena, but the increases in the metrics have lower magnitudes. We conclude that familiarity with artists is a weaker confounder than familiarity with songs.

3.3.3.5 *Interestingness* and quality metrics

The $\mathit{interestingness}$ function is our computational means for assessing whether a segue found by DAVE is good or not. We would like to verify whether it agrees with the human perception of quality or not. To this end, we compute the correlation of the *valence* quality metrics and $\mathit{interestingness}$ for all of DAVE's

Table 3.7: All segues. Average value of quality metrics, dividing answers based on the familiarity with the two songs connected by the segue. Values range from one to five. We conduct a significance test, comparing the first group (familiar with neither song) with the other three. Stars (*) indicate statistically significant different values of metrics between the first group, and the other three. *: $p<0.05$; **: $p<0.01$; ***: $p<0.001$.

Familiar with	neither song	just 1^{st} song	just 2^{nd} song	both songs
likeable	3.11	3.22	3.29	3.49^{**}
high-quality	3.03	3.08	3.18	3.29^{*}
sparked-interest	2.81	3.08^{*}	3.27^{***}	3.47^{***}
funny	2.65	2.64	2.79	2.78
informative	3.08	3.30^{*}	3.16	3.28
creative	3.36	3.45	3.57^{*}	3.61^{*}
understandable	3.67	3.87	3.86	3.93^{*}
well-written	3.25	3.40	3.41	3.61^{**}

segues that were used in the user trial.

We find that there is a statistically significant correlation between the *valence* quality metrics and $interestingness$ for informative segues. This indicates that the $interestingness$ function, without being aware of semantics, relying only on statistical information and simple content descriptors, can successfully rate the quality of informative segues: on average, segues rated low by the trial participants are low also in $interestingness$, and vice versa. We believe that this is a very good result, given the complexity of the task. We observe worse results with funny segues, where we do not find any statistically significant correlations: deeper considerations might be needed, e.g. the role of semantics. We also compute the correlation with *content* quality metrics, but we do not find any strong statistically significant correlations. This is expected, since $interestingness$ is independent from the semantics of the segues.

Finally, we turn to the correlation with *text* quality metrics. We do not find any correlation between $interestingness$ and *well-written*. This is as expected, since *well-written* does not depend directly on $interestingness$, but on $path\_to\_text$. But, we do find statistically significant correlation in informative segues for *understandable* (0.22, $p<0.001$). This is an indication that $interestingness$, even though it is built around the concept of infrequency, does not favour obscure segues.

We report all the results in Table 3.8.

Table 3.8: DAVE's segues. Correlation of quality metrics and *interestingness* score. *: $p < 0.05$; **: $p < 0.01$; ***: $p < 0.001$.

	interestingness	
	informative segues	funny segues
likeable	0.25^{***}	0.13
high-quality	0.23^{***}	0.11
sparked-interest	0.17^{**}	0.00
funny	-0.06	-0.17^{*}
informative	0.14^{*}	0.04
creative	0.15^{*}	-0.09
understandable	0.22^{***}	0.10
well-written	0.11	0.05

3.4 Conclusion

In this Chapter we introduced an algorithm for generating song-to-song segues, called DAVE. DAVE can provide a wide variety of segues, that can be categorized as either funny or informative. The core of our method is $interestingness$, a domain-independent function for scoring the interestingness of paths in knowledge graphs.

We evaluate DAVE by means of a user trial, where we compare it against curated segues from a segment of the Radcliffe & Maconie Show on BBC Radio 6 program, called THE CHAIN. The use of THE CHAIN may be a limitation of this work. Segues from THE CHAIN have a peculiar style that fits the radio program, and are tailored to a particular kind of audience. They tend to be amusing and very creative. DAVE, on the other hand, tends to be factual, and can only deliver funny segues made of word-play. In order to alleviate such problems, we compared funny segues and informative segues from the two methods separately. Notice too that, even though THE CHAIN has a high percentage of funny segues, this does not mean that its informative segues are weak: THE CHAIN is curated by experts and draws on the considerable knowledge of thousands of BBC listeners. In any case, the user trial was intended as a method to assess how DAVE's segues compare with curated segues from a trustworthy source — segues that we can assume to be "really good". This gives a way of finding how far DAVE is from being "really good". Our goal is not to demonstrate that our algorithm can be substituted for the listeners to the show, instead, we aim to provide an evaluation in a scenario where no algorithmic baseline is available.

We find that DAVE can produce informative segues of the same quality, if not better, than THE CHAIN. We believe that this is an astonishing result, that gives an idea of the quality of our method. But, when turning to funny segues, the results are not as good. We believe that this is partially due to the *interestingness* function, as we find evidence that it is much better suited to rate the quality of informative segues. Another reason might be that funny segues from DAVE are limited to word-play, and this kind of humour does not appeal to everyone and may, in particular, not appeal to the participants in our user trial. In fact, even curated funny segues from THE CHAIN are not perceived as funny by participants in our trial. This may be a mismatch in sense of humour between listeners to the show and participants in the trial. It may also be due, in part, to the fact that segues are being read rather than being spoken. It is fair to say that, overall, the task of tackling the funny segues is only partly solved by the proposed model.

In the next two chapters, we employ DAVE as a building block for song-level intelligibility algorithms.

Chapter 4

Generating music tours

4.1 Introduction

Song-level intelligibility is the degree to which users can understand transitions between consecutive songs in playlists, see Chapter 1. The way we achieve song-level intelligibility is by interleaving songs with segues, so as to generate a tour. For example, Figure 4.1 (a) and (b) shows two tours of the same three songs.

In this Chapter, we introduce three algorithms for generating music tours from the set of songs in a playlist. In particular, we formalise the problem of tour generation and we propose three algorithms to solve the problem. Later, we offer an offline evaluation where we compare the characteristics of tours produced by those algorithms. Our offline evaluation in this Chapter does not yet take into account the user perspective. We offer a user-centered evaluation of the algorithms in Chapter 5[1].

To the best of our knowledge, there is just one other work on music tour generation: [BMT$^+$19]. Behrooz *et al.* develop a simple prototype able to find song-to-song segues for consecutive songs in an input playlist, to produce a list of segues to decorate the playlist. Our work in this Chapter is different from [BMT$^+$19], as we do not consider the order of the songs to be fixed in advance. Instead, we strive to arrange the songs, and to produce a list of segues to decorate this arrangement of the songs. We refer the reader to Chapter 2 for

[1]The user-centered evaluation of Chapter 5 involves only two of the three algorithms that we propose here. This is because the offline evaluation reported in this Chapter shows that the one of the algorithms produces tours that are extremely similar to those produced by one of the other two algorithms; see Section 4.3.4.

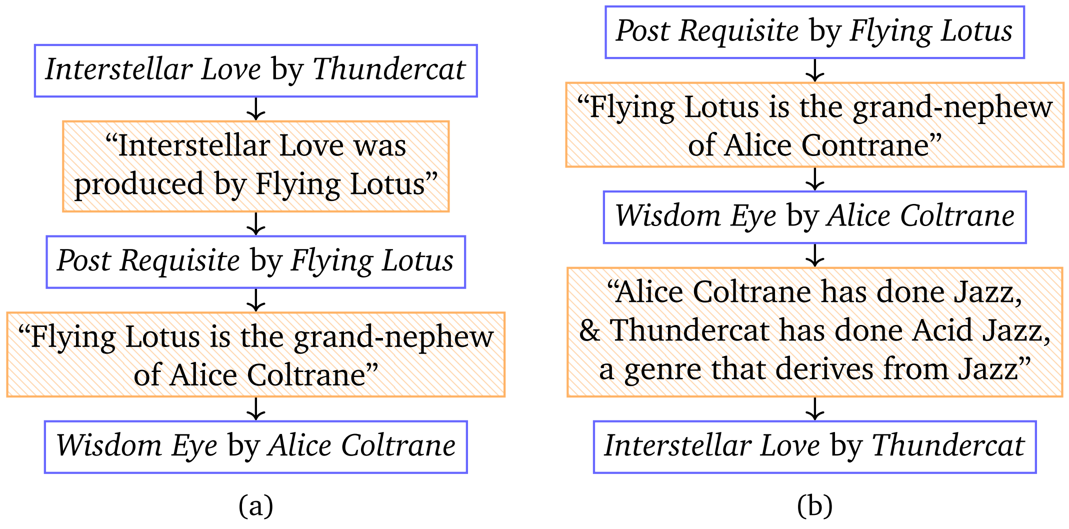

Interstellar Love by Thundercat
"Interstellar Love was produced by Flying Lotus"
Post Requisite by Flying Lotus
"Flying Lotus is the grand-nephew of Alice Coltrane"
Wisdom Eye by Alice Coltrane
(a)
Post Requisite by Flying Lotus
"Flying Lotus is the grand-nephew of Alice Contrane"
Wisdom Eye by Alice Coltrane
"Alice Coltrane has done Jazz, & Thundercat has done Acid Jazz, a genre that derives from Jazz"
Interstellar Love by Thundercat
(b)

Figure 4.1: Two tours of the same three set of songs: *Interstellar Love* by *Thundercat*, *Post Requisite* by *Flying Lotus* and *Wisdom Eye* by *Alice Coltrane*.

more details on the research on tours, and for a broader understanding of how research on tours relates to other MIR research on playlists.

The source code supporting this Chapter is freely available[2].

4.2 Method

4.2.1 Problem formulation

Let I be the set of songs in a given playlist. It is reasonable to assume I to be a 'small' set of songs, e.g. several order of magnitudes smaller than the whole catalogue of songs. Our goal is to find a tour of the songs in I, so as to obtain a solution of the kind shown in Figure 4.1.

More formally, given two songs $i, i' \in I$, let $segues(i, i')$ be the set of segues from i to i'. We assume that $segues(i, i')$ always contains at least one segue, as it is always possible to find a *null* segue [BMT+19]. Let L_j be the jth element in a list L, given that $j \in \{0, \ldots, |L| - 1\}$.

Let $\mathcal{P}(I)$ be the permutations of set I. Then let $\mathcal{T}(I)$ be decorated permutations of I:

$$\mathcal{T}(I) = \{\langle O, S\rangle : S_j \in segues(O_{j-1}, O_j), \forall j \in \{1, \ldots, |I| - 1\}, \forall O \in \mathcal{P}(I)\} \tag{4.1}$$

[2]https://github.com/GiovanniGabbolini/play-it-again-sam.

In other words, $\mathcal{T}(I)$ is all the candidate solutions, and a given candidate solution $\langle O, S\rangle \in \mathcal{T}(I)$ comprises an ordering O of the items in I, and a corresponding sequence of segues S.

Problem 1 (Tour Finding Problem)**.** *Given a set of songs I, find a solution $\langle O, S\rangle \in \mathcal{T}(I)$ that maximises some $utility(\langle O, S\rangle)$.*

In this Chapter, we define $utility(\langle O, S\rangle)$ in a simple way, based only on the interestingness scores for the segues, S. Let $score(s)$ be a real number ranging from zero to one. Then we define the utility of a solution $\langle O, S\rangle$ as the mean score of the segues:

$$utility(\langle O, S\rangle) = \frac{\sum_{s \in S} score(s)}{|I| - 1}$$

$utility(\langle O, S\rangle)$ is a real number ranging from zero to one.

More sophisticated definitions of $utility$ are, of course possible. For example, we might reward solutions in which similar songs are near each other in O, or where similar segues are more distant from each other in S. These are matters for future exploration with users. In this Chapter, we focus on obtaining insights into different algorithms for finding segues with high utility, rather than on the best definition of utility.

Notice that, even though our focus in this work is on repeated consumption of music, Problem 1 can be solved to find a tour of any set of songs, including ones that are new to the user. Also, notice that Problem 1 is NP-hard; intuitively it is related to the Travelling Salesman Problem. We make the analogy clearer in Section 4.2.2.3.

4.2.2 Algorithms

We introduce three algorithms for Problem 1: two are heuristic methods (GREEDY and HILL-CLIMBING) and one is an exact algorithm (OPTIMAL).

4.2.2.1 GREEDY

The GREEDY algorithm builds a solution iteratively, by choosing the next song to be the one with the segue of highest score. We give pseudo-code in Algorithm 2.

Algorithm 2: GREEDY

```
1  $O \leftarrow$ empty list
2  $i \leftarrow$ random element from $I$
3  append $i$ to $O$; remove $i$ from $I$
4  while $|I| > 0$ do
5      $i^* \leftarrow arg\,max_{i' \in I}(\,max_{s \in segues(i,i')} score(s))$
6      $s^* \leftarrow arg\,max_{s \in segues(i,i^*)} score(s)$
7      append $i^*$ to $O$; remove $i^*$ from $I$
8      append $s^*$ to $S$
9      $i \leftarrow i^*$
10 end
11 return $\langle O, S \rangle$
```

4.2.2.2 HILL-CLIMBING

Hill-climbing is a local search algorithm that starts from a random candidate solution, then iteratively replaces that candidate by a neighbouring candidate whose utility is highest; it stops if no neighbour would result in an improvement in utility. Many flavours of hill-climbing exist, e.g. see [RN10]. We adopt the version with two parameters: restarts and patience. So, we run the algorithm multiple times (the restarts) and, for a certain number of iterations (the patience), we tolerate replacement by neighbours even if none of them improves the utility.

In order to use hill-climbing for Problem 1, we have to decide how to define the neighbourhood of a candidate solution. In this Chapter, we define the neighbourhood of a solution (a tour) as all the solutions that can be obtained by swapping two consecutive songs at random. More formally, given a candidate solution $\langle O, S \rangle$ and a random $r \in \{2, \ldots, |I|\}$, the neighbourhood of the candidate induced by r is:

$$N(\langle O, S \rangle, r) = \{\langle O', S' \rangle \in \mathcal{T}(I) : O' = [O_1, \ldots, O_r, O_{r-1}, \ldots, O_{|I|}]\}$$

The members of this set of neighbours share the same new ordering of the items but they differ in their segues.

We give pseudo-code for HILL-CLIMBING in Algorithm 3.

Algorithm 3: HILL-CLIMBING

```
$candidate\_solutions \leftarrow$ empty list
while $restarts > 0$ do
    $patience\_left \leftarrow patience$
    $current \leftarrow$ random element from $\mathcal{T}(I)$
    while True do
        $r \leftarrow$ random element from $\{2, ..., |I|\}$
        $neighbor \leftarrow arg\,max_{t \in N(current,r)} utility(t)$
        if $utility(neighbor) \geq utility(current)$ then
            $patience\_left \leftarrow patience$
            $current \leftarrow neighbor$
        end
        else
            if $patience\_left > 0$ then
                $patience\_left \leftarrow patience\_left - 1$
                $current \leftarrow neighbor$
            end
            else
                break
            end
        end
    end
    append $current$ to $candidate\_solutions$
end
return $arg\,max_{t \in candidate\_solutions} utility(t)$
```

4.2.2.3 OPTIMAL

The OPTIMAL algorithm finds an optimal solution t^* to Problem 1, i.e. a solution to Problem 1 with maximum $utility$.

Given a set of songs I, the algorithm builds a complete and weighted graph $\mathcal{G}(I)$. The nodes of $\mathcal{G}(I)$ are the songs I. $\mathcal{G}(I)$ is complete, so between every two songs there is an edge. $\mathcal{G}(I)$ is weighted, so every edge has a weight, equal to one minus the score of the highest-scoring segue between the two songs. More formally, the weight of the edge between two nodes (songs) $i, i' \in I$ is:

$$w(i, i') = 1 - max_{s \in segues(i,i')} score(s). \tag{4.2}$$

A Hamiltonian path $H = [i_1, ... i_{|I|}]$ in $\mathcal{G}(I)$ is a path in $\mathcal{G}(I)$ that visits every song exactly once. The set of all H in $\mathcal{G}(I)$ is equal to the permutations of I, and so it follows that $H \in \mathcal{P}(I)$. We define the weight W of H as:

$$W(H) = \sum_{j=1}^{|I|-1} w(H_j, H_{j+1}) \tag{4.3}$$

Intuitively, an H corresponds to a solution $t \in \mathcal{T}(I)$. Also intuitively, a Hamiltonian path with minimum weight H^* corresponds to the optimal solution t^*. More formally, it is possible to define a function f to map an H to a t, as follows:

$$f(H) = \langle H, S \rangle \;\; where \;\; S_j = arg\,max_{s \in segues(H_j, H_{j+1})} score(s),\; \forall j \in \{1, \dots, |I|-1\}.$$

$f(H)$ satisfies the membership conditions expressed in Equation 4.1, so $f(H) \in \mathcal{T}(I)$.

Lemma 2. *There is an* H *such that* $utility(f(H)) = utility(t^*)$.

Proof. Let $t^* = \langle O, S \rangle$. O is a permutation of I and so it is a valid Hamiltonian path H. By construction, $utility(f(H)) = utility(t^*)$. □

Lemma 3. H^* *is such that* $utility(f(H^*)) = utility(t^*)$.

Proof. By contradiction, we assume that $utility(f(H^*)) \neq utility(t^*)$. t^* is the optimal solution, so $utility(f(H^*)) < utility(t^*)$. Because of Lemma 2, there is an $\overline{H}$ such that $utility(f(\overline{H})) = utility(t^*)$. $\overline{H} \neq H^*$, as $utility(f(\overline{H})) = utility(t^*) \neq utility(f(H^*))$. Also, $W(H^*) \leq W(\overline{H})$, because H^* is the Hamiltonian path with minimum weight. From Equations 4.3 and 4.2, we get the following:

$$\sum_{j=1}^{|I|-1} 1 - max_{s^* \in segues(H^*_j, H^*_{j+1})} score(s^*) \leq \sum_{j=1}^{|I|-1} 1 - max_{\overline{s} \in segues(\overline{H}_j, \overline{H}_{j+1})} score(\overline{s})$$

$$\sum_{j=1}^{|I|-1} max_{s^* \in segues(H^*_j, H^*_{j+1})} score(s^*) \geq \sum_{j=1}^{|I|-1} max_{\overline{s} \in segues(\overline{H}_j, \overline{H}_{j+1})} score(\overline{s})$$

and so $utility(f(H^*)) \geq utility(f(\overline{H})) = utility(t^*)$, which contradicts the hypothesis. □

In conclusion, it is enough to compute $f(H^*)$ to obtain an optimal solution t^*. One way of finding H^* is by solving a Travelling Salesman Problem, or TSP. As suggested by [LK75], we can add a dummy node n to $\mathcal{G}(I)$ with zero-weighted edges to all the songs, then solve the TSP with start and end in n, and finally exclude n from the solution to obtain H^*. In the experiments that we run in this Chapter, we use the Concorde TSP solver[3].

4.3 Experiments

4.3.1 Implementation

The algorithms of Section 4.2 assume the existence of two functions: $segues$ and $score$. We resort to the implementation of those two function proposed in Chapter 3. In Chapter 3, the $score$ function is based on the interestingness of the segues. The $segues$ function can find segues of two kinds: informative (based on a knowledge graph) and funny (based on simple word-play). In the work reported in this Chapter, we restrict to informative segues. In Chapter 3 we found that the interestingness of informative segues is correlated with human perceptions of segue quality.

4.3.2 Dataset

We build a dataset from the Spotify Million Playlists Dataset (MPD) [CLSZ18]. The MPD dataset contains playlists, which is what our algorithms take as input. Of course, we ignore the ordering of the playlist, treating it as a set rather than a list, because our algorithms impose a fresh ordering, as well as decorating with segues.

We limit ourselves to playlists of maximum 50 songs. We believe longer inputs to be unlikely in practice, as it would lead to a listening time of more than three hours, assuming every song to last four minutes. We sample the MPD with stratified random sampling by playlist length: we sample at random 20 playlists of length 50, 20 of length 49, and so on, down to 20 playlists of length five, as five is the minimum length of MPD playlists. In fact, we apply this procedure twice, with two different random seeds, to obtain a *main* dataset and a *side* dataset.

4.3.3 Parameter tuning

HILL-CLIMBING has two parameters, as described in Section 4.2.2.2: the patience and the restarts. We set the parameters by choosing the configuration

[3]`https://github.com/jingw2/pyconcorde`.

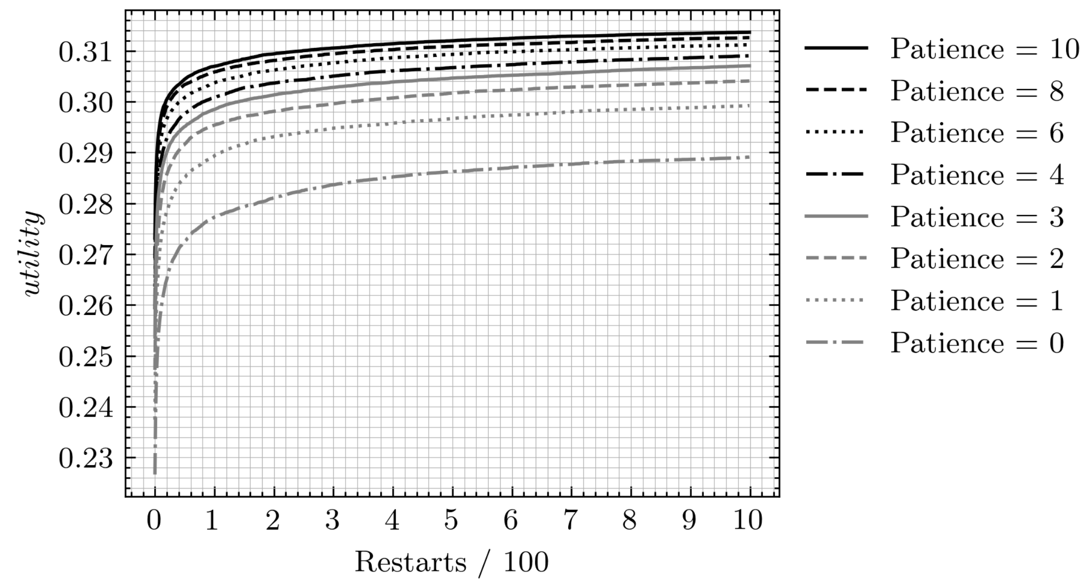


Figure 4.2: Average *utility* of HILL-CLIMBING, as a function of the restarts and of the patience.

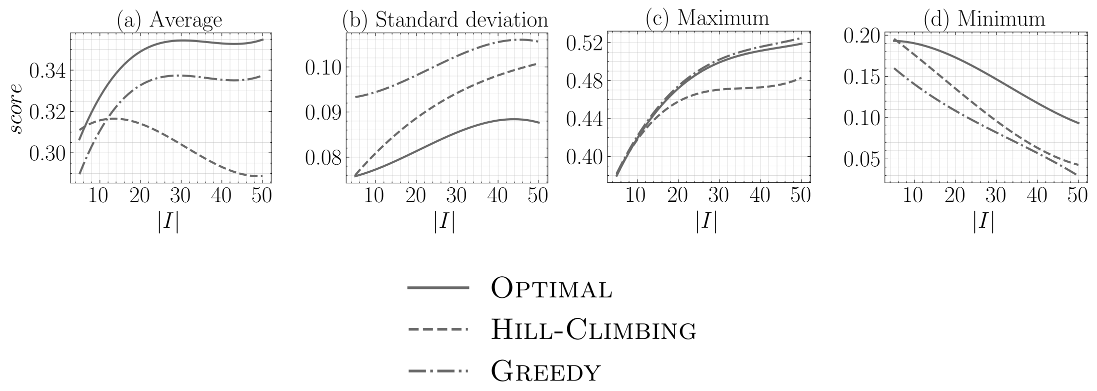


Figure 4.3: Performance of the algorithms, as a function of the input size.

that maximises the average *utility* of solutions. We run HILL-CLIMBING on the *side* dataset of Section 4.3.2, with various parameter configurations, and we measure the average *utility* of the solutions. We report the results in Figure 4.2. We notice that the average *utility* grows with patience and restarts, until it saturates. We choose the parameter values to be those before saturation, and so we set the patience to 10 and the restarts to 40.

4.3.4 Performance

We benchmark the algorithms by monitoring the *score* of the segues in the tours they produce using the *main* dataset. We construct four curves showing the mean of: average *score*; standard deviation; maximum; and minimum *score* of segues in the solutions, as a function of the input size. The average $score$ of the segues in a solution is equivalent to the $utility$ of the solution. For presentation purposes, we do a least square fitting and plot the fitted curves in Figure 4.3[4].

GREEDY, by construction, produces segues of high $score$ at the beginning of a solution, and of low $score$ at the end of a solution. That is, the segues in the solutions found by GREEDY exhibit falling average $score$, with highest values of maximum $score$, and lowest values of minimum $score$, which result in the highest standard deviation. The other two algorithms, HILL-CLIMBING and OPTIMAL, do not share the characteristic of falling average $score$, and produce solutions that are different from those produced by GREEDY. The solutions found by OPTIMAL are characterised by highest average $score$, and high values of maximum and minimum $score$, which contribute to the lowest values of standard deviation. Figure 4.3 provide some indication that HILL-CLIMBING and OPTIMAL produce, in general, the same solutions for small inputs. In the case of HILL-CLIMBING, as the input size grows, the average $score$ falls, the maximum stops increasing, and the minimum falls, causing an increase in standard deviation. As we see, HILL-CLIMBING struggles for large inputs, but it is likely to find the optimal solution for small inputs. On the other hand, GREEDY behaves similarly for small and large inputs, but it is unlikely to find the optimal solution for both small and large inputs.

4.3.5 Runtime

We compare the times each algorithm requires from taking an input to producing an output. More specifically, we carry out what we will call a granular analysis. By this, we mean that we consider the time required by calls to the functions $segues$ and $score$. All the algorithms call these functions, but they call them a different number of times. There are other parts of each algorithm which are not shared, and that are written in different programming languages, e.g. the optimal TSP solver is in C, while the hill-climber is in Python. We found that the granular runtimes, computed as above, differ from the full runtimes by only a small extent, and so we believe that the granular runtimes give us a fair comparison.

We run the algorithms on the *main* dataset of Section 4.3.2. We construct a curve featuring the granular runtimes as a function of the input size. For presentation purposes, we do a least square fitting, as in Section 4.3.4. We

[4]We use a polynomial of degree three, as we find empirically that it produces a good fit.

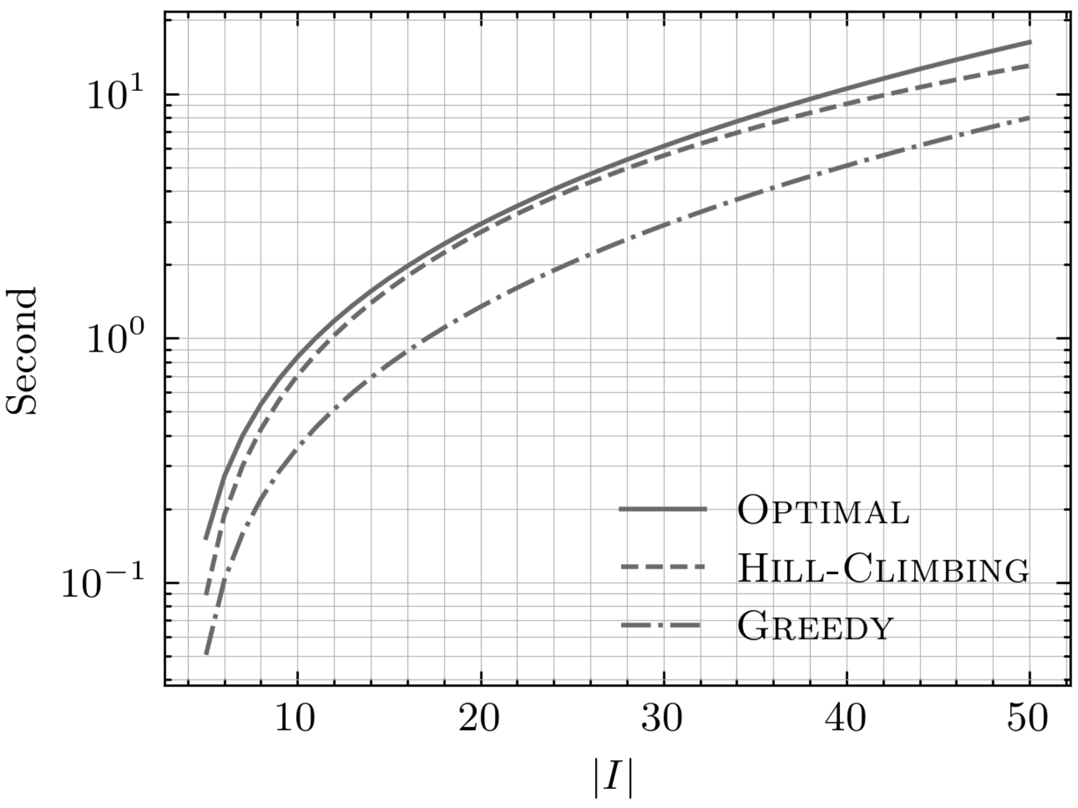


Figure 4.4: Granular runtimes of the algorithms, as a function of the input size.

report the fitted curve in Figure 4.4. We find that Greedy is the fastest, while Optimal is the slowest. Hill-Climbing is slightly faster than Optimal, and much slower than Greedy. The runtime of Hill-Climbing might decrease if we decrease the values of its parameters, at the cost of lower performances, as we discussed in Section 4.3.4.

4.4 Conclusion

In this chapter, we introduce three algorithms that can find tours: two heuristic methods (Greedy and Hill-Climbing) and one exact algorithm (Optimal). The results of our experiments highlight that Greedy produces tours that are substantially different from those found by Hill-Climbing and Optimal. Tours found by Greedy have segues with falling $score$s but highest maximum $score$s. Hill-Climbing and Optimal find similar tours to each other for small input sizes, and do not have falling $score$s. In this Chapter, we evaluate the algorithms with offline experiments only, which ignore the user perspective. In the next Chapter, we present a user-centered investigation of music tours.

Chapter 5

User-centered perspectives on music tours

5.1 Introduction

Music tours are our mean of achieving song-level intelligibility. In Chapter 4, we introduced three algorithms for generating music tours, and we evaluated those algorithms with offline experiments only, which ignore the user perspective. Therefore, in this Chapter, we present a user-centered investigation of music tours. We consider two of the algorithms proposed in Chapter 4, GREEDY and OPTIMAL, and we set up semi-structured interviews, where interviewees judge tours recommended by the two algorithms. We omit Chapter 4's third algorithm, HILL-CLIMBING as we found it to produce tours mostly equivalent to those produced by OPTIMAL, especially for small inputs, see Section 4.3.4. In the offline experiments of Chapter 4, we find that GREEDY and OPTIMAL differ in their total segue interestingness, and in the distribution of interestingness throughout the tour.

With the interviews, we contribute to the first user-centered evaluation of the two algorithms. We are interested in checking the results of the off-line experiments, in assessing the overall quality of the tours recommended by the two algorithms, and the interestingness of their segues, all from the user perspective. We are also interested in evaluating other aspects of tours, which we did not investigate in the offline experiments, such as segue diversity, segue narrativity, defined as the quality that sequences of segues in a tour present a narrative with a coherent text, and song arrangement. Finally, we are interested in investigating the fundamental issue of whether users value the concept of tours in general.

Summing up, we aim to fulfil five goals. **Goal (1)** is finding attributes of the tours. Attributes of interest could be: perceived interestingness distribution within tours; segue diversity; segue narrativity; and song arrangement. **Goal (2)** is identifying what attributes of tours are desirable and not desirable. **Goal (3)** is assessing the quality of tours, in terms of segue interestingness and overall tour quality. **Goal (4)** is assessing whether users value the concept of tours in general. **Goal (5)** is identifying possible improvements to tours.

We formulate the following research questions, as a guide to navigate the wealth of results we gather from the interviews:

RQ1: What are the attributes of the tours?

RQ2: What attributes of tours are good/bad?

RQ3: Which algorithm recommends better tours, and why?

RQ4: How valuable is the concept of tours in general?

RQ5: What are possible improvements to tours?

As mentioned previously, to the best of our knowledge, there is just one other work on music tours generation: [BMT+19]. Behrooz *et al.* develop a simple prototype for generating tours, which work by finding song-to-song segues for consecutive songs in an input playlist, and evaluate their prototype using semi-structured interviews. Our work in this Chapter is similar to [BMT+19], as we also employ semi-structured interviews as an experimental protocol, but different, as we evaluate two different tour-generation algorithms: Greedy and Optimal. We draw important distinctions between the prototype proposed in [BMT+19] and Greedy & Optimal in Section 4.1. Moreover, our interviews investigate several topics not investigated in the interviews reported in [BMT+19], such as the topic of segue diversity and narrativity. We refer the reader to Chapter 2 for more details on the research on tours, and for a broader understanding of how research on tours relates to other MIR research on playlists.

5.2 Method

We employ two of the tour generation algorithms we introduced in Chapter 4, Greedy and Optimal, without any alteration. The algorithms assume that the set of songs to be included in the tour are given *a priori*, and the algorithms must recommend segues and an arrangement of songs, so as to generate a tour of the kind shown in Chapter 4, Figures 4.1 (a) and (b).

Given a 'small' set of songs, e.g. several orders of magnitudes smaller than the whole catalogue of songs, the goal of the algorithms is to find a permutation of the songs, and a list of segues between the songs, maximising the average $interestingness$ of the segues. The Greedy algorithm works by building a tour iteratively, by choosing the next song to be the one with segue of highest score. The Optimal algorithm, instead, works by reducing the problem of finding a tour to the TSP problem, solved with an optimal solver. We refer the reader to Chapter 4 for a complete description of Greedy and of Optimal.

5.3 Experiment

We set-up a semi-structured interview experimental protocol, as it suits our need to answer both open and closed questions in the same sitting [Ada15]. For example, one closed question is which of the tours produced by the two algorithms, Greedy or Optimal, is preferred (part of **RQ3**), while one open question is what can be improved in tours (**RQ5**).

The interview protocol follows a within-subject design, i.e. each participant is subject to both treatments (Greedy and Optimal), in order to elicit an explicit comparison of the algorithms. Greedy recommends tours which are different from those recommended by Optimal. For example, the first segue in tours by Greedy tends to be more interesting than the last, while in tours by Optimal the first and last segues tend to be equally interesting, see Chapter 4.

The conversation revolves around example tours recommended by the algorithms. Some days ahead of the interview, we ask participants to send us a list of ten songs they are familiar with, making sure they are all from different artists. We set the number of songs to ten so as to reduce fatigue effects. We run the algorithms on the ten songs, and we prepare a *pdf* textual file with a visual representation of the resulting tours, side by side, similar to Figures 4.1 (a) and (b). We randomise the order, so that sometimes Optimal is on the left hand-side and Greedy is on the right hand-side, and vice versa. We also prepare an *mp3* audio file of the tour, comprising 15-second song previews, and segues that are read by a Text-to-Speech (TTS) engine, for a total duration of approximately seven minutes. Songs previews are from the Spotify API [1], and the TTS engine is the Google Text-to-Speech engine[2]. We use song previews rather than entire songs because previews are freely available. The first tour in the *mp3* corresponds to the one that the participant will see on the left in the *pdf*, and the second tour will be the one on the right in the *pdf*. Both the *mp3* and the *pdf* files are used during the interview to support the conversation. Algorithm names are excluded from both the *pdf* and *mp3* files. A sample *pdf* file and *mp3* file used during an interview are in the additional materials.

This interview study is approved by our organisation's ethics committee and it is conducted one-to-one by Giovanni Gabbolini (the interviewer) and a participant (the interviewee) via Zoom.

5.3.1 Interviewees

We recruit interviewees via a student mailing list within our university in Ireland. We make sure that the interviewee's first language is English, so that they can properly understand the segues. We ended up with 16 interviewees, who we will refer to as i_1 to i_{16}. One interviewee is a graduate, while the other 15 are currently students. Among them, four are undergraduates and 11 are postgraduate students. The interviewees have/seek degrees in a range of disciplines: arts, psychology, medicine, geography, computer science, history, music, social sciences, food sciences and human-computer interaction. Also, 15 out of 16 interviewees are users of music streaming services. We do not collect other demographic information, as we considered it not essential for the scope of the study.

5.3.2 Interview guide

At the beginning of the interview, the interviewee listens to the *mp3* version of their two tours, and views the *pdf*. Both the interviewer and the interviewee can hear the *mp3* and both can see the *pdf*. The *pdf* remains visible to both, until the end of the interview. Once the *mp3* is over, we ask questions aimed at answering the **RQ**s of Section 5.1.

[1]`https://developer.spotify.com/documentation/web-api/libraries/`
[2]`https://pypi.org/project/gTTS/`

To answer **RQ1-2**, we ask the interviewee to identify different attributes of the tours. We start with two open questions about what the interviewee likes and does not like about the tours, and we continue with other open questions probing for specific attributes of tours, such as distribution of segue interestingness throughout the tour ("is the first segue more interesting than the last?", and "does the interestingness of segues, from first to last, increase, decrease, stay equal, or something in the middle?"), segue diversity, segue narrativity ("are segues linked together to form a narrative, or are they independent pieces of text?"), segue top-bottom bias ("is the first segue more interesting than the last?") and song arrangement. Then, we ask the interviewer to assess each tour with respect to these attributes, and eventually express whether that attribute is important for the tours. To answer **RQ3**, we ask interviewees to assess the quality of the tours according to two dimensions: the interestingness of the segues and the overall tour quality. In particular, we ask interviewees to compare tours recommended by Greedy and tours recommended by Optimal, according to the two dimensions. To answer **RQ4**, it is necessary to take extra-care, as the interviewee might assume that tours are the work of the interviewer, and may tend to provide a positive concept evaluation accordingly. One strategy to relieve the pressure from the interviewee is showing non-judgemental acceptance, as suggested in [Ada15]. Therefore, to answer **RQ4**, the interviewer asks "People have mixed opinions on whether they would welcome a tour of their music or not. How do you see this issue?". To answer **RQ5**, the interviewer starts with two open questions about what the interviewee would like and would not like to see in tours, and continues with open questions probing whether tours should feature non-familiar music or familiar music. We provide the full interview guide text in the additional materials.

5.4 Results

Our interviews produce a wealth of material: 11 hours of videos, i.e. 42 minutes per interviewee, on average. Interviews are manually transcribed by Giovanni Gabbolini, and we end up with a corpus of 44-thousand transcribed words.

We conduct a thematic analysis of our data to identify important ideas and themes from our interviews. Thematic analysis is used for many kinds of qualitative analysis work in human-computer interaction (HCI), e.g. [WMH^{+}21]. We follow the process described by Braun and Clarke [BC06] in which researchers familiarize themselves with the data, then generate codes and group them to identify higher-level themes. Each phase of the analysis is conducted by Giovanni Gabbolini independently, and validated by Giovanni Gabbolini and Derek Bridge. We identify four main themes, i.e. “tour attributes”, “tour quality”, “concept evaluation” and “tour improvements”. The first theme addresses **RQ1-2**, the second theme addresses **RQ3**, the third theme addresses **RQ4**, and the fourth theme addresses **RQ5**. Every theme is covered in one Section below. Each theme has a number of sub-themes, covered in the subsections.

5.4.1 Tour attributes

We ask interviewees to identify different attributes of the tours, to assess the tours with respect to each attribute, and eventually express whether that attribute is important for the tours, as we explained in Section 5.3.2. Each subsection below accounts for a different attribute of tours, while Table 5.1 shows how the tours recommended by the algorithms compare according to the attributes.

5.4.1.1 Segue top-bottom bias

Many interviewees mention that the first segue in a tour is more interesting than the last segue, which is a phenomenon we refer to as top-bottom bias. Among the 16 interviewees, 14 find that Greedy exhibits top-bottom bias, as we might expect. In Optimal, the trend is less clear, as we might also expect: seven interviewees find top-bottom bias, seven do not, and two interviewees cannot decide. In total, seven interviewees say that the top-bottom bias is stronger in Greedy than in Optimal and the rest do not offer an opinion. The results corroborate the offline results in Chapter 4, where Greedy is found to have top-bottom bias, and Optimal is not.

In Chapter 4 we also investigate how the interestingness of segues varies throughout the tour, referring to this as the *outline*, finding that in Greedy the outline is falling and in Optimal it is not falling. In our study here, the outline of Greedy is judged to be falling by six interviewees, bumpy by six, pyramidical by two, and flat by two. The outline of Optimal is judged rising by five interviewees, falling by four, flat by three, bumpy by two, pyramidical by one, and in one case we do not have an answer. So, the results corroborate only partially those in Chapter 4, probably because the $interestingness$ function used in Chapter 4 to infer the outlines, and in the work at hand, only partially corresponds with what humans judge to be interesting, as we detail in Section 5.4.2.1.

Table 5.1: Tour comparison according to several attributes (rows one to four) and overall quality (rows five and six). Columns two to five indicate how many interviewees say that their Greedy tour features the attribute more than their Optimal tour (Greedy $>$ Optimal), less than Optimal (Greedy $<$ Optimal), approximately the same as Optimal (Greedy $\simeq$ Optimal), and the number of missing answers (N/A). The total number of interviewees is 16.

	Greedy $>$ Optimal	Greedy $<$ Optimal	Greedy $\simeq$ Optimal	N/A
Segue top-bottom bias	7	0	0	9
Segue diversity	9	1	4	2
Segue narrativity	0	6	4	6
Song arrangement	5	6	1	4
Segue interestingness	9	5	2	0
Tour quality	9	5	1	1

5.4.1.2 Segue diversity

The majority of the interviewees (14 out of 16) mention that the diversity of the segues in the tours, or *segue diversity* for short, is an important attribute to have. (Notice that this attribute concerns the diversity of the segues, not of the songs.) Many interviewees mention that tours with low segue diversity would not be nice to listen to. For example, i_8 says: "*then (the segues) start repeating, and so it gets less interesting as we go down*"; and i_{16} says: "*I think if I was getting the same information [...] that would be becoming boring after a while, I [...] want diversity in information*". Many other interviewees make a similar point, that tours with higher segue diversity are nice to listen to. For example, i_5 says: "*I just preferred the diversity, and it was like you didn't know what was coming next, and what bit of information you were going to learn was. That was kind of nice, that you didn't know!*"; and i_1 says: "*I feel like the information is [...] diverse [...] I'd say it keeps your attention more that way*". Most of the above comments refer to the diversity of segues that are relatively close to each other in a tour. That is, it is important for neighbouring segues to be diverse, while we have less guidance from our interviewees for segues that are more remote from each other. For example, i_5 says: "*I felt like (the segues) grabbed my attention a bit more (in* Greedy*), because (in* Optimal*), the last 6 six were about genres ... Maybe those six were also on (*Greedy*), but actually I did not notice as much*".

Respectively three and four interviewees say that Greedy and Optimal have low segue diversity, while respectively six and five of them say that Greedy and Optimal have high segue diversity. We do not have an answer in the rest of the cases. The diversity is sometimes low because neither algorithm implements an explicit segue diversity mechanism.

Nine interviewees say that their Greedy tour is better than their Optimal in diversity, one says the opposite, four say they are not sure, and the rest do not offer an opinion.

The result may be explained by our previous work. In Chapter 4 we found that the standard deviation of the $interestingness$ scores of the segues in Greedy's tours is higher than in Optimal's tours. That is, Greedy generates segues with a broader range of $interestingness$ values, some segues are very interesting and some are not, and segues with different interestingness values typically have different topics.

5.4.1.3 Segue narrativity

The narrativity of the segues in tours, or *narrativity* for short, refers to the quality that sequences of segues in tours present a narrative with a coherent text. Very few interviewees (three out of 16) are in favour of narrativity, as two points against narrativity are made. Point (1) is that narrativity can correlate with low diversity, and low diversity is something to avoid, as mentioned in Section 5.4.1.2. For example, i_{12} says: *"I don't think (narrativity) really matters per se, like, if you had all (segues) linked to each other, it could get quite repetitive"*. Point (2) is that the textual flow would not be easy to follow in any case with songs in between. For example, i_6 says: *"I don't think [...] that any kind of coherent narrative it's something I'd look for [...] I don't think it is all that important really."*; and i_8 says *"I think (the segues) should be independent! Because if there was a flow there I'd forget what it was saying before while listening to the song, so I'd lose the flow! So I guess independent makes much more sense"*.

Only two and six interviewees say that respectively GREEDY and OPTIMAL have high segue narrativity, while respectively nine and six of them say that GREEDY and OPTIMAL have low segue narrativity. We do not have an answer in the rest of the cases. The narrativity is oftentimes low because neither algorithm implements an explicit segue narrativity mechanism.

Six interviewees say that their OPTIMAL tour is better than their GREEDY tour in narrativity, four say they are not sure, and the rest do not offer an opinion.

5.4.1.4 Song arrangement

Many interviewees (13) give their opinion on what makes a good song arrangement. The majority of them (11) mention some notion of similarity, with a variety of different terms, reflecting the fact that music similarity remains a concept with many definitions [JDE07]. For example, they say that subsequent songs should have the same "tempo", "tone", "mood", "energy", "melody" and "artist voice".

Interviewees do not agree on whether song arrangement is important in tours or not, as eight of them say it is important, while six say it is not. Interviewees i_6 and i_7 from the former group respectively say: *"Oh definitely! No matter what the context, if you were putting a bunch of songs together, you are going to want some elements of musical flow between them"* and *"you cannot set the tone in one tempo or a mood and change it rapidly [...] there's nothing worse"*.

Interviewees from the latter group say that segues make song arrangement not matter. For example, i_2 says *"with the segues [...] the order (of songs) doesn't really matter, cause there's still a connection"*; and i_{12} says: *"you are not listening to this song, and the next song begins: you listen to this song, and there's a break where some text is read out, then you listen to the next one [...] so no, I don't think the (song) order matters"*. Some interviewees even appreciate when subsequent songs are diverse. (Here, we are referring to song diversity, and not segue diversity.). For example, i_4 says: *"it was interesting seeing the connections [...] (between) different songs [...] (like that two very diverse bands) both started in 2006, I suppose that was really interesting"*; and i_2 says: *"(the tours) are both pretty interesting [...] because [...] I tried to choose music that was diverse, so it's interesting to see (the segues)"*. It may be that segues between diverse songs are more unexpected, hence more interesting, as the interestingness guideline (3) of Section 5.4.2.1 indicates. However, the interviews do not provide enough material to confirm this speculation.

Respectively six and seven interviewees say that song arrangement is not good in GREEDY and OPTIMAL. In fact, of course, neither algorithm optimises for song arrangement, but for segue interestingness. However, in our own separately published work on interpretable similarity measures [GB21b], we find that segue interestingness is related to similarity, which could explain why some interviewees say they are happy with the song arrangements (eight interviewees for GREEDY and seven for OPTIMAL).

Interviewees do not agree on which algorithm is better in song arrangement, as five of them say that in GREEDY the song arrangement is better than OPTIMAL, six say the opposite, one cannot decide, and four interviewees do not offer any opinion. The result is not surprising as neither algorithm directly considers song arrangement.

5.4.2 Tour quality

We ask interviewees to compare the quality of the tours recommended by OPTIMAL and GREEDY according to segue interestingness and the overall tour quality. The results are in the subsections below, and in Table 5.1.

5.4.2.1 Segue interestingness

Interviewees mostly agree that Greedy has more interesting segues than Optimal: nine interviewees say so, five say the opposite, and two cannot decide. The result is perhaps surprising, as Optimal is supposed to maximise overall segue interestingness. But Optimal's model is only partially accurate for two reasons, as noted in Section 5.4.4.1: (1) the $interestingness$ function itself is only partially accurate; (2) the interestingness of a list of segues is not a simple sum; other factors intervene, for example the segue diversity. In particular, we suspect that Greedy outperforms Optimal because it is perceived to be more diverse, and diversity positively impacts segue interestingness, as noted in Section 5.4.1.2.

5.4.2.2 Overall tour quality

Interviewees mostly agree that Greedy produces overall better tours than Optimal: nine interviewees say so, five say the opposite, one cannot decide, and in one case we do not have an answer. Some interviewees mention the reason behind their choice: nine of them mention more interesting segues; also nine participants mention better song arrangement. Note that interviewees were free to give multiple reasons, and some of them mention both segue interestingness and the song arrangement. We infer that Greedy is preferred to Optimal because of the segues, which are more interesting in Greedy, as noted in Section 5.4.2.1, while song arrangement is equally good in both algorithms, as noted in Section 5.4.1.4.

5.4.3 Concept evaluation

Interviewees evaluate tours positively, as the general opinion is that the sequences of songs and segues are nice to listen to. For example, i_5 says: *"I'd be happy to listen to that (tour) [...] you learn something new about all tunes you like to listen to"*; i_7 says: *"I really like it! [...] it is nice to see how all songs are linked together, it is really cool!"*; and i_1 says: *"I like how the songs [...] were linked by pieces of information, something that they shared"*. Some interviewees would like to see tours available in streaming services. For example, i_8 says: *"I'd certainly use this prototype if it was implemented in a music streaming platform."*; i_6 says: "*(the segues) would be quite interesting for a listener, if they were available in a streaming service*"; and i_5 says: *"I would like to see (tours) available, like they aren't very available. It'd be nice to have the option (on streaming services)* *to switch them on and off. It is nice to listen to text, or to listen to something being spoken, in between listening to songs"*. Finally, the majority of interviewees (nine out of 16) mention that they would welcome a tour like those they were presented with.

The majority of interviewees find the segues in tours to be interesting, along a number of different dimensions. For example, i_{11} and i_9 find segues to be non-trivial, as i_{11} says: *"It was a pretty cool (segue) actually. Like, I wouldn't have thought about making a connection"*; and i_8 finds segues to be surprising "*being surprised, by interesting facts, like what this segues are producing here, that would make the listening experience more exciting"*. More generally, i_{1-5}, i_7, i_8, i_{10-12} and i_{15} like the segues. For example, i_{15} says: *"I liked the connections, from top to bottom, like something that ties up everything"*. Some interviewees recognise the value of having segues. For example, i_5 finds segues to be educational: *"I also liked the bits of information [...] It was kind of educational as well as listening to songs"*; and i_{10} finds that segues add to the listening experience: "*(segues) add to the music, like you'd enjoy more your listening experience"*. The above results provide some evidence that the $interestingness$ function, even though only partially accurate, is nevertheless related to user-centered perceptions of segue interestingess.

However, four interviewees report that segues are occasionally uninteresting. For example, i_4 says: "*Some of the connections are a bit weaker [...] But I think it's mostly very interesting yeah!"*; and i_{12} says: "*some of the links are quite tenuous, like two artists are both from the USA, that one felt a bit grasping"*. These opinions agree with our previous work. In Chapter 4 we found that the interestingness of segues in tours is not constant, but subject to considerable deviation. Hence, from Chapter 4 we expect some segues to be considerably less interesting than others, which is exactly what we report here.

Interviewees mention that tour quality is context-dependent, a similar point being made in [BMT+19]: tours are suited for active listening, i.e. when attending to the music, and not passive listening, e.g. when the music plays in the background to some other activity. For example, i_4 says: "*sometimes [...] you may just want to listen to your music in the background. But sometimes, if you want to sit down and think about the music your are listening to, (the segues) might be like an interesting, like a different, fun, new way to consume your music."*; and i_9 says: "*a lot of times you listen to music as a background, whereas I think (for a tour you need) designated hours in your day, like I'm going to sit down and listen* *to this"*.

5.4.4 Tour improvements

Interviewees identify a number of potential improvements for tours.

5.4.4.1 Interestingness modelling

Many interviewees say what interestingness means for them, and, from this, we extract four guidelines on how to distinguish interesting from uninteresting segues.

Guideline (1) is to be specific, as some interviewees say that specific segues are more interesting than general segues. For example, i_1 says: *"I wouldn't like to see information that seems kind of generic maybe [...] it just seems no care and attention has been put in creating that"*. i_7 gives an example of very general segue: *"The bit the says "one is based in the US and the other is based in the same country", because I think this is a way of linking nearly every one even, I like knowing a little bit more of information about someone"*. The very same kind of segue is criticised also by i_{12}. Interviewees have a different opinion about specific segues. Again i_{12} says, *"the segues that are specific, are the most interesting, but the other like [...] these two people are from the same large country, it is a bit ... uninteresting"*, and provides an example of a more interesting segue, which involve a more specific place: *"this group founded in (New) Jersey and also this other"*.

Guideline (2) is to limit text length, as some interviewees say that long segues are typically undesirable. For example, i_2 says: *"I would not want to see overly long segues"*, and i_{14} says *"Not too much information, not too many words, otherwise I'd be like: please!"*. We do not investigate how much text is too much for an interviewee. However, we report that the longest segue that was shown to i_{14} was of 22 words, while for i_2 it was of 19 words.

Guideline (3) is to be aware of prior knowledge, as some interviewees say that segues that contradict their prior knowledge are to be avoided. For example, i_{15} says *"apart from that (same genre segue) on Christmas music, which is not correct. Apart from that, I enjoyed it"*. Also, many interviewees say that known segues are uninteresting. For example, i_{14} says *"One of the segues was U2 and Fionn Reagan are Irish and [...] I know that because I'm Irish! [...] that is just tedious and boring."*; and i_{12} says *"the rest (of the segues) was kind of ... I knew* *them already [...] so I'd be kind of "ehm, okay""*. Finally, many interviewees say that novel/unexpected segues are interesting. For example, i_{12} says: *"I didn't know some of this stuff, such as (these two bands) being from the same state, it's interesting"*; and i_{16} says: *"(in good segues) you'd have kind of information that you didn't know and you were just learning. Something interesting, like one of those of moments like oh my god I didn't know that these two bands were connected"*.

Guideline (4) is to avoid controversies, as some interviewees point out that controversial segues might be typically undesirable, such as those involving artists' personal lives. For example, i_{11} says: *"(I don't want to hear) scandals related to the artists [...] It'd ruin the listening experience"*. Similarly, i_8 says *"(I wouldn't want to hear something) too particular like, when this artist was married, or when this artist was in prison"*.

Only two of the guidelines above match the $interestingness$ function used by the algorithms in this work. This function, that we define in Chapter 3, combines rarity, which arguably entails specificity, and shortness. Therefore, Greedy and Optimal strive to maximise an imperfect $interestingness$ function, that could be improved by considering all four of the guidelines above. Even so, $interestingness$ does a good job in distinguishing actually interesting segues from others, as reported in Section 5.4.2.1. Moreover, in this work we model the interestingness of a list of segues as the sum of the $interestingness$ function applied to the individual segues in the list. This will only approximate human perceptions of interestingness because also other factors matter. For example, diversity of neighbouring segues makes the tour more interesting overall. One improvement is to correct the estimate of the interestingness of a list of segues by taking into account also diversity, and especially the diversity of nearby segues in tours, as noted in Section 5.4.1.2.

5.4.4.2 More biographical segues, less genre segues.

Many interviewees highlight examples of interesting and uninteresting segues seen in the tours. One class of interesting segues are (factual) biographical segues about the artists. For example, i_5 likes segues about awards won by artists; i_9 likes segues about locations, such as those about where an artist was born; i_3 likes segues about live events, such as in which festival an artist performed; and i_9 likes segues about dates, such as when an artist was born. Interviewees would like to see even more biographical segues, including ones that are not available in tours at the moment. For example, i_{12} says they would like segues to draw from music news databases, that gather information such as emerging artists listings, recording studios, and albums release dates. One class of uninteresting segues are (musical) genre segues, such as the "jazz" segue in Figure 4.1 (b). For example, i_1, i_3, i_4, i_6, i_{12}, i_{15} say that genre segues are not likeable and uninteresting.

We believe that biographical segues and genre segues are perceived as respectively interesting and uninteresting at least in part because of interestingness guideline (3) of Section 5.4.2.2: to be aware of prior knowledge. Biographical segues, as facts, match guideline (3): they are not likely to contradict user prior knowledge, unless the user is wrongly informed, and specific facts are likely to be novel/unexpected for non-expert users. For example, the segue "Flying Lotus is the grand-nephew of Alice Coltrane" of Figure 4.1 (a) cannot contradict user prior knowledge, unless the user is wrongly informed about the relation between the two artists, and they are likely to be novel/unexpected for the user, unless the user is an expert in artists' biographies. Genre segues, instead, may often not match guideline (3): they can contradict user prior knowledge, as people are found to often disagree on music genres [SCBG08], and they can be known to the user, as genres are one of the most common ways in which people characterise music [PDM$^+$22]. Interviewees agree with our interpretation, as i_{16} is disappointed by a genre segue contradicting their prior knowledge: *"I'd like not to grow a question that would put me off [...] you might thinking about that instead of enjoying the music [...] that Leonard Cohen (segue) is a good example [...] I started thinking: is he rock music?"*; while i_9 says they know genre segues already: *"There were someones that were linked by same genre [...] and maybe that's something you could have come up with yourself [...] I suppose that is something you have an idea of already"*; i_8 also thinks that genre segues are very trivial: "*those (genre) segues have problems, they are [...] very trivial*".

Biographical and genre segues belong to the broader groups of respectively factual and opinionable segues, which we suspect to be more and less interesting, again because of interestingness guideline (3): factual segues are not likely to contradict user prior knowledge, unless the user is wrongly informed, while opinionable segues, by definition, are more likely to do so. However, future work is needed to verify our supposition. Some factual segues that could be included in tours are suggested by the interviewees. For example i_{14} would like segues drawn from artists' biographies, both related and not related to music, e.g. which and how many pets an artist has; i_7 would like acoustical information segues, such as the tempo or key of a song; and i_9 would like lyrics segues, such as which keywords features in the song lyrics.

Summing up, segues in tours could be improved by including more biographical information and by avoiding musical genres. Another potential improvement is to include more factual information, such as those in the last paragraph, and avoid opinionable statements. The topics of segues can be modified by altering the knowledge graph available to algorithms, described in Section 5.2, to include the information that should feature in segues, and to exclude the information that should not feature in the segues.

5.4.4.3 Song selection

The algorithms we are using are given a set of songs as input, and return a sequence made from this same set of songs, broken up by segues. That is, the algorithms are not designed to perform any song selection. In this work, we ask interviewees to send us ten songs they are familiar with, that we input to the algorithms, as described in Section 5.3. In this Section, we report what interviewees say about how to appropriately select the songs in input to the algorithms. In summary, we find that the right level of familiarity is key.

The tours we show during the interviews feature familiar music only. Many interviewees mention that they would like tours to feature some unfamiliar songs, along with familiar songs, a concept we refer to as UF-tours (Unfamiliar/Familiar tours). A UF-tour is similar to popular features of music steaming services, such as Spotify's *daily mix*[3], which is a playlist of familiar and unfamiliar music. A UF-tour, however, is different from a *daily mix* because the unfamiliar songs are introduced by a segue. The majority of interviewees (10 out of 16) say they would welcome a UF-tour. For example, i_{12} says "*Spotify would give your daily mix (that is) songs you know mixed with similar songs [...] I would have never listened to. (A UF-tour) can be quite interesting actually, because it creates a connection, it is not just a vague music floating around [...] having that little bit of information to introduce the artist and the actual songs would be quite interesting I think, I would be you more inclined to go to the actual artist page and listen to more of their music.*"; and i_3 says: "*I'd be very interested in this, because on Apple Music, if you set auto-play from your songs, they will bring on just a random song [...] while this will give you why they are similar, what connection do they have.*" It is not clear whether UF-tours are preferred over F-tours (tours

[3]https://newsroom.spotify.com/2018-05-18/how-your-daily-mix-just-gets-you/

of familiar music only), because not enough interviewees explicitly compared the two concepts. Interviewee i_5 suggests a button to switch between an F-tour and a UF-tour, which can be a sound design choice before future investigation on the subject.

Many interviewees comment on tours that feature only unfamiliar music, a concept we refer to as U-tours. The general opinion is mostly negative, as no interviewee says that they would welcome a U-tour. The main reason is that segues between two unfamiliar songs lose their meaning. For example, i_9 says "*(segues are valuable) with music that is important to you. If that music is not important to you in the first place, I think (the segue) it's just going to be semantic information [...] it wouldn't bring any value to me*". Familiar songs are needed to keep interest in the tour alive. For example, i_{12} says "*I think the mix of familiar and novel songs works, because the familiar songs are kind of an anchor to you, whereas with two unfamiliar songs [...] (a segue) wouldn't mean much to me*". The above result also suggests avoiding segues between two unfamiliar songs in UF-tours.

One difficulty in implementing UF-tours and U-tours is recommending which unfamiliar music to display in the tour. Interviewees suggest that recommended music should fit the user's tastes. For example, i_2 says "*having music you don't like [...] is kind of inevitable when finding new music but ideally a tour would have music you will like in advance [...] having a tour that doesn't read your preference very well [...] wouldn't be good*". Moreover, it may be important that all songs in a tour are similar to each other, as noted in Section 5.4.1.4.

5.4.4.4 Presentation

Tours are presented to interviewees through two mediums, visual and aural, as explained in Section 5.3. The visual presentation is similar to Figure 4.1 (a) and (b), while the aural presentation is a sequence of songs, alternated by segues, read by a TTS engine. It is not clear which medium is preferred, as we do not investigate the matter explicitly.

Interviewees do however identify a number of flaws in the presentation of the tours. For example, the TTS is perceived as too "robotic", not human-like, and it is not liked by some interviewees. For example, i_6 says: "*(would have been better) if the audio links were not spoken as obviously by a computer, you know if it was more human-like, more engaging*". Also, some interviewees do not like the short pauses between the music and the segues, and would like the song and the segue to overlap a little, similar to radio programs. For example, i_5 says: *"you could merge the songs a little bit (so that) the voice over came in towards the end of the first song, while the second song began while the voice over is running, eliminating that downtime between the two"*. Finally, some interviewees question the rule of having a segue between every pair of songs, and suggest that, if not interesting enough, a segue could be skipped. For example, i_{16} says: *"If the information isn't going to be something that'd grab, I wouldn't want a segment, you-know? If you can't find an interesting enough segment, you will just have to not have a segment"*.

5.5 Conclusion

In this Section, we summarise the interviews reported in Section 5.4 to answer all the **RQ**s we pose and we discuss the limitations of the work that we have presented here.

5.5.1 Discussion of RQs

RQ1. What are the attributes of tours? (1) According to the interviewees, tours recommended by GREEDY feature a segue top-bottom bias, that is the first segue tends to be more interesting than the last segue, while tours recommended by OPTIMAL do not. (2) Tours do not always feature segue diversity, but tours recommended by GREEDY are found to be more diverse than tours recommended by OPTIMAL. (3) Tours do not always feature segue narrativity, that is the quality that the sequence of segues in tours presents a narrative with a coherent text, but tours recommended by OPTIMAL are found to feature more narrativity than tours recommended by GREEDY. (4) Tours do not always feature good song arrangement, and tours recommended by GREEDY are found to be equally good in song arrangement to tours recommended by OPTIMAL.

RQ2. What attributes of the tours are good/bad? Segue diversity is definitely a good tour attribute, as 14 out of 16 interviewees say so. Song arrangement is another good attribute, as interviewees say that good song arrangement is a reason as important as good segues to prefer a tour over another. Segue narrativity, instead, is considered a good attribute of tours by only three interviewees. In conclusion, segue diversity and song arrangement appear to be the two most important attributes of the tours investigated in this work.

RQ3. Which algorithm recommends better tours, and why? Greedy seems to recommend higher quality tours than Optimal. Interviewees motivate their choice mentioning higher segue interestingness and better song arrangement. We infer that Greedy is preferred to Optimal because of segue interestingness, which is higher in Greedy, while the quality of song arrangement is similar in both algorithms. In turn, segues in Greedy could be more interesting because of the higher diversity, which is regarded as one important characteristics of segues.

RQ4. How valuable is the concept of tours in general? The majority of interviewees find that the algorithms recommend tours with interesting segues, and say they would welcome a tour such as those they were presented with. However, accepting tours is dependent on context: tours require active listening, and they are not suited for passive listening. In conclusion, participants mostly evaluate the concept of tours positively in general.

RQ5. What are possible improvements to tours? (1) Take into account user prior knowledge and controversial content when scoring the segues. In particular, known segues, segues contradicting user prior knowledge and controversial segues should have low scores. (2) Include more segues with biographical information and fewer segues about musical genres. (3) Choose music carefully, admitting, from time to time, unfamiliar music, always chosen to fit the user's tastes. (4) Mind the presentation, employing an appropriate TTS engine to read the segues, avoiding silences by overlapping songs with segues, and skipping segues if not interesting enough. (5) Implement a good song arrangement mechanism based on song similarity, as song arrangement is a fundamental attribute of tours, but it is not considered by the algorithms we use. (6) Implement a segue diversity mechanism, as segue diversity is a fundamental attribute of tours, but it is not considered by the algorithms we use.

5.5.2 Limitations

Our work has limitations that should be acknowledged. We conduct semi-structured interviews, as they suit best our need of answering the **RQ**s, as argued in Section 5.3. However, this kind of experimental setting could affect the validity of some results. For example, when evaluating the concept of tours (**RQ4**), interviewees could infer that tours are the work of the interviewer. Participants will realise that tours are the focus of the work and will know that the tours they are being shown have been created by the interviewer. Hence, they may feel pressure to provide a positive evaluation. We attempt to relieve the pressure, as explained in Section 5.3.2. However, we cannot prove the effectiveness of the attempt.

Another limitation of our experimental protocol is that we present interviewees with tours of familiar music (F-tours), but we ask questions about the three variant of tours: tours with familiar music (F-tours), tours with both familiar and unfamiliar music (UF-tours), and tours with unfamiliar music only (U-tours). As such, the interview data may be positively biased towards F-tours, which were presented to interviewees, and may be negatively biased towards U-tours and UF-tours, which were not presented to interviewees. We find that UF-tours are preferred over F-tours by interviewees, which suggests that any bias is limited. However, we do not have further means for disentangling bias and real preference on this issue.

Moreover, semi-structured interviews are a high cost protocol, which constrains the number of participants [BC06], 16 in our case. In our case, we resort to a convenience sampling of university students, which does not imply that our result extends to a broader sample of the population. Moreover, the small number of participants limits the statistical power of the experiment, as we cannot afford to have more than two treatments [KW15]. In our case, we fix the two treatments to be the two tour generation algorithms, as we explain in Section 5.1. Therefore, we do not include any playlist recommendation algorithms among the treatments, so we cannot assess whether tours are preferable to playlists or not. By analysing the results of Section 5.4, we may infer that playlists are preferable to tours when listening to music passively, and that tours can be preferable to playlists when listening to music actively, which corroborates previous work [BMT$^+$19]. However, at this stage this is more an intuition than a definite answer. Similarly, we do not include other interesting treatments in the experiment, such as a tour generation algorithm which accounts for segue diversity, such as the one proposed in [BMT$^+$19]. In any case, the work of [BMT$^+$19] is not fully reproducible, as some fundamental parameters of their algorithm, such as the author-defined segue importance weights, are not shared in the paper.

Finally, while the interviews hint at several guidelines for the construction of functional tour generation algorithms, we cannot forecast the impact of those guidelines, and their effectiveness can only be assessed by comparing a tour generation algorithm that implements those guidelines and a tour generation algorithm that does not.

Chapter 6

Playlist tagging

6.1 Introduction

In the previous Chapters of this dissertation we focus on song-level playlist intelligibility. In the next two Chapters, we instead focus on playlist-level intelligibility, which we define in Chapter 1 as the degree to which the characteristics of a playlist can be understood by a human audience. One way to achieve playlist-level intelligibility is by playlist tagging, which is the task of assigning to a playlist one or more tags, drawn from a fixed vocabulary of tags[1].

In this Chapter, we focus on playlist tagging. There exist different categories of playlist tags, such as genre tags, e.g. "rock"; decade tags, e.g. "90s"; and listening context tags, e.g. "running". In this Chapter, we focus on tagging with listening context tags, for which we had a dataset available to use. The dataset we use was the only dataset of tagged playlists at the time we conducted our research. Subsequently, at the time of writing this dissertation, a new dataset has become available, with a broader set of tags [FKL$^+$21]. It would be interesting to apply the algorithms we propose below to this broader set of tags, which is a direction we leave as future work.

To the best of our knowledge, there exists only one attempt to predict the listening context of music playlists: [CKE20]. The authors of [CKE20] set up a multi-label classification problem, in which playlists are classified for their listening contexts, and they propose four classifiers: two matrix factorisation (MF)-based classifiers, that work by counting how many times a song is associ-

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

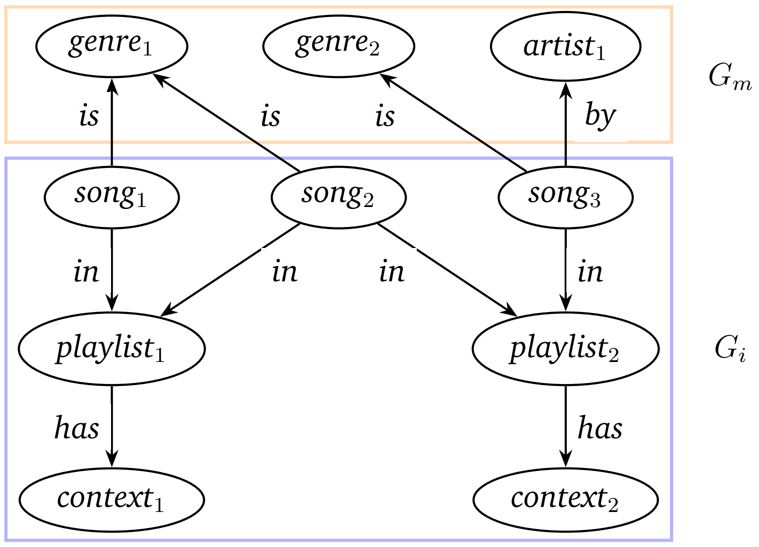


Figure 6.1: A knowledge graph representing two playlists and three songs in total. The bottom and top boxes indicate two portions: G_i, which contains song, playlist and listening context nodes, and G_m, which contains metadata nodes, such as musical genres.

ated with each playlist listening context, and two convolutional neural network (CNN)-based classifiers, that work with song audio. However, these classifiers are limited in that they do not incorporate song metadata, such as musical genres.

In this Chapter, we formulate two novel knowledge graph (KG)-based classifiers. KGs are a powerful data model, suitable for storing heterogeneous information [WBDB17]. Figure 6.1 depicts a KG like those we use, made up of two distinct portions: G_i and G_m. The portion G_i represents the membership of songs to playlists, and of playlists to listening contexts. The portion G_m represent song metadata, solving the limitation of existing classifiers that they do not use song metadata. The KG-based classifiers that we propose work by building a KG, such as the one depicted in Figure 6.1, embedding the KG, so that each node and edge is transformed to a feature vector, and using the song embeddings to predict the listening contexts of playlists.

We benchmark the classifiers with a dataset of playlists annotated with their listening contexts, originally proposed in [CKE20]. The two KG-based classifiers we propose achieve approximately 10% higher performance than the existing predictors. A sensitivity analysis reveals that the KG-based classifiers can incorporate song metadata effectively.

However, the two KG-based classifiers do not consider song audio. So, we formulate another two novel predictors, as the hybrid of the CNN-based and KG-based classifiers. As expected, the hybrid classifiers outperform MF-based, KG-based and CNN-based predictors, setting the new state-of-the-art performance in the task.

Related to playlist tagging is song tagging, a popular research topic in MIR. Song tagging is the task of assigning to a song one or more tags, drawn from a fixed vocabulary of tags. CNNs are the algorithms of choice in state-of-the-art song tagging research. [WFBS20], for example, offers a comparison of recent Convolutional Neural Network (CNN)-based classifiers: a CNN extracts learned features from the audio of a song, and leverages these features to output appropriate tags. Similarly, the state-of-the-art classifiers proposed in [PNP+18, WCNS20, CFSC17] are CNN-based. Progress in song tagging is enabled by the availability of large scale datasets, such as the Million Songs Dataset [BMEL+11], the MagnaTagATune Dataset [LWM+09] and the MG-Jamendo Dataset [BWT+19]. These datasets contain songs annotated with tags of several categories: genre tags (e.g. “jazz”), instrumentation tags (e.g. “guitar”), decade tags (e.g. “80s”), mood tags (e.g. “happy”) and listening context tags (e.g. “party”).

We release the source code and the dataset that supports our work here, so as to allow reproducibility and foster new research on the subject[2].

In summary, our contributions are:

1. the first two KG-based listening context predictors of music playlists that incorporate song metadata;
2. another two novel predictors that incorporate KGs and song audio;
3. a comparison of the predictors reporting approximately 10% higher performance than the state-of-the-art, and showing the impact of song metadata on performance.

6.2 Method

Predicting the listening contexts of playlists is framed by the authors of [CKE20] as a multi-label classification problem. The same authors propose four such classifiers (MF-AVG, MF-SEQ, CNN-AVG and CNN-SEQ). Here, we propose an-

[2]`https://github.com/GiovanniGabbolini/playlist-context-prediction`

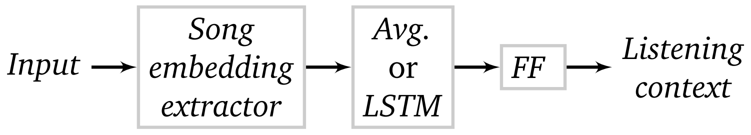


Figure 6.2: Schematic architecture of MF-AVG, MF-SEQ, CNN-AVG, CNN-SEQ, KG-AVG and KG-SEQ.

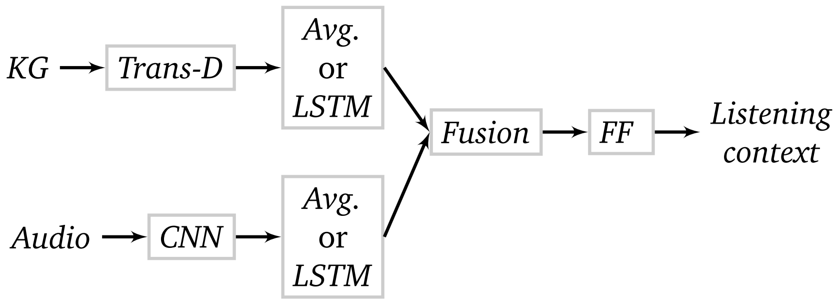


Figure 6.3: Schematic architecture of HYBRID-AVG and HYBRID-SEQ.

other four such classifiers (KG-AVG, KG-SEQ, HYBRID-AVG, HYBRID-SEQ). As we will explain below, six of the classifiers that we consider follow the schema depicted in Figure 6.2. The two hybrid classifiers follow the schema depicted in Figure 6.3. In the rest of this section, we summarise the four classifiers that were proposed in [CKE20], and we describe the four classifiers that we propose.

6.2.1 Matrix factorisation-based

The two matrix factorisation (MF)-based classifiers (MF-AVG and MF-SEQ), originally proposed in [CKE20], take as input a matrix $\mathbf{X} \in \mathbb{R}^{N,M}$ where N is the number of songs and M is the number of listening contexts. The element at row n and column m of $\mathbf{X}$ is equal to the number of times the n^{th} song appears in playlists that have the m^{th} context. The matrix $\mathbf{X}$ is factorised into two matrices, $\mathbf{S} \in \mathbb{R}^{N,H}$ and $\mathbf{C} \in \mathbb{R}^{H,M}$, using WR-MF, which is the MF procedure described in [HKV08], so that $\mathbf{SC} \approx \mathbf{X}$. H is the embedding dimension, which is a hyper-parameter of WR-MF. The rows of $\mathbf{S}$ and the columns of $\mathbf{C}$ contain, respectively, song and listening context embeddings. Then, the song embedding vectors for the songs in a given playlist (a subset of the embeddings contained in $\mathbf{S}$) are either averaged song-wise (in MF-AVG) or input to a single-layered LSTM network (in MF-SEQ), to get a playlist embedding vector, which is fed into a single-layered feed-forward (FF) network that outputs a score for each listening context.

The architectures of MF-AVG and MF-SEQ fit into the schema of Figure 6.2 as the matrix $\mathbf{X}$ is the *input*, and WR-MF is the *song embedding extractor*. Notice that MF-AVG and MF-SEQ work in two steps, that is the *song embedding extractor* is trained separately from the rest.

6.2.2 Convolutional neural network-based

The two convolutional neural network (CNN)-based classifiers (CNN-AVG and CNN-SEQ), originally proposed in [CKE20], extend the state-of-the-art in song tagging to playlist tagging. Given a song, they consider the full audio, and compute mel-spectrograms for every contiguous 3-seconds of audio. The mel-spectrogram is a hand-crafted feature extracted from audio, commonly used in many music information retrieval tasks, such as song tagging, e.g. [PNP+18, WCNS20, CFSC17]. The mel-spectrograms are input to a Convolutional Neural Network (CNN) with five *1D*-convolutional layers, which outputs an embedding vector for every 3-seconds of audio. Such embeddings are averaged point-wise, to get one song embedding vector. Given a playlist, the song embedding vectors are computed as above, and either averaged song-wise (in CNN-AVG) or input to a single-layered LSTM network (in CNN-SEQ), to get a playlist embedding vector, which is fed into a single-layered (FF) network that outputs a score for each listening context.

The architecture of CNN-AVG and CNN-SEQ fit into the schema of Figure 6.2 as the mel-spectrograms are the *input*, and the CNN is the *song embedding extractor*. Notice, however, that CNN-AVG and CNN-SEQ are end-to-end, that is the *song embedding extractor* is trained jointly with the rest.

6.2.3 Knowledge graph-based

A knowledge graph (KG) is a set of triples $G = \{(e, r, e') \,|\, e, e' \in E, r \in R\}$, where E and R denote, respectively, the sets of entities (nodes) and relationships (edges). KGs are suitable for representing heterogeneous information [WBDB17]. For example, [OON+16] builds a KG representing users, their interactions with songs, and acoustical metadata, such as what musical instruments are played in the songs.

The information we want to represent is: songs; playlists; listening contexts; and song metadata. So, we build a KG composed of two portions. (1) G_i: the portion containing song nodes, playlist nodes and listening context nodes. These nodes are connected by edges according to membership: a song node is connected to the playlist nodes the song belongs to, and a playlist node is connected to its listening context node. (2) G_m: the portion containing song metadata, i.e. the record label associated with the song, the musical genres associated with the song, the year and the month when the song was released, the artist of the song, the city and the country where the artist is currently based, and where they were born. We selected these items of metadata empirically, through informal experimentation, and by taking inspiration from previous work; for example, [KBJ20] finds that the release year of a song can be a predictor for the listening context. In future work, G_m can be readily expanded to include more song metadata, such as information extracted from song lyrics. For each piece of song metadata, there is a node in G_m. Song nodes are connected by edges to their metadata nodes. Song metadata may be missing, e.g. we may not know the record label for a particular song. We obtain metadata from the crowd-sourced database MusicBrainz[3].

Figure 6.1 depicts a KG, like those that we build.

We embed the KG using the TRANS-D algorithm, which is a state-of-the-art KG embedding algorithm [DWXG20]. TRANS-D produces an embedding vector for every node and edge in the KG, in such a way that the topology of the KG is preserved. In particular, given a KG G, and given a triple $(e, r, e') \in G$, TRANS-D produces three embedding vectors v_e, v_r and $v_{e'}$ that satisfy a relationship similar to $v_e + v_r \approx v_{e'}$, for every triple in G. The embedding vectors of the song nodes in the KG for the songs in a playlist are either averaged song-wise (in KG-AVG) or input to a single-layered LSTM network (in KG-SEQ), to get a playlist embedding vector, which is fed into a (FF) network, that outputs a score for each listening context.

The architecture of KG-AVG and KG-SEQ fit into the schema of Figure 6.2 as the KG we build is the *input*, and TRANS-D is the *song embedding extractor*. Notice that KG-AVG and KG-SEQ work in two steps, that is the *song embedding extractor* is trained separately from the rest.

The MF-based and KG-based algorithms both leverage information about listening contexts when computing song embeddings. However, KG-based algorithms exploit that information more effectively. For example, let us consider the scenario depicted by the portion G_i of the KG in Figure 6.1 where there are two playlists, $playlist_1$ and $playlist_2$, whose listening contexts are respectively $context_1$ and $context_2$, and which contain respectively the songs $song_1$ & $song_2$ and $song_2$ & $song_3$. MF song embeddings are aligned with their listening

[3]`https://musicbrainz.org`

contexts, as explained in Section 6.2.1. In the example above, the MF embedding of $song_1$ is aligned with $context_1$, the MF embedding of $song_3$ is aligned with $context_2$, and the embedding of $song_2$ is aligned with both $context_1$ and $context_2$. However, $song_1$ and $song_2$ are in the same playlist ($playlist_1$). As such, we expect the embedding of $song_1$ to be aligned, to some extent, also with $context_2$, and not only with $context_1$; similarly for the embedding of $song_3$. That is, MF-based algorithms 'short-circuit' the representation of playlists by modelling the association of songs to playlist listening contexts directly. KG embeddings preserve the topology of the KG, and so can overcome the short-circuiting problem of the MF-algorithms. In the example above, the songs in G_i are all connected with each other, via the explicit representation of the playlists as well as the listening contexts. That is, the embeddings of $song_1$, $song_2$ and $song_3$ are all aligned, to some extent with $context_1$, and to some other extent with $context_2$. The short-circuiting problem undermines the performance of the MF-based classifiers, as we empirically prove in Section 6.3.3.2. In a similar vein, [LCC18] propagates tags among songs in the same playlists, and measure an increase in performance.

6.2.4 Hybrid

The CNN-based classifiers and the KG-based classifiers differ on their input data, as the CNN-based classifiers rely on song audio, while the KG-based classifiers rely on a KG representation of songs, playlists, listening contexts, and song metadata. The audio and the KG differ in modality, as well as availability. For example, while song audio is available for every song in the catalogue, a KG such as the one we use may represent the most famous songs well, but it may fail to represent properly more niche songs, which is a manifestation of the long-tail problem [KS16], and it may also fail to represent newly-released songs. To address this limitation, we complement the KG-based classifiers with the CNN-based classifiers, by formulating two hybrid classifiers.

The hybrids work by jointly running a KG-based classifier (KG-AVG or KG-SEQ) and a CNN-based classifier (CNN-AVG or CNN-SEQ), and by fusing the two playlist embedding vectors that they compute, before they are passed to a single-layered FF network that outputs a score for each listening context. We refer to HYBRID-AVG as the hybrid of KG-AVG & CNN-AVG and to HYBRID-SEQ as the hybrid of KG-SEQ & CNN-SEQ. The architecture of the two hybrids follow the schema of Figure 6.3.

For the embedding fusion, both the audio and KG-based playlist embedding vectors are input to two separate linear layers, two separate non-linearities, and then summed point-wise, as suggested by [BAM18]. We did experiment with other simple fusion strategies, e.g. concatenation, but they achieved lower performance.

6.2.5 Implementation details

Our implementation of CNN-AVG and CNN-SEQ is a little different from the original paper [CKE20] as we make two simplifications. First, we use Spotify's 30-second audio previews of the songs instead of their full audio. These audio previews are freely available, unlike the full audio, which is expensive to access due to copyright restrictions. Moreover, the usage of audio previews make our work reproducible. Second, we average the 3-second mel-spectrograms of a song point-wise in input to the CNN. As such, the CNN receives only one spectrogram, and outputs the song embedding directly. This second simplification saves computing resources. In Section 6.3, we show that our implementation of the CNN-based models outperforms the MF-based models, which is consistent with the original paper. More specifically, our implementation of CNN-SEQ achieves 7% higher performance than MF-SEQ, which is consistent with the original paper; similarly for CNN-AVG and MF-AVG. Given those results, we are confident that our implementations of the CNN-based models, although simplified, are as valid as the original implementations presented in [CKE20].

We compute the mel-spectrograms for the CNN-AVG and CNN-SEQ classifiers with 22,050 Hz sampling rate, 1,024 FFT size, 512 hop size, and 128 mel bins. We set hyper-parameters of the MF and CNN-based classifiers as in the original paper [CKE20]. That is, we set the song embedding dimension to 50, and we use *ReLU* as the non-linearity. We do the same in the KG-based and hybrid classifiers. We train the classifiers with early-stopping, monitoring *FH@1* on the validate set, with patience equal to ten. We tune other hyper-parameters of the eight classifiers (learning rate, weight decay and batch size) using Bayesian optimisation [VM21]. We fix the number of trials of the Bayesian optimiser to 20. For the WR-MF and TRANS-D embedding procedures, we use the default parameters and we set the number of epochs to ensure convergence of the loss function.

For other implementation details, we refer the reader to the source code that supports our work here.

6.3 Experiments

We compare the classifiers described in Section 6.2, and variants of those, on their performance in predicting the listening context of music playlists.

6.3.1 Dataset

We use a dataset of playlists annotated with their listening contexts. The dataset was annotated by the authors of [CKE20], starting from user playlists contained in the Spotify Million Playlist Dataset (MPD) [CLSZ18], and retaining only the portion of playlists that have a listening context as title[4]. Examples of listening contexts present in the dataset are: driving, studying and summertime. For other examples, we refer the reader to the dataset that supports our work here. Also, we refer the reader to [CKE20] for more information on the annotation procedure. Each playlist is annotated with one listening context. We split the dataset randomly into train, validate and test sets, accounting respectively for 60%, 20% and 20% of the total playlists. Similar to [CKE20], we filter out songs that occur in the validate and test sets but not in the train set, as some classifiers cannot handle at testing time songs not seen at training time. The classifiers that have this limitation are MF-AVG, MF-SEQ, KG-AVG and KG-SEQ. They work by training a *song embedding extractor* model in a first step, separately from the classifier that outputs the listening context, see Sections 6.2.1 and 6.2.3. As a result, embeddings for songs not present at training time are not available at test time. In a real world scenario, where new releases are frequently added to the songs catalogue, it would be necessary to incrementally train the models so that the training set covers all songs in the catalogue. An alternative is to use CNN-AVG and CNN-SEQ, as they rely on the audio signal, which is available for songs not seen at training time.

Table 6.1 contains statistics of the dataset that we use (train, validate and test splits together).

6.3.2 Metrics

We call D the test set, and we call p a playlist in the test set, that is $p \in D$. We call $|D|$ the number of playlists in the test set. The classifiers described in

[4]The dataset we use is not the one used in [CKE20], which is proprietary, but it was supplied to us by the authors of [CKE20] as a dataset annotated with the same procedure, and in which similar results can be obtained.

Table 6.1: Dataset statistics.

Statistic	Value
Number of playlists	114,689
Average playlist length	62.6
Number of unique songs	418,767
Number of unique listening contexts	102

Section 6.2 predict a score for each listening context. As such, given a playlist, a classifier predicts a ranking of listening contexts, by decreasing score. Given a ranking of listening contexts for a playlist p, we call $rank_p$ the position of the correct listening context in the ranking. For example, if a classifier assigns the highest score to the correct listening context, then $rank_p = 1$. Instead, if the classifier assigns the lowest score to the correct listening context, then $rank_p = 102$ (see Table 6.1).

We compare the classifiers for their performance in predicting the listening contexts of the playlists in D. We measure performance with four metrics, as in [CKE20]:

Flat hits (*FH@1*, *FH@5*) Flat hits is the percentage of playlists D such that $rank_p \leq k$. In our case, since the goal is classification rather than retrieval, we consider only $k = 1$ and $k = 5$ and no higher values for k. In formulas:

$$FH@k = \frac{1}{|D|} \sum_{p \in D} \mathbb{1}(rank_p \leq k)$$

where $\mathbb{1}(rank_p \leq k)$ is the indicator function. That is, $\mathbb{1}(rank_p \leq k) = 1$ if $rank_p \leq k$ and 0 otherwise. In other words, *FH@1* is the percentage of playlists for which the classifier predicts the listening context correctly. And, *FH@5* is the percentage of playlists for which the classifier predicts the correct listening context among the first five predictions.

Mean reciprocal rank (*MRR*) The reciprocal rank is the reciprocal of $rank_p$. The *MRR* is the average of those reciprocals ranks. In formulas:

$$MRR = \frac{1}{|D|} \sum_{p \in D} \frac{1}{rank_p}.$$

Mean average precision (*MAP@5*) *MAP* is equivalent to *MRR*, except that we

Table 6.2: Performance of the classifiers.

	FH@1	*FH@5*	*MRR*	*MAP@5*
MF-AVG	0.299	0.536	0.416	0.386
MF-SEQ	0.327	0.595	0.452	0.423
CNN-AVG	0.291	0.583	0.425	0.395
CNN-SEQ	0.352	0.639	0.484	0.456
KG-AVG	0.388	0.678	0.521	0.495
KG-SEQ	0.389	0.678	0.520	0.494
HYBRID-AVG	**0.395**	**0.687**	**0.528**	**0.503**
HYBRID-SEQ	0.389	0.678	0.520	0.495

set the reciprocal rank to 0 when $rank_p > k$[5]. That is, if $rank_p > k$ for every $p \in D$, then $MAP@k = 0$. In our case, we consider $k = 5$. In formulas:

$$MAP@k = \frac{1}{|D|} \sum_{p \in D} \frac{1}{rank_p} \times \mathbb{1}(rank_p \leq k).$$

On the one hand, *FH@k* disregards the actual position of the correct listening context in the ranking, but counts how frequently this position is lower than a threshold k. On the other hand, *MAP@k* and *MRR* do account for the actual position of the correct listening context in the ranking. Therefore, these metrics give a multi-sided view of the classifiers' performance.

We set up significance tests to check whether differences in performance are statistically significant or not. Following [DBSR18], we set up a t-test for *MRR* and *MAP@5*, and a paired bootstrap test for *FH@1* and *FH@5*. Similar to [Koe04], we fix the number of bootstrap replicas to 1000.

6.3.3 Results

We conduct two experiments: a comparison with the state-of-the-art, and a sensitivity analysis.

6.3.3.1 Comparison with state-of-the-art

We measure the performance of the classifiers that we propose (KG-AVG, KG-SEQ, HYBRID-AVG, HYBRID-SEQ), and the performance of the state-of-the-art

[5]Our formulation of MAP is different from others, which allow for multiple relevant items. In our case, there is only one relevant item: the correct listening context.

baselines, i.e. the existing listening context classifiers (MF-AVG, MF-SEQ, CNN-AVG, CNN-SEQ). The results are in Table 6.2.

The classifiers that we propose outperform the baselines by a considerable amount. HYBRID-AVG scores highest performance, improving by approximately 10% over the baselines. The improvement in performance is statistically significant ($p < 10^{-4}$). In general, all the classifiers we propose improve performance over the baselines ($p < 10^{-4}$).

The improvement in performance has real world relevance. For example, HYBRID-AVG achieves 12% higher *FH@1* than the best baseline (0.395 *vs* 0.352), which means than in a sample of 1000 playlists, our algorithm predicts the listening context correctly 395 *vs* 352 times, on average. Considering that the current databases contain millions of playlists, the 12% increase over the baselines is particularly 'tangible'.

We notice that the more complex SEQ variants of the algorithms are not always superior to their simpler AVG variant. MF-SEQ and CNN-SEQ have higher performance than, respectively, MF-AVG and CNN-AVG ($p < 10^{-4}$). But we do not find any statistically significant differences between the performance of KG-AVG and KG-SEQ, while HYBRID-SEQ has lower performance than HYBRID-AVG ($p < 10^{-4}$). Probably, the architecture of HYBRID-SEQ is too complex for the task at hand, and may overfit the training set, while the simpler HYBRID-AVG generalises better to new data. Moreover, the result corroborates previous work [CKE20], where the SEQ variant is found to be sometimes superior and sometimes inferior to the AVG variant.

The hybrid classifiers are the combination of the (audio) CNN-based and KG-based classifiers. Accordingly, HYBRID-AVG has higher performance than CNN-AVG and KG-AVG. Though statistically significant ($p < 10^{-4}$), the increase in performance is only slight. We can understand the result by looking at the literature on the well-studied task of music similarity [AP02c]. Flexer [Fle14] shows that increasing the performance of similarity algorithms is particularly challenging after a certain threshold, as there exists an upper bound to performance, caused by the low agreement of different users in the perception of music similarity [GB21b]. Likewise, humans can have different perceptions of the right listening context for a given playlist. In the dataset we use, each song is associated with 17 different playlist listening contexts, on average. As such, we expect that increasing the performance of classifiers can become particularly challenging after a certain threshold. For example, HYBRID-SEQ has higher

Table 6.3: Performance of KG-based classifiers with (w) and without (wo) song metadata.

	FH@1	*FH@5*	*MRR*	*MAP@5*
KG-AVG wo metadata	0.375	0.665	0.507	0.481
KG-AVG w metadata	0.388	0.679	0.521	0.495
KG-SEQ wo metadata	0.382	0.668	0.513	0.487
KG-SEQ w metadata	0.388	0.679	0.520	0.495

performance than CNN-SEQ ($p < 10^{-4}$), but not over KG-SEQ (no statistically significant difference).

6.3.3.2 Sensitivity analysis

KG-based classifiers have as input a KG with songs, playlists, their listening contexts (portion G_i) and song metadata (portion G_m). We measure the performance of variants of the KG-based classifiers that have as input only the portion G_i of the full KG. The results are in Table 6.3, and show an increase in performance when using metadata ($p < 10^{-4}$). This indicates that the KG-based classifiers make effective use of song metadata for predicting listening contexts. However, the increase in performance is only slight, and again can be explained by the work of Flexer [Fle14], as in Section 6.3.3.1.

The G_i portion of the KG contains the same information as the input to the MF-based classifiers, i.e. playlist listening contexts. However, as argued in Section 6.2.3, MF-based classifiers suffer from what we called the playlist short-circuiting problem, i.e. they model the association of songs to playlist listening contexts directly, while KG-based classifiers do not. A comparison of the results of the KG-based classifiers without metadata in Table 6.3 and the MF-based classifiers in Table 6.2 reveals the consequences of these two ways of modelling the information. The comparison shows that the KG-based algorithms exploit that information more effectively, since their results are significantly superior to those of the MF-based algorithms ($p < 10^{-4}$).

6.4 Conclusion

In this Chapter, we proposed four novel systems for predicting the listening contexts of music playlists, which include, for the first time, song metadata in their models. In two of them, we represent songs, playlists, listening contexts and song metadata in a KG, that we embed, and we use the song embeddings to make predictions. In the other two, we combine the KG and song audio in a unique hybrid model. We benchmark the performance of the predictors we propose, reporting an increase in performance of approximately 10% over the state-of-the-art. We also show, through a sensitivity analysis, that the KG-based predictors can incorporate the song metadata effectively. We argued that the improvement in performance that we have achieved has real world relevance.

Chapter 7

Playlist captioning

7.1 Introduction

In the last part of this dissertation, to which this Chapter belongs, we focus on playlist-level intelligibility[1]. In the previous Chapter, we achieved playlist-level intelligibility by assigning tags, drawn from a fixed vocabulary, to playlists. In this Chapter, we achieve playlist-level intelligibility by captioning, that is describing a playlist using natural language [CFS16].

Playlist captioning can be seen as a more general problem than playlist tagging. In fact, while tagging assigns one or more tags to a playlist, in playlist captioning, playlists are described in natural language, possibly including one or more tags, linked together in a fully-formed and coherent sentence. Captioning may be more suited than tagging for closing the semantic gap between audio and language, which is the key to achieving intelligibility. This is especially because playlists are sometimes centered around elaborate themes, see Section 2.6.1, 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-charting Jamaican songs in the UK. Natural language, instead, allows for precise characterisation of playlists at a high semantic level. However, captioning is a tough problem, whose solution may require massive quantities of training data and compute resources. Tagging, by contrast, might be more feasible in data or compute-bounded scenarios. Because of their pluses and minuses, we believe that the problems of

[1]This chapter is based on work done by Giovanni Gabbolini in collaboration with Elena Epure and Romain Hennequin from Deezer. See Section 1.3 for a statement of the division of contributions.

tagging and captioning deserve to coexist, and hence we cover both of them in this dissertation.

Related to playlist captioning is captioning of other kinds of objects. In fact, automated captioning is an active research field that has attracted much attention in recent years. We can find attempts to caption images [SCB+22], videos [GGZ+17], audio [DLV20] and music [MBQF21]. Here we review works in audio and music captioning, which are closest to our contribution.

Audio captioning focuses on identifying the human-perceived information in a general audio signal and expressing it through natural language [DLV20]. For example, an audio caption is: "a door creaks as it slowly revolves back and forth". [KMN+20] propose a transformer architecture for audio captioning that is similar to the original transformer for machine translation [VSP+17]. The input audio is embedded with a pre-trained Convolutional Neural Network (CNN) [HCE+17]. The embeddings are input to a transformer encoder. Then, a transformer decoder is tasked with generating the target caption.

While audio captioning is concerned with general audio signals, music captioning deals with music audio signals. [MBQF21] propose a Recurrent Neural Network (RNN) encoder-decoder to tackle single song captioning. The multimodal encoder takes as input the song audio embedding and the caption embedding up to token t. The song embedding is obtained with a pre-trained CNN [PNP+18]. The caption embedding is obtained with a pre-trained word2vec (w2v)-like model. The decoder is tasked with generating token $t+1$. Our work here is also on music captioning, but instead of captioning single songs we caption a sequences of songs, i.e. playlists.

Our interest, as stated above, is playlist captioning. We are aware of two other contributions on playlist captioning: [CFM+16] and [DLN21]. These two works are afflicted by two main limitations.

The first limitation is poor data quality. Existing work on playlist captioning uses datasets of user playlists, i.e. playlists created by users for personal use [DLN21, ZSLC19]. An alternative to user playlists are editorial playlists, i.e. playlists created by professional editors for a public audience. [CBF06] find that user playlists may not have a strictly defined theme, while the editorial ones usually do. Therefore, user playlists may not be an optimal source to learn *representative* captions, especially considering that the theme is regarded as a common playlist descriptor [KBJ20].

The second limitation is the semantic gap. The playlist captioning algorithms in [CFM$^+$16] and [DLN21] take in track embeddings as input, and strive to generate a correct caption as output. [CFM$^+$16] represent tracks as audio embeddings; and [DLN21] represent them as random embeddings indexed by track id and updated by back-propagation. In both cases, the low-level input embeddings and the high-level output caption are separated by a "semantic gap" [CS08] that the algorithms are tasked to close. Closing the semantic gap is challenging, especially because playlists may be built around themes not easily deducible from embeddings alone. For example, an Ireland playlist and a UK playlist could be confused given the cultural similarity these countries share. Artist themed playlists, e.g. "100% The Beatles", are even more difficult to caption due to data sparsity. In the whole dataset, an artist can be absent or be mentioned only in some playlists, making it difficult for existing models to learn track-to-artist mappings that could be exploited to produce relevant captions [SG18].

In this work, we propose PLAYNTELL, a new multi-modal, data-efficient [Ada21] playlist captioning model that overcomes the above limitations by leveraging linguistic and musical knowledge, to generate English-language thematic captions. First, PLAYNTELL narrows the semantic gap by using musical knowledge. In particular, it leverages user tags, e.g. "Ireland", "rock" and "90s", that provide information at the same high semantic level as the expected captions[2]. We also introduce an ad-hoc strategy to deal with artist themed playlists, by masking artist mentions, and supplying the PLAYNTELL encoder with an artist distribution vector.

Second, we train PLAYNTELL on a new high-quality dataset of editorial playlists assembled from two major music streaming services, which we release together with the code[3]. However, editorial playlists have the drawback of sparsity. Our dataset in particular is composed of only a few thousand samples. Training PLAYNTELL from scratch to generate correct natural language captions with suhc a dataset is challenging [WLS19]. Inspired by existing work in computer vision [CGY$^+$22], we address this limitation by warm-starting the decoder with a pre-trained language model, GPT-2 [BMR$^+$20].

We validate PLAYNTELL with extensive quantitative experiments: PLAYNTELL outperforms existing playlist captioning algorithms, achieving 2x higher *BLEU@4*

[2]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.

[3]`https://github.com/deezer/playntell`

Table 7.1: Example of output generated by PLAYNTELL vs. the corresponding ground truth.

	Caption
PLAYNTELL	“80s smash hits the best tracks of the decade”
Ground truth	“all out 80s the biggest songs of the 1980s”

and 3x higher *CIDEr*. Also, we observe via qualitative evaluation that it can generate relevant editorial-like captions. We report in Table 7.1 a caption generated by PLAYNTELL and its ground truth. More examples can be found in Table 7.11.

In summary, our contributions are:

1. PLAYNTELL, a new multi-modal, data-efficient playlist captioning encoder-decoder model that leverages audio, linguistic and musical knowledge to generate thematic captions;
2. A new high-quality dataset of thematic playlists created by editors from two major music streaming services. We enrich each playlist with user tags automatically collected at track level from the crowdsourced database Discogs[4];
3. An extensive evaluation of PLAYNTELL, reporting 2x-3x higher *BLEU@4* and *CIDEr* than existing playlist captioning algorithms. We also provide a qualitative analyses of PLAYNTELL, as well as an ablation study and sensitivity analysis to validate the contribution of different modalities and model components, and a user study.

7.2 Dataset

Current public playlist captioning datasets contain user playlists, known to be noisier and sometimes not focused on a theme [DLN21, CBF06]. We address the above limitation by introducing a new dataset of thematic editorial playlists. We created this new dataset by collecting public editorial playlists from Spotify and Deezer, which are two major music streaming services[5].

Each playlist in the dataset consists of a sequence of track ids, a title and a

[4]https://data.discogs.com/

[5]https://midiaresearch.com/blog/announcing-midias-state-of-the-streaming-nation-2-report

Table 7.2: Dataset statistics.

	Statistic	Value
Deezer playlists	Number of playlists	5467
	Average caption length (words)	21.1
	Average playlist length (tracks)	47.8
	Number of unique words	20312
	Number of unique tracks	182638
Spotify playlists	Number of playlists	1104
	Average caption length (words)	16.8
	Average playlist length (tracks)	69.1
	Number of unique words	5551
	Number of unique tracks	60765

description. Both tracks, title and description are curated by a professional editor. Each track is associated with at least one artist.

Some of the playlists that we collected were extremely short, e.g. just two tracks. We get rid of these outliers by filtering out playlists where the number of tracks is below the fifth percentile. Some other playlists were extremely long, e.g. more than 200 tracks, or have long captions, e.g. more than 50 words. In practice, such outliers have the effect of increasing, respectively, the memory requirements and inference time of algorithms. For these reasons, we filter out playlists where the number of tracks is above the 95^{th} percentile, or where the caption length in words is above the 95^{th} percentile. We end up with 5467 Deezer playlists and 1104 Spotify playlists. We judge the playlists and their captions to be of high quality, so no further pre-processing is needed. We release both the raw and the filtered datasets with the code.

We present some statistics of the filtered editorial public playlists collected from Deezer and Spotify in Table 7.2.

We consider both the playlist title and its description to be part of its caption. In particular, a caption is defined by the template: *<title> [the title] <description> [the description].* An example is: *<title> 100% The Beatles <description> The best music from The Beatles.* Further examples of captions are given as Ground truths in Table 7.11. The choice of the template follows recent advances in few-shot learning, where the samples are enriched by task-specific tokens [LL21, ZWF+21]. We did experiment with other templates, e.g. *[the title] <sep> [the description]*, but with no difference in performance.

The captions we consider account for a variety of playlist themes, such as musical genres, moods, activities, events and artists, and were written by a number of professional editors coming from different cultural backgrounds. Also, the dataset statistics we report in Table 7.2 provide further evidence of the captions' linguistic diversity. For example, there are 5467 Deezer playlists, and their captions contain 20312 unique words.

We partition the Deezer playlists randomly in training, validation and test, accounting respectively for 60%, 20% and 20% of the total. Validation and test splits allow internal evaluation, i.e. on in-distribution samples. We use the Spotify playlists as an additional test set for an external evaluation, i.e. on out-of-distribution samples.

PlayNTell, the model we propose, is designed to bridge the semantic gap between a playlist and its corresponding caption by leveraging different sources of musical knowledge, including track audio and user tags. For audio, we retrieve 30-second long audio previews of the tracks. Audio previews are convenient because they can be freely accessed from the public APIs of music streaming services. For user tags, we resort to the crowdsourced database Discogs, which offers user tags at album level. We propagate those user tags to every track in the album. The user tags cover a range of aspects: genres, countries, years and moods (e.g. "pop", "italy", "2020" and "happy"). We release the tracks' user tags with the captioned playlists.

Track audio and user tags differ in availability. The existence of audio is implied by the existence of a track. But user tags may not be available for a new release, for example, just because no user has tagged it yet. The model we propose leverages user tags if available, and, if not, can rely on audio only to generate sensible captions, as we show in Section 7.4.3.

7.3 Method

Editorial data are available in small quantities, as their existence depends on the expensive work of professional editors. In our case, we can rely on a few thousand editorial playlists, as we explained in Section 7.2, which is a challenging setting in which to learn a model from scratch. In this Section, we describe PlayNTell, our playlist captioning model that leverages linguistic and musical knowledge to cope with sparsity of captions.

PlayNTell is composed of an encoder and decoder, shown in the left and right

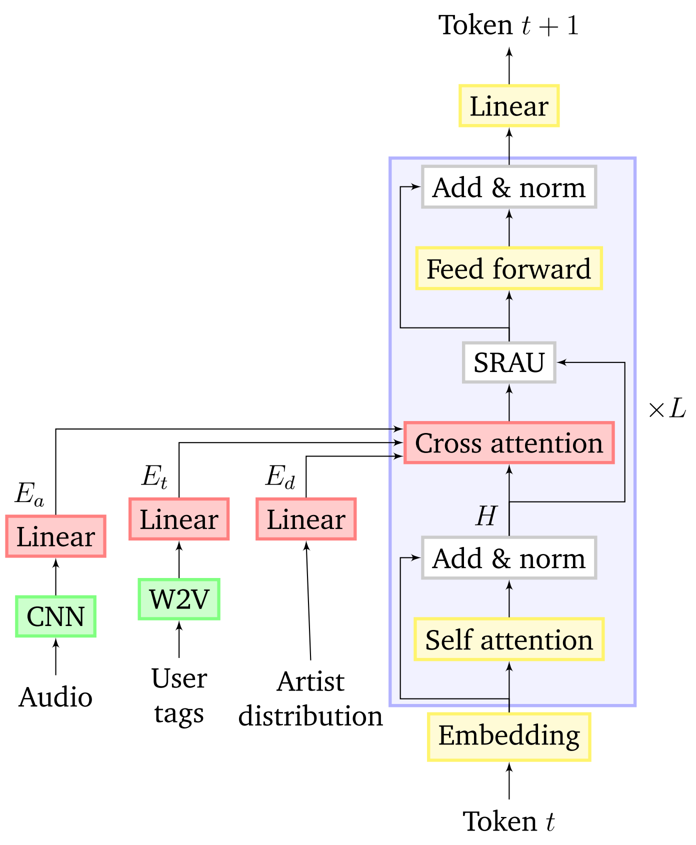


Figure 7.1: PLAYNTELL – model architecture.

hand-sides of Figure 7.1.

7.3.1 Encoder

The encoder has three branches to handle three types of musical knowledge — audio, user tags and artist distribution.

7.3.1.1 Audio

Track audio is commonly used in playlist tagging [CKE20] and playlist captioning [CFM+16]. For every track, we retrieve 30-second long audio previews, as detailed in Section 7.2, and create an audio embedding by using a pre-trained CNN. The CNN architecture is VGGish [PS19]. The CNN extracts a 256-dimensional embedding vector for every three seconds of audio. This gives ten embeddings for each 30 second track, which we average. For a playlist with P tracks, we then obtain a $P \times 256$ matrix, which we transform into a $P \times h$ matrix using a learned linear layer, where h is the output dimension of this linear layer.

7.3.1.2 User tags

Audio and captions are separated by a wide semantic gap that algorithms may struggle to close. Therefore, we propose to inform PLAYNTELL with user tags, e.g. "pop", "happy", which provide information at the same semantic level as captions. For every track in a playlist, we retrieve user tags from Discogs, as in Section 7.2. We consider the playlist user tags to be the union of all the distinct track user tags. We embed user tags with a pre-trained word2vec-like model specific for music-related text, called music-w2v [DLPN20]. Music-w2v embeddings are 300-dimensional. We embed all playlist user tags to obtain a 300-dimensional matrix, which we transform to a h-dimensional matrix using a learned linear layer.

7.3.1.3 Artist distribution

Some playlists are themed by artist, e.g. "100% The Beatles". [KBM12, CBF06] report that artists are a common organisation principle among playlist-makers. While audio and user tags inform the model with "general" musical knowledge, e.g. genres and styles, they may not help to detect artist themed playlists, which is challenging even with large scale datasets [RLHTM18]. As a remedy, we introduce the artist distribution vector. It has as i^{th} value the share of tracks in the playlist authored by the i^{th} most popular artist in the playlist. For example, if a playlist has one track by John Lennon and 99 tracks by The Beatles, the vector is $[0.99, 0.01]$. In tracks authored by multiple artists, we consider only the main artist for simplicity. We limit the length of the vector to ten and pad it when necessary. We experimented with higher vector length, without any performance gain. We project the vector to a h dimensional space using a learned linear layer.

The output of each encoder branch is 0-padded to a common number of rows N, to obtain the three embedding matrices E_a, E_t, $E_d \in \mathbb{R}^{N\times h}$.

CNN and w2v embeddings are frozen, i.e. not updated during training. This is similar to the state-of-the-art in image captioning, where the input image is embedded by means of a pre-trained and frozen CNN, before being fed to a transformer-like model tasked with generating the output caption [CSBC20, HKBS19, GLZ$^+$20].

7.3.2 Decoder

The decoder is very similar to a transformer decoder [VSP+17]. An attention function is at its core. Given matrices $Q \in \mathbb{R}^{n_q \times d}$, $K \in \mathbb{R}^{n_k \times d}$ and $V \in \mathbb{R}^{n_k \times d}$, representing query, key and value, the attention is defined as:

$$Att(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d}}\right)V. \tag{7.1}$$

Attention computes a weighted sum of V's rows according to the similarity between Q's and K's rows. In practice we implement the multi-head variant of attention [VSP+17].

The decoder is composed of three parts: input, hidden and output. In the input part, we use learned token embeddings to convert a caption token to an embedding of dimension h. In the output part, we use a learned linear layer with softmax to produce predicted next token probabilities. In practice, we provide as input all caption tokens shifted right, and we predict all next tokens in parallel. The hidden part is made up of L identical layers. Every layer is composed of three sub-layers: self-attention, cross-attention, and feed-forward.

7.3.2.1 Self-attention & feed forward

We apply the attention function using the same matrix as query, key and value in the self-attention layer. The attention function is modified so as to avoid prediction of token t depending on subsequent tokens. The feed-forward layer consists of a fully connected neural network [VSP+17]. We wrap each layer around a residual connection [HZRS16] and a normalisation layer [BKH16].

7.3.2.2 Cross-attention

We apply the attention function using the encoder output as key and value, and the self-attention output H as query. Our encoder has three outputs. We apply the attention function to every output separately and then sum the results, similar to [ZSLS19]. Then:

$$\begin{aligned} Crs\text{-}att(H, E_a, E_t, E_d) = {} & Att(H, E_a, E_a) \\ & + Att(H, E_t, E_t) \\ & + Att(H, E_d, E_d). \end{aligned} \tag{7.2}$$

We wrap the cross-attention layer around a self-resurrecting activation unit (SRAU) layer, proven effective in data-efficient image captioning [CGY+22].

We inject linguistic knowledge in the decoder, similar to [CGY+22]. GPT-2 [BMR+20] has a transformer-like architecture, compatible with PLAYNTELL. We load pre-trained GPT-2 (small) weights in the layers of the decoder: embedding, self-attention, feed-forward and linear. These layers are fine-tuned during training. The choice of GPT-2 follows previous work [CGY+22]. Other pre-trained decoders, such as T5 [RSR+20] or BART [LLG+20], could be used. The choice of GPT-2 fixes the decoder hyper-parameters to $N = 12$ transformer layers, 12 attention heads and hidden size $h = 768$. As a result, the three linear layers in the decoder are tasked to project from respectively 256, 300 and 10 dimensional spaces to a 768 dimensional space.

7.3.3 Artist masking

Correctly generating artist mentions is important, as artist themed playlists is a common category of editorial playlists [KBM12, CBF06]. However, correctly generating artist mentions is a particularly challenging task, mainly due to data sparsity. In the whole dataset, an artist can be absent or be mentioned only in some playlists, making it difficult for algorithms to learn a mapping between input data and artist mentions.

We introduce artist masking as a remedy to data sparsity, similar to [ZSLS19]. We pre-process the training captions by substituting artist mentions with placeholders. For example, the caption “100% The Beatles” is pre-processed as “100% artist1”. If a caption mentions more than one artist, we will have more than one placeholder (“artist1”, “artist2”, . . .). The mapping between placeholders and artists is decided in advance. We use popularity within the playlist: the author of most tracks in the playlist has placeholder “artist1”; the second most popular artist has placeholder “artist2”; and so on. We post-process the output captions by substituting placeholders with actual artist mentions. The artist masking strategy we present is designed to be used in conjunction with the artist distribution vector, which provides useful knowledge on how to generate artist placeholders.

7.4 Experiments

7.4.1 Experimental setting

7.4.1.1 Metrics

We adopt eight accuracy metrics: *BLEU@1* to *4* (*B@1* to *4*), *METEOR* (*M*), *ROUGE-L* (*R-L*), *CIDEr* (*C*) and *BERT-Score* (*B-S*) [CCG20]. The first seven are functions that are based on precision and recall of n-grams that are common to the generated caption and the corresponding ground truth caption. *BERT-Score* exploits pre-trained BERT embeddings to represent and match the tokens in the ground truth with respect to the generated caption. We use *BERT-Score* with recall and *idf* weighting, which is the suggested configuration [ZKW$^+$20].

We also adopt two diversity metrics. *% novel* is the percentage of generated captions that are not among the training captions. *Vocab* is the number of unique words used in all the generated captions.

Following [DBSR18], we set up a t-test for *ROUGE-L*, *CIDEr* and *BERT-Score*, and a paired bootstrap test for *BLEU@1* to *4* and *METEOR*. Following [Koe04], we fix the number of bootstrap replicas to 1000.

7.4.1.2 Implementation details

We convert a caption to tokens using Byte Pair Encoding (BPE) [SHB16]. We use learned positional embeddings to distinguish the order of tokens.

We optimise the model using simple cross-entropy loss, computed on the generated caption against the ground truth. During inference, the prediction of the previous time step is fed to the input of the next time step. We use a beam search of size three to compute the most likely output sequence.

We train the models with the AdamW optimizer [LH18]. We use a learning rate equal to 10^{-4} and a batch size equal to 10. We use early stopping with patience equal to 40. We set threshold τ of the SRAU gate to 0.2, as recommended in [CGY$^+$22].

7.4.2 Comparison with state-of-the-art

7.4.2.1 Baselines

We compare PLAYNTELL with a simple NN (Nearest Neighbor) baseline and with state-of-the-art music captioning algorithms:

NN Given a test playlist p, find the training playlist $\tilde{p}$ closest (using cosine distance) to the test playlist, and output $\tilde{p}$'s caption as p's caption. A playlist is represented as the average audio embedding of its tracks. Track audio embeddings are computed as in Section 7.3.1.

MUSCAPS We adapt the song captioning model proposed in [MBQF21] to do playlist captioning, by replacing song audio embeddings with playlist audio embeddings. We take a playlist audio embedding to be the average of the audio embeddings of the songs in the playlist. We use the authors' implementation with default parameters.

DOHRNN and **DOHTRA** We adopt the seq2seq and transformer models for playlist captioning proposed in [DLN21]. We use the authors' implementation with default parameters.

We use as training set the Deezer dataset training split. We use as test sets the Spotify dataset and the Deezer dataset test split. We assess the algorithms according to accuracy and diversity metrics.

The accuracy results are reported in Table 7.3. PLAYNTELL largely outperforms all baselines, achieving 2x higher *BLEU@4* and 3x higher *CIDEr*. The large improvement on *CIDEr* may be due to the artist mentions, as *CIDEr* is particularly sensitive to infrequent words [VLZP15]. While the baselines may struggle, PLAYNTELL takes advantage of the musical knowledge in terms of artist distribution to correctly generate artist mentions. We can notice smaller differences of *BERT-Score*. This is expected, as *BERT-Score* is known to assume values in a narrow range [ZKW+20]. We test the significance of differences in accuracy, as explained in Section 7.4.1. The differences are statistically significant ($p < 10^{-4}$).

The Spotify dataset is used for external validation. We observe quite high values for the metrics, but lower overall than for the Deezer test set. As expected, this is due to differences in "style" between the two streaming platforms. Artist themed playlists in Spotify are captioned as, e.g., "This is The Beatles"; in Deezer, we would have "100% The Beatles". And the most common words in

	Deezer								Spotify							
	B@1	*B@2*	*B@3*	*B@4*	*M*	*R-L*	*C*	*B-S*	*B@1*	*B@2*	*B@3*	*B@4*	*M*	*R-L*	*C*	*B-S*
NN	18.7	6.2	3.4	2.4	16.6	17.4	11.4	81.2	15.8	3.5	1.2	0.5	17.4	17.1	4.2	81.5
MusCaps	18.7	8.9	4.9	2.6	17.8	23.8	12.2	77.2	19.0	6.3	2.3	0.9	17.4	19.6	6.8	75.8
DohRNN	19.6	8.5	4.7	2.5	16.7	21.1	7.0	80.0	18.5	5.0	1.7	0.5	17.3	18.5	2.6	80.5
DohTra	20.6	8.8	5.1	3.2	17.2	20.5	12.0	81.1	18.3	4.4	1.4	0.4	17.4	18.0	2.4	81.1
PlayNTell	**29.0**	**18.9**	**14.4**	**11.8**	**22.9**	**32.7**	**114.9**	**84.4**	**23.4**	**9.0**	**4.3**	**2.3**	**19.8**	**22.9**	**26.3**	**82.8**

Table 7.3: State-of-the-art accuracy as measured on the Deezer and Spotify test sets. Bold indicates the most accurate algorithm. PlayNTell scores 2x-3x higher *BLEU@4* and *CIDEr* than the baselines.

	Deezer								Spotify							
	B@1	*B@2*	*B@3*	*B@4*	*M*	*R-L*	*C*	*B-S*	*B@1*	*B@2*	*B@3*	*B@4*	*M*	*R-L*	*C*	*B-S*
NN	4.4	2.7	1.8	1.3	2.1	4.2	12.6	81.2	2.4	1.1	0.0	0.0	1.4	2.8	6.1	81.5
MusCaps	10.8	8.1	2.3	0.0	5.9	13.1	20.0	77.2	5.2	1.0	0.0	0.0	2.2	5.1	9.0	75.8
DohRNN	8.9	7.2	3.3	2.0	4.5	10.1	13.1	80.0	1.7	0.6	0.0	0.0	0.6	1.5	2.7	80.5
DohTra	8.7	6.3	2.6	1.4	4.1	8.7	12.4	81.1	0.8	0.3	0.0	0.0	0.3	0.8	1.4	81.1
PlayNTell	**34.6**	**30.3**	**27.9**	**25.4**	**19.0**	**34.4**	**218.7**	**84.4**	**12.8**	**7.5**	**4.0**	**2.3**	**6.8**	**12.8**	**42.7**	**82.8**

Table 7.4: State-of-the-art accuracy as measured in the Deezer and Spotify test sets in titles only. Bold indicates the most accurate algorithm.

	Deezer								Spotify							
	B@1	*B@2*	*B@3*	*B@4*	*M*	*R-L*	*C*	*B-S*	*B@1*	*B@2*	*B@3*	*B@4*	*M*	*R-L*	*C*	*B-S*
NN	10.4	4.3	2.6	1.9	3.9	8.8	11.7	81.2	7.6	2.0	0.7	0.3	3.1	7.4	3.7	81.5
MusCaps	10.5	5.0	2.6	1.5	4.2	14.3	13.9	77.2	9.2	3.6	1.4	0.5	3.1	9.8	5.3	75.8
DohRNN	10.7	4.6	2.4	1.5	3.0	11.3	6.7	80.0	8.4	2.7	1.0	0.3	2.8	8.0	2.5	80.5
DohTra	11.7	5.4	3.2	2.3	3.9	11.4	13.1	81.1	9.0	2.9	1.1	0.0	3.1	8.2	3.1	81.1
PlayNTell	**17.3**	**10.9**	**8.1**	**6.6**	**8.1**	**20.4**	**73.8**	**84.4**	**11.2**	**4.3**	**1.9**	**0.8**	**4.2**	**10.9**	**14.7**	**82.8**

Table 7.5: State-of-the-art accuracy as measured in the Deezer and Spotify test sets in descriptions only. Bold indicates the most accurate algorithm.

Table 7.6: State-of-the-art diversity as measured on the Deezer and Spotify test sets.

	Deezer		Spotify	
	% novel	*Vocab*	*% novel*	*Vocab*
NN	0.0	1750	0.0	2100
MUSCAPS	100.0	66	100.0	60
DOHRNN	100.0	838	100.0	764
DOHTRA	100.0	2015	100.0	1767
PLAYNTELL	97.6	2585	98.3	2147

Spotify captions are: “cover”, “from”, “tracks”; in Deezer, they are: “by”, “best”, “music”.

The results on diversity are reported in Table 7.6. NN scores 0 in *% novel*, as NN can only ‘generate’ captions from the training set. By contrast, the music captioning baselines only generate novel captions, hence they score 100 in *% novel*. This may seem surprising, as captioning algorithms are known sometimes to generate novel captions, and other times to replicate training captions [SCB+22]. We believe that the we are recording 100% novelty because of problems the music captioning baselines have in generating natural text. This prevents the music captioning baselines from replicating any of the training captions. For example, we find that MUSCAPS often generates incorrect text, such as the caption: *<title> 100% jazz <description> the best of the best of the best music*. We provide more evidence of such problems in Section 7.4.5.

Baselines score modest *Vocab*s. For example, MUSCAPS’s *Vocab* in the Deezer dataset is only 2% of PLAYNTELL’s *Vocab*. MUSCAPS’s *Vocab* may be low because the hyper-parameters are not optimised for the dataset. PLAYNTELL, on the other hand, can replicate a proportion of its training captions, i.e. *% novel* $<$ 100, and has the largest *Vocab*.

Finally, we assess the algorithms considering titles and descriptions separately. That is, we use as training set the Deezer dataset training split; we use as test sets the Spotify dataset and the Deezer dataset test split; we isolate title and description from the generated caption and ground truth, and we compute accuracy and diversity metrics, in both datasets, for title and description separately. The results are reported respectively in Tables 7.4, 7.5, 7.7, 7.8. The results corroborate those in Tables 7.3 and 7.6, where we considered the full captions, using a template that combines titles and descriptions (*<title> the*

Table 7.7: State-of-the-art diversity as measured on the Deezer and Spotify test sets in titles only.

	Deezer		Spotify	
	% novel	*Vocab*	*% novel*	*Vocab*
NN	0.0	540	0.0	644
MUSCAPS	85.8	39	90.8	37
DOHRNN	89.5	393	89.7	381
DOHTRA	41.9	900	34.6	785
PLAYNTELL	72.5	1257	65.9	959

Table 7.8: State-of-the-art diversity as measured on the Deezer and Spotify test sets in descriptions only.

	Deezer		Spotify	
	% novel	*Vocab*	*% novel*	*Vocab*
NN	0.0	1608	0.0	1939
MUSCAPS	98.6	41	99.0	37
DOHRNN	100.0	579	100.0	511
DOHTRA	100.0	1567	100.0	1375
PLAYNTELL	88.7	2025	93.2	1671

title <description> the description).

7.4.3 Ablation study

PLAYNTELL is informed by three sources of musical knowledge: audio, user tags, and artist distribution. We consider two variations to the architecture of PLAYNTELL in order to check whether the three sources are actually exploited. The first variation has only the audio branch; the second has both audio and artist distribution branches. We compare the two variations with the original PLAYNTELL, which features also the user tags branch. In the first variation we do not use the artist masking strategy, while in the second variation and the original PLAYNTELL we do.

We measure accuracy with the same training-evaluation setup as Section 7.4.2. The results are in Table 7.9. The accuracy increases with the number of input modalities, on both datasets. The differences are statistically significant in the Deezer dataset and in the Spotify dataset ($p < 0.05$). Thus, we have a good indication that PLAYNTELL can successfully leverage all three sources of musical knowledge.

	Deezer								Spotify							
	B@1	*B@2*	*B@3*	*B@4*	*M*	*R-L*	*C*	*B-S*	*B@1*	*B@2*	*B@3*	*B@4*	*M*	*R-L*	*C*	*B-S*
Audio	23.2	11.9	7.6	5.2	19.5	25.7	27.8	82.4	20.8	6.2	2.5	1.1	18.7	20.9	12.9	82.0
+ artist	27.3	17.4	13.2	10.8	22.1	30.7	105.7	84.0	22.9	8.6	4.0	2.0	19.5	22.2	24.1	82.7
Rand init	21.0	12.0	8.8	7.0	18.5	24.4	72.6	82.0	17.9	5.4	2.4	1.2	17.2	17.9	16.3	81.0
PlayNTell	**29.0**	**18.9**	**14.4**	**11.8**	**22.9**	**32.7**	**114.9**	**84.4**	**23.4**	**9.0**	**4.3**	**2.3**	**19.8**	**22.9**	**26.3**	**82.8**

Table 7.9: Accuracy with one and two input modalities and with random initialisation compared to PlayNTell (3 modalities and initialized with GPT-2) on the Deezer and Spotify test sets. Bold indicates highest accuracy.

	Deezer								Spotify							
	B@1	*B@2*	*B@3*	*B@4*	*M*	*R-L*	*C*	*B-S*	*B@1*	*B@2*	*B@3*	*B@4*	*M*	*R-L*	*C*	*B-S*
Rand init	21.0	12.0	8.8	7.0	18.5	24.4	72.6	82.0	17.9	5.4	2.4	1.2	17.2	17.9	16.3	81.0
GPT-2 init	**29.0**	**18.9**	**14.4**	**11.8**	**22.9**	**32.7**	**114.9**	**84.4**	**23.4**	**9.0**	**4.3**	**2.3**	**19.8**	**22.9**	**26.3**	**82.8**

Table 7.10: Accuracy with random and GPT-2 initialisation, as measured on the Deezer and Spotify test sets. Bold indicates the most accurate algorithm.

7.4.4 Sensitivity analysis

We investigate the impact of the GPT-2 initialisation on the generated captions. We consider a variation of PLAYNTELL that has random decoder initialisation, which we name RAND INIT. We measure accuracy with the same training-evaluation setup as Section 7.4.2. The results are in Table 7.10.

PLAYNTELL largely outperforms RAND INIT, achieving 63% higher *BLEU@4* and 71% higher *CIDEr*. The differences are statistically significant on both Deezer and Spotify datasets ($p < 10^{-4}$). We observe that PLAYNTELL produces syntactically correct captions, while RAND INIT does not, e.g. for one playlist PLAYNTELL generates the caption *<title> relaxing piano <description> relax with calm classical tunes*, while RAND INIT generates the caption: *<title> relaxing music < piano>description <> the of music the playlist classicalind.* Thus, we have a clear indication that the GPT-2 initialisation has a positive impact on the generated captions, and that the linguistic knowledge acquired by GPT-2 is leveraged by PLAYNTELL in the playlist captioning task.

7.4.5 Discussion

The baselines differ from PLAYNTELL along three axes: input data, linguistic knowledge and architecture. The baselines leverage only track audio or learned embeddings. PLAYNTELL accommodates three input modalities, being informed by track audio, user tags and artist distribution. Comparing the results in Table 7.9 with the baselines in Table 7.3, we notice that PLAYNTELL, when informed by audio only, is still consistently superior to the baselines.

Similarly, compared to the baselines, PLAYNTELL leverages linguistic knowledge held by GPT-2 weights. However, when comparing the performance of PLAYNTELL without GPT-2 weights (RAND INIT in Table 7.10) with the baselines in Table 7.3, we notice that RAND INIT outperforms the baselines with in-distribution data (Deezer dataset). With out-of-distribution data (Spotify dataset), RAND INIT outperforms all the baselines for some metrics (e.g. *BLEU@1*, *CIDEr*), but only some baselines for some other metrics (e.g. *BLEU@1*). Thus, there is evidence that our novel encoder, which is the main architectural difference between PLAYNTELL and the baselines, is enough to outperform the baselines, at least with in-distribution data. With out-of-distribution data, the contribution of the encoder may be blurred by overfitting in the small in-distribution dataset, as PLAYNTELL has a far more complex decoder than the baselines. Hence, we expect that, when trained on a larger in-distribution dataset, PLAYNTELL can outperform the baselines in both in- and out-of-distribution data, without using GPT-2 weights.

We show captions generated by the algorithms on the Spotify test set when trained on the Deezer training set and the corresponding ground truths in Table 7.11. Captions produced by the baselines are not syntactically correct, probably because the dataset is too small to learn a sound language model. On the other hand, PLAYNTELL can take advantage of the pre-trained GPT-2 weights and generate syntactically correct captions.

Captions produced by the baselines are not always semantically correct. In the first example, none of the baselines matches the playlist theme. In the second example, MUSCAPS aligns with the playlist theme, while DOHRNN and DOHTRA do not. MUSCAPS holds pre-trained audio knowledge that may be helping in this example. PLAYNTELL, on the other hand, takes advantage of musical knowledge (audio, user tags, and artist distribution), and matches the theme in both cases. The third example features a playlist of “desi” music[6] (traditional south-asian music). None of the algorithms produces a caption that matches this theme, probably because the input data do not properly represent the concept of “desi” music. For example, the “desi” user tag is not among the playlist user tags we use as input to PLAYNTELL. This example highlights the western-centered perspective of Music Information Retrieval research, which is a debated topic [HSH21]. The dataset we employ features playlists of mainly western music. As such, PLAYNTELL can find the right caption for a playlist of, e.g., UK pop-rock made in the early ’00s, but it struggles for a playlist of, e.g., traditional south-asian music, or “desi” music.

The NN baseline only outputs captions from the training set. So, by design, NN produces syntactically correct captions. NN is aligned with the playlist theme in just one of the three examples. Although simple, NN is a competitive baseline.

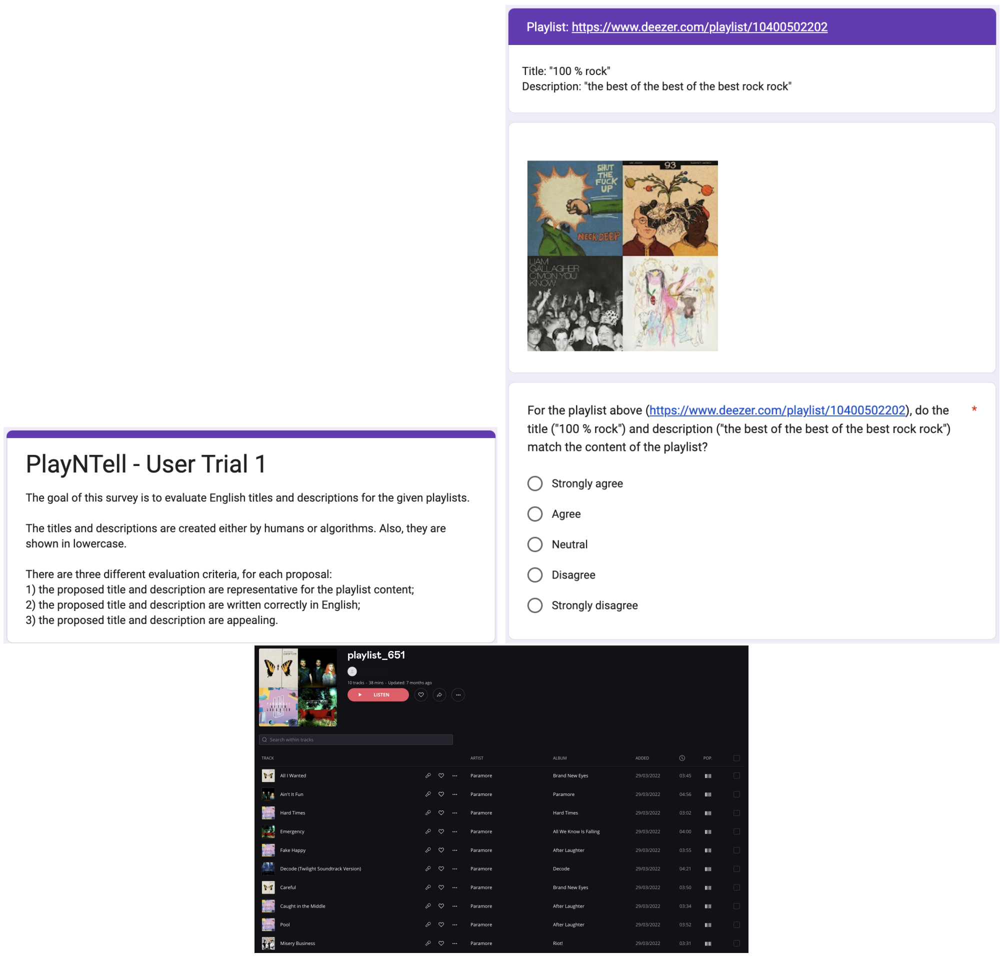
7.4.6 User study

We conducted a user study to evaluate PLAYNTELL. The study takes the form of a survey, in which the participants were asked to rate playlist captions on three different aspects: (1) *Content match*: Do the title and description match the playlist content? (2) *English correctness*: Are the title and description correct in

[6]`https://en.wikipedia.org/wiki/Desi`

	Caption
NN	*<title> best of calm piano 2020 <description> the best of the neo classical scene*
MUSCAPS	*<title> 100 % jazz <description> the best of the best of the best music*
DOHRNN	*<title> boogie # 2016 <description> all the best songs from the best recordings*
DOHTRA	*<title> rock 101 <description> the best of the decade in this playlist # pulse # pulse*
PLAYNTELL	*<title> relaxing piano <description> relax with calm classical tunes*
Ground truth	*<title> peaceful piano <description> relax and indulge with beautiful piano pieces*
NN	*<title> top trending hip hop <description> a collection of hip hop hits and viral trends that 's updated weekly*
MUSCAPS	*<title> 100 % rock <description> the best of the best of the best of the best of the best of the best rock*
DOHRNN	*<title> chill fi <description> <description> the hottest hottest tracks to the hottest focus %*
DOHTRA	*<title> an introduction to the best of <description> the best british music that have made the great playlists %*
PLAYNTELL	*<title> 100 % ramones <description> enjoy the kings of rock this is ramones collection*
Ground truth	*<title> this is ramones <description> this is ramones the essential tracks all in one playlist*
NN	*<title> trap & bass by spinnin records <description> trap & bass by the finest 808 suppliers*
MUSCAPS	*<title> top hits hits <description> the best tracks of the best tracks*
DOHRNN	*<title> tiktok party <description> <description> the playlist <description> the the the most party hits*
DOHTRA	*<title> 100 % queen latifah <description> all the best songs of the amazing lebanese tracks*
PLAYNTELL	*<title> rising african hits <description> the best african tracks of our favorite african pop music in one playlist*
Ground truth	*<title> desi hits <description> desi hits from south asia cover badshah*
NN	*<title> best of dance 2020 <description> top dance tracks of 2020*
MUSCAPS	*<title> 80s hits <description> the best hits of the best hits*
DOHRNN	*<title> best of <description> the best of of the best of the best of the %*
DOHTRA	*<title> acoustic pop <description> your favorite tracks %*
PLAYNTELL	*<title> 90s pride <description> the best tracks of the decade to celebrate all colours of love*
Ground truth	*<title> all out 90s <description> the biggest songs of the 1990s*
NN	*<title> mashups | club mix <description> smash hit club & chart remixes sure to get the party started*
MUSCAPS	*<title> dance hits <description> the best tracks of the best tracks*
DOHRNN	*<title> workout workout <description> <description> the the love with the finest and love %*
DOHTRA	*<title> 100 % the 1975 <description> it 's the essential tracks to keep your little bit %*
PLAYNTELL	*<title> dance music <description> the biggest dance tracks out there*
Ground truth	*<title> power hour <description> tap it back or go for a spin with these uptempo tracks*
NN	*<title> japanese k pop <description> listen to the japanese rendition of your favorite k pop songs*
MUSCAPS	*<title> top hits <description> the best hits of the best hits in one playlist*
DOHRNN	*<title> lo fi <description> <description> the best to the best and best music %*
DOHTRA	*<title> 100 % the script <description> need by the best tracks from the legend %*
PLAYNTELL	*<title> k pop party <description> experience the best k pop music in hifi quality*
Ground truth	*<title> sing along k pop <description> fancy belting out your favourite korean songs*

Table 7.11: Ground truth and generated captions on the Spotify dataset.



Playlist: https://www.deezer.com/playlist/10400502202

Title: "100 % rock"
Description: "the best of the best of the best rock rock"

For the playlist above (https://www.deezer.com/playlist/10400502202), do the title ("100 % rock") and description ("the best of the best of the best rock rock") match the content of the playlist? *

- Strongly agree
- Agree
- Neutral
- Disagree
- Strongly disagree

PlayNTell - User Trial 1

The goal of this survey is to evaluate English titles and descriptions for the given playlists.

The titles and descriptions are created either by humans or algorithms. Also, they are shown in lowercase.

There are three different evaluation criteria, for each proposal:
1) the proposed title and description are representative for the playlist content;
2) the proposed title and description are written correctly in English;
3) the proposed title and description are appealing.

playlist_651

10 tracks - 38 mins - Updated: 7 months ago

TRACK	ARTIST	ALBUM	ADDED		POP.
All I Wanted	Paramore	Brand New Eyes	29/03/2022	03:45	
Ain't It Fun	Paramore	Paramore	29/03/2022	04:56	
Hard Times	Paramore	Hard Times	29/03/2022	03:02	
Emergency	Paramore	All We Know Is Falling	29/03/2022	04:00	
Fake Happy	Paramore	After Laughter	29/03/2022	03:55	
Decode (Twilight Soundtrack Version)	Paramore	Decode	29/03/2022	04:21	
Careful	Paramore	Brand New Eyes	29/03/2022	03:50	
Caught in the Middle	Paramore	After Laughter	29/03/2022	03:34	
Pool	Paramore	After Laughter	29/03/2022	03:52	
Misery Business	Paramore	Riot!	29/03/2022	03:31	

Figure 7.2: User survey instructions, questions and playlist presentation.

English? (3) *Appeal*: Are the title and description appealing? Participants could respond to each question by choosing from five options on a Likert scale. We used as inspiration for these questions user trial designs in image captioning, where (1) and (2) were often assessed [ZSLS19].

We recruited seven participants. Each participant was a Deezer playlist editor, thus guaranteeing that participants had sufficient musical knowledge to accurately assess a playlist caption.

The participants were shown the first ten songs of curated playlists. The songs were shown as a playlist page on the Deezer website thus including the song title, the artist name and the album title, see Figure 7.2. These playlists were sampled randomly from the Deezer test set. Each participant was presented with seven different playlists (with no overlap between participants).

For each of these playlists, a participant was asked to rate five different captions: the ground truth caption (the original caption assigned to the playlist

Table 7.12: Means and standard deviations of Likert scores (in $[1, 5]$) and p-values obtained from a paired t-test between the considered method and PLAYNTELL scores.

Method	Mean	Std.	p-value
Content match			
PLAYNTELL	3.20408	1.27442	Ref
NN	1.63265	0.83401	<1e-3
MUSCAPS	2.04082	0.99915	<1e-3
DOHTRA	1.55102	0.86750	<1e-3
GroundTruth	3.665306	1.199844	0.066
English correctness			
PLAYNTELL	3.14286	1.32288	Ref
NN	3.67347	0.92168	0.019
MUSCAPS	2.34694	1.09070	0.002
DOHTRA	2.73469	1.15064	0.094
Ground Truth	3.83673	1.23063	0.004
Appeal			
PLAYNTELL	2.87755	1.14805	Ref
NN	3.22449	1.00551	0.071
MUSCAPS	1.97959	0.87773	<1e-3
DOHTRA	2.30612	0.96186	0.005
Ground Truth	3.44898	1.19131	0.013

by a human editor) and captions generated by each of four methods described in this Chapter: MUSCAPS, DOHTRA, NN and PLAYNTELL. We did not include DOHRNN as its offline results are comparable to DOHTRA's, see Section 7.4.2. Excluding DOHRNN reduces the burden that we place on the participants.

This user study gives us a comparison between PLAYNTELL and the ground truth and baselines. It serves to assess how far our algorithm is from, respectively, human editors, considered as the golden standard, and the state-of-the-art in playlist captioning.

Results are shown in Table 7.12. The first thing to be noted is that the ground truth captions get scores that are quite far from the perfect score of 5. One reason that could explain this phenomenon is that playlist captioning is intrinsically subjective. Then, the NN baseline has the best results in terms of *Appeal* and *English correctness* with metrics close to the ground truth. This is not surprising as NN is the only baseline that does not generate captions but outputs captions written by humans but associated with other playlists. However, the

NN baseline is not able to output captions that match the content of the associated playlist as shown by the *Content match* metric. On the other hand, PLAYNTELL outperforms all baselines in terms of *Content match*, with a score that is significantly higher than all other baselines and not significantly lower than the ground truth. Also, PLAYNTELL is the captioning method that comes second for *Appeal* and *English correctness* after NN with no significant difference for *Appeal*. PLAYNTELL brings a significant improvement in terms of *Appeal* compared to other generative models, and over MUSCAPS in terms of *English correctness*.

These results tend to confirm that PLAYNTELL is able to generate realistic captions, matching the content of the playlists with a significant improvement over the state-of-the-art.

7.5 Conclusion

Our work adds to the literature on playlist captioning. We present both a new dataset and a new model that sets a new state-of-the-art. PLAYNTELL leverages general linguistic knowledge from a pre-trained language model to generate coherent captions, and musical knowledge to make the captions consistent with the playlist content.

With this Chapter, we conclude the presentation of our contributions to playlist intelligibility. The next Chapter concludes the dissertation, by highlighting promising avenues for future work.

Chapter 8

Conclusions and future work

In this dissertation, we explore the concept of the intelligibility of music playlists, at two different levels: song-level and playlist-level. We introduce several algorithms, some operating at song-level, and some others operating at playlist level, which implement intelligibility in practice. We evaluate the algorithms with offline experiments and user studies. We find evidence that the algorithms can help accomplish the two goals of intelligibility, i.e. enhancing listening experiences, and facilitating organisation and access. In particular, we find that users would welcome the tours produced by our song-level intelligibility algorithms, which highlights the potential for enhanced listening experiences. We also find that tags and captions produced by our playlist-level intelligibility algorithms accurately describe the content of playlists, more accurately than state-of-the-art algorithms, which could be used in systems for playlists organisation and access.

We identify possible improvements to the algorithms we propose, which present opportunities for short-term future work. Song-level intelligibility algorithms may be improved by better interestingness modelling. Interestingness is a scoring function for segues, at the heart of the mechanism for constructing tours. At the moment, the interestingness function is not personalised. However, we find compelling evidence in Chapter 5 that different users may have different perceptions of interestingness. For example, a knowledgeable user may be interested in highly specific segues, as they may already be aware of more general segues. By contrast, a casual user may find highly specific segues to be obscure, and may prefer general segues.

There are several ways of building a personalised version of interestingness.

One way is to set the weights used by the $interestingness$ function in a personalised way, and in particular to learn personalized weights, that would fit the tastes of the user. Another idea, guided by the finding that users find known segues to be less interesting (Chapter 5), is to develop a mechanism to separate known segues from novel segues. Then, in the $interestingness$ function, known segues would be pensalised and the novel segues would be rewarded. Separating known and unknown segues can be done explicitly, by asking users directly, or implicitly, by building a personalised classifier. Hybrid strategies are also possible, e.g. ask users whether they know some segues or not, and use their answers to improve a classifier, which is a mechanism similar to active learning [RKS11].

We also identify improvements that can be made to the algorithms for playlist-level intelligibility. For example, the listening context tagger we introduce in Chapter 6 could be applied to a broader set of tags, for example including moods and genres, by using the recently released Melon dataset [FKL+21]. And, the playlist captioning algorithm we introduce in Chapter 7, PLAYNTELL, can also be improved. There is anecdotal evidence that PLAYNTELL fails to caption non-western music. This bias can be overcome by ensuring that playlists from different parts of the world are equally represented in the training dataset. Finding such datasets might be challenging, however, given the general western-centered perspective of the whole field of Music Information Retrieval [HSH21].

As well as short-term future work, we envision some longer-term future work, under the idea of employing the intelligibility algorithms for other down-stream tasks or analysis. For example, the playlist-level intelligibility algorithms describe music at a high semantic level, and these descriptions may be used within explanations of playlist recommendations made by recommender systems in streaming services [AMS+22]. Similarly, segues in playlists may be used to introduce unfamiliar music to the user, starting from other songs in the playlist that the user already knows.

We would be particularly interested in exploring the relationship between these two applications of intelligibility and the concept of serendipity. Serendipity and explainability are two active research topics in recommender systems. However, the intersection of these two topics, and especially the impact of explanations on serendipity, is largely unexplored. In recommender systems research, serendipity is usually defined by three components: novelty, unexpectedness and relevance. It is relatively easy to achieve novelty and unexpectedness. For example, including a random song in a playlist would probably achieve novelty and unexpectedness. It is more difficult to achieve relevance, especially for unexpected items. This is known as the unexpectedness-relevance trade-off. We argue that intelligibility can help increase the perceived relevance of unexpected items, and thus can help increase serendipity. In Chapter 3, we found that segues can spark interest in items, which leads credence to our argument[1].

Another longer term avenue for future work is the testing of intelligibility algorithms ‘in the wild’. Throughout this dissertation, we strongly rely on user trials, which offer user-centered evaluations of the algorithms, but they are conducted in artificial settings. It would be extremely interesting to implement intelligibility algorithms in a real streaming service, and gather user feedback. Just before submitting this dissertation, the streaming service Spotify launched a new service, called “DJ”, with the promise to “deliver a curated lineup of music alongside commentary around the tracks and artists we think you’ll like”[2]. The idea behind Spotify’s DJ resonates with the concept of song-level intelligibility, and provides evidence of the potential impact of the research we have presented in this dissertation.

[1]We described the relationships between explanations and serendipity in an abstract we presented at the “Symposium on Serendipity and Recommender Systems”, held online from the 22nd to the 25th November 2021. `https://theserendipitysociety.wordpress.com/symposium-serendipity-and-recommender-systems/`.

[2]Quoted from: `https://newsroom.spotify.com/2023-02-22/spotify-debuts-a-new-ai-dj-right-in-your-pocket/`

References

[ABC$^+$18] Sebastiano Antenucci, Simone Boglio, Emanuele Chioso, Ervin Dervishaj, Shuwen Kang, Tommaso Scarlatti, and Maurizio Ferrari Dacrema. Artist-driven layering and user's behaviour impact on recommendations in a playlist continuation scenario. In *Proceedings of the ACM Recommender Systems Challenge*, pages 1–6, 2018.

[ABTU22] Gediminas Adomavicius, Konstantin Bauman, Alexander Tuzhilin, and Moshe Unger. Context-aware recommender systems: From foundations to recent developments. In *Recommender Systems Handbook*, pages 211–250. Springer, 2022.

[Ada15] William C Adams. Conducting semi-structured interviews. *Handbook of Practical Program Evaluation*, 4:492–505, 2015.

[Ada21] Amina Adadi. A survey on data-efficient algorithms in big data era. *Journal of Big Data*, 8(1):1–54, 2021.

[AH05] Andreja Andric and Goffredo Haus. Estimating quality of playlists by sight. In *Proceedings of the International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution*. IEEE, 2005.

[AH06] Andreja Andric and Goffredo Haus. Automatic playlist generation based on tracking user's listening habits. *Multimedia Tools and Applications*, 29(2):127–151, 2006.

[AKS12] Natalie Aizenberg, Yehuda Koren, and Oren Somekh. Build your own music recommender by modeling internet radio streams. In *Proceedings of the International Conference on World Wide Web*, pages 1–10, 2012.

[AMHWA^{+}05] Boanerges Aleman-Meza, Christian Halaschek-Wiener, I. Budak Arpinar, Cartic Ramakrishnan, and Amit P. Sheth. Ranking complex relationships on the semantic web. *IEEE Internet Computing*, 9(3):37–44, 2005.

[AMNS02] Paolo Avesani, Paolo Massa, Michele Nori, and Angelo Susi. Collaborative radio community. In *Proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 462–465, 2002.

[AMS^{+}22] Darius Afchar, Alessandro Melchiorre, Markus Schedl, Romain Hennequin, Elena Epure, and Manuel Moussallam. Explainability in music recommender systems. *AI Magazine*, 43(2):190–208, 2022.

[Anb18] Caroline Anbuhl. *Social and Cultural Practices around Using the Music Streaming Provider Spotify-A qualitative study exploring how German Millennials use Spotify (Master's degree thesis)*. Malmö universitet/Kultur och samhälle, 2018.

[AÖ02] Fredrik Axelsson and Mattias Östergren. Soundpryer: Joint music listening on the road. In *Consuming Music Together*, 2002.

[AP02a] J-J Aucouturier and François Pachet. Scaling up music playlist generation. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, volume 1, pages 105–108, 2002.

[AP02b] Jean-Julien Aucouturier and Francois Pachet. Finding songs that sound the same. In *Proceedings of IEEE Benelux Workshop on Model based Processing and Coding of Audio*, pages 1–8, 2002.

[AP02c] Jean-Julien Aucouturier and Francois Pachet. Music similarity measures: What's the use? In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 13–17, 2002.

[AT00] Masoud Alghoniemy and Ahmed H Tewfik. User-defined music sequence retrieval. In *Proceedings of the ACM International Conference on Multimedia*, pages 356–358, 2000.

[AT01] Masoud Alghoniemy and Ahmed H Tewfik. A network flow model for playlist generation. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, 2001.

[AWMC17] David Paul Allen, Henry Jacob Wheeler-Mackta, and Jeremy R Campo. The effects of music recommendation engines on the filter bubble phenomenon. *Interactive Qualifying Projects*, 2017.

[BAB06] Jacob T Biehl, Piotr D Adamczyk, and Brian P Bailey. Djogger: a mobile dynamic music device. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, pages 556–561, 2006.

[BAM18] Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. Multimodal machine learning: A survey and taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(2):423–443, 2018.

[Bas04] Sumit Basu. Mixing with mozart. In *Proceedings of the International Computer Music Conference*, 2004.

[BBB10] Dominikus Baur, Sebastian Boring, and Andreas Butz. Rush: repeated recommendations on mobile devices. In *Proceedings of the International Conference on Intelligent User Interfaces*, pages 91–100, 2010.

[BC06] Virginia Braun and Victoria Clarke. Using thematic analysis in psychology. *Qualitative Research in Psychology*, 3(2):77–101, 2006.

[BCB15] Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*, 2015.

[BCC17] Alejandro Bellogín, Pablo Castells, and Iván Cantador. Statistical biases in information retrieval metrics for recommender systems. *Information Retrieval Journal*, 20(6):606–634, 2017.

[BCR$^+$22] Théo Bontempelli, Benjamin Chapus, François Rigaud, Mathieu Morlon, Marin Lorant, and Guillaume Salha-Galvan. Flow moods: Recommending music by moods on deezer. In *Proceedings of the ACM Conference on Recommender Systems*, pages 452–455, 2022.

[BELK$^+$17] Shay Ben-Elazar, Gal Lavee, Noam Koenigstein, Oren Barkan, Hilik Berezin, Ulrich Paquet, and Tal Zaccai. Groove radio: A bayesian hierarchical model for personalized playlist generation. In *Proceedings of the ACM International Conference on Web Search and Data Mining*, pages 445–453, 2017.

[BF20] Christine Bauer and Bruce Ferwerda. Conformity behavior in group playlist creation. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, pages 1–10, 2020.

[BGH$^+$17] Rachel M Bittner, Minwei Gu, Gandalf Hernandez, Eric J Humphrey, Tristan Jehan, Hunter McCurry, and Nicola Montecchio. Automatic playlist sequencing and transitions. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 442–448, 2017.

[BH13] James D Belcher and Paul Haridakis. The role of background characteristics, music-listening motives, and music selection on music discussion. *Communication Quarterly*, 61(4):375–396, 2013.

[BHBB11] Dominikus Baur, Bernhard Hering, Sebastian Boring, and Andreas Butz. Who needs interaction anyway: exploring mobile playlist creation from manual to automatic. In *Proceedings of the International Conference on Intelligent User Interfaces*, pages 291–294, 2011.

[BJ13] Geoffray Bonnin and Dietmar Jannach. Evaluating the quality of playlists based on hand-crafted samples. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 263–268, 2013.

[BJ14] Geoffray Bonnin and Dietmar Jannach. Automated generation of music playlists: Survey and experiments. *ACM Computing Surveys (CSUR)*, 47(2):1–35, 2014.

[BJC11] Jared S Bauer, Alex Jansen, and Jesse Cirimele. Moodmusic: a method for cooperative, generative music playlist creation. In *Proceedings of the Annual ACM Adjunct Symposium on User Interface Software and Technology*, pages 85–86, 2011.

[BJK18] Jared S Bauer, Aubury L Jellenek, and Julie A Kientz. Reflektor: An exploration of collaborative music playlist creation for social context. In *Proceedings of the ACM Conference on Supporting Groupwork*, pages 27–38, 2018.

[BKH16] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[BMA06] Arianna Bassoli, Julian Moore, and Stefan Agamanolis. tuna: Socialising music sharing on the move. In *Consuming Music Together*, pages 151–172. Springer, 2006.

[BMEL$^+$11] Thierry Bertin-Mahieux, Daniel PW Ellis, EE LabROSA, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings of the International Society for Music Information Retrieval Conference*, 2011.

[BMR$^+$20] 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. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, 2020.

[BMT$^+$19] Morteza Behrooz, Sarah Mennicken, Jennifer Thom, Rohit Kumar, and Henriette Cramer. Augmenting music listening experiences on voice assistants. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 303–310, 2019.

[BNJ03] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022, 2003.

[BOL09] Luke Barrington, Reid Oda, and Gert RG Lanckriet. Smarter than genius? human evaluation of music recommender systems. In *Proceedings of the International Society for Music Information Retrieval Conference*, volume 9, pages 357–362, 2009.

[BP06] Claudio Baccigalupo and Enric Plaza. Case-based sequential ordering of songs for playlist recommendation. In *Proceedings of the European Conference on Case-Based Reasoning*, pages 286–300, 2006.

[BP07] Claudio Baccigalupo and Enric Plaza. Sharing and combining listening experience: a social approach to web radio. In *Proceedings of the International Conference on Computer Music*, 2007.

[BPK09] Klaas Bosteels, Elias Pampalk, and Etienne E Kerre. 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*, volume 9, pages 351–356, 2009.

[BSDK12] Karan Kumar Budhraja, Ashutosh Singh, Gaurav Dubey, and Arun Khosla. Probability based playlist generation based on music similarity and user customization. In *Proceedings of the National Conference on Computing and Communication Systems*, pages 1–5, 2012.

[Bur02] Robin Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.

[BvdH10] Wietse Balkema and Ferdi van der Heijden. Music playlist generation by assimilating GMMs into SOMs. *Pattern Recognition Letters*, 31(11):1396–1402, 2010.

[BWT$^+$19] Dmitry Bogdanov, Minz Won, Philip Tovstogan, Alastair Porter, and Xavier Serra. The MTG-Jamendo Dataset for Automatic Music Tagging. In *Proceedings of the Machine Learning for Music Discovery Workshop at the International Conference on Machine Learning*, 2019.

[CA09] Zeina Chedrawy and Syed Sibte Raza Abidi. A web recommender system for recommending, predicting and personalizing music playlists. In *International Conference on Web Information Systems Engineering*, pages 335–342, 2009.

[Cas06] Michael Castelluccio. The music genome project. *Strategic Finance*, pages 57–59, 2006.

[CB09] Ya-Xi Chen and Andreas Butz. MusicSim: Integrating Audio Analysis and User Feedback in an Interactive Music Browsing UI. In *Proceedings of the International Conference on Intelligent User Interfaces*, pages 429–434, 2009.

[CBB17] Sally Jo Cunningham, David Bainbridge, and Annette Bainbridge. Exploring personal music collection behavior. In *Proceedings of the International Conference on Asian Digital Libraries*, pages 295–306. Springer, 2017.

[CBF05] Dennis L Chao, Justin Balthrop, and Stephanie Forrest. Adaptive radio: achieving consensus using negative preferences. In *Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work*, pages 120–123, 2005.

[CBF06] Sally Jo Cunningham, David Bainbridge, and Annette Falconer. '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*, 2006.

[CBH02] Andrew Crossen, Jay Budzik, and Kristian J Hammond. Flytrap: Intelligent group music recommendation. In *Proceedings of the International Conference on Intelligent User Interfaces*, pages 184–185, 2002.

[CCC17] Chia-Hao Chung, Yian Chen, and Homer H Chen. Exploiting playlists for representation of songs and words for text-based music retrieval. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 478–485, 2017.

[CCG08] Stuart Cunningham, Stephen Caulder, and Vic Grout. Saturday night or fever? Context-aware music playlists. In *Proceedings of the International Audio Mostly Conference*, 2008.

[CCG20] Asli Celikyilmaz, Elizabeth Clark, and Jianfeng Gao. Evaluation of text generation: A survey. *arXiv preprint arXiv:2006.14799*, 2020.

[CDR13] Filipe Coelho, José Devezas, and Cristina Ribeiro. Large-scale crossmedia retrieval for playlist generation and song discovery. In *Proceedings of the Conference on Open Research Areas in Information Retrieval*, pages 61–64, 2013.

[CDW$^+$22] Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He†. Bias and debias in recommender system: A survey and future directions. *ACM Transactions on Information Systems*, 2022.

[Cel10] Oscar Celma. *Music recommendation and discovery: The long tail, long fail, and long play in the digital music space*. Springer Science & Business Media, 2010.

[CFM$^+$16] Keunwoo Choi, George Fazekas, Brian McFee, Kyunghyun Cho, and Mark Sandler. Towards music captioning: Generating music playlist descriptions. *arXiv preprint arXiv:1608.04868*, 2016.

[CFS15] Keunwoo Choi, George Fazekas, and Mark Sandler. Understanding music playlists. *arXiv preprint arXiv:1511.07004*, 2015.

[CFS16] Keunwoo Choi, George Fazekas, and Mark Sandler. Towards playlist generation algorithms using RNNs trained on within-track transitions. *arXiv preprint arXiv:1606.02096*, 2016.

[CFSC17] Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho. Convolutional recurrent neural networks for music classification. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pages 2392–2396, 2017.

[CG16] Tianqi Chen and Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the ACM SIGKDD International Conference on knowledge Discovery and Data Mining*, pages 785–794, 2016.

[CGY$^+$22] Jun Chen, Han Guo, Kai Yi, Boyang Li, and Mohamed Elhoseiny. Visualgpt: Data-efficient adaptation of pretrained language models for image captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18030–18040, 2022.

[Cha02] Amar Chaudhary. An extensible representation for playlists. In *Proceedings of the International Society for Music Information Retrieval Conference*, 2002.

[CJJ04] Sally Jo Cunningham, Matt Jones, and Steve Jones. Organizing digital music for use: an examination of personal music collections. In *Proceedings of the International Society for Music Information Retrieval Conference*, 2004.

[CKE20] Jeong Choi, Anis Khlif, and Elena Epure. Prediction of user listening contexts for music playlists. In *Proceedings of the Workshop on NLP for Music and Audio (NLP4MusA)*, pages 23–27, 2020.

[CKN$^+$11] Joseph A Calandrino, Ann Kilzer, Arvind Narayanan, Edward W Felten, and Vitaly Shmatikov. " you might also like:" privacy risks of collaborative filtering. In *Proceedings of the IEEE Symposium on security and privacy*, pages 231–246. IEEE, 2011.

[Cli00] Dave Cliff. Hang the dj: Automatic sequencing and seamless mixing of dance-music tracks. Technical report, HP Laboratories Technical Report, 2000.

[CLSZ18] Ching-Wei Chen, Paul Lamere, Markus Schedl, and Hamed Zamani. Recsys challenge 2018: Automatic music playlist continuation. In *Proceedings of the ACM Conference on Recommender Systems*, page 527–528, 2018.

[CLW09] Ching-Wei Chen, Kyogu Lee, and Ho-Hsiang Wu. Towards a class-based representation of perceptual tempo for music retrieval. In *Proceedings of the International Conference on Machine Learning and Applications*, pages 602–607, 2009.

[CMTJ12] Shuo Chen, Josh L Moore, Douglas Turnbull, and Thorsten Joachims. Playlist prediction via metric embedding. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 714–722, 2012.

[CN09] Sally Jo Cunningham and David M Nichols. Exploring social music behaviour: An investigation of music selection at parties. In *Proceedings of the International Society for Music Information Retrieval Conference*, 2009.

[CNBA14] Sally Jo Cunningham, David M Nichols, David Bainbridge, and Hasan Ali. Social music in cars. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 457–462, 2014.

[COKG11] Yandre MG Costa, Luiz S Oliveira, Alessandro L Koericb, and Fabien Gouyon. Music genre recognition using spectrograms. In *Proceedings of the International Conference on Systems, Signals and Image Processing*, pages 1–4, 2011.

[COWH20] Sylvia Chan-Olmsted, Rang Wang, and Kyung-Ho Hwang. Substitutability and complementarity of broadcast radio and music streaming services: The millennial perspective. *Mobile Media & Communication*, 8(2):209–228, 2020.

[CPP11] Luís Cardoso, Renato Panda, and Rui Pedro Paiva. Moodetector: A prototype software tool for mood-based playlist generation. In *Simpósio de Informática*, 2011.

[CS08] Òscar Celma and Xavier Serra. Foafing the music: Bridging the semantic gap in music recommendation. *Journal of Web Semantics*, 6(4):250–256, 2008.

[CS14] Zhiyong Cheng and Jialie Shen. Just-for-me: an adaptive personalization system for location-aware social music recommendation. In *Proceedings of International Conference on Multimedia Retrieval*, pages 185–192, 2014.

[CSBC20] Marcella Cornia, Matteo Stefanini, Lorenzo Baraldi, and Rita Cucchiara. Meshed-memory transformer for image captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10578–10587, 2020.

[CTLH10] Chung-Yi Chi, Richard Tzong-Han Tsai, Jeng-You Lai, and Jane Yung-jen Hsu. A reinforcement learning approach to emotion-based automatic playlist generation. In *Proceedings of the International Conference on Technologies and Applications of Artificial Intelligence*, pages 60–65. IEEE, 2010.

[CVER07] Michel Crampes, Jean Villerd, Andrew Emery, and Sylvie Ranwez. Automatic playlist composition in a dynamic music landscape. In *Proceedings of the International Workshop on Semantically aware document Processing and Indexing*, pages 15–20, 2007.

[CZHY22] Haonan Cheng, Ruyu Zhang, Xiaoying Huang, and Long Ye. Global-local similarity function for automatic playlist generation. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, pages 01–06. IEEE, 2022.

[CZS11] Luca Chiarandini, Massimiliano Zanoni, and Augusto Sarti. A system for dynamic playlist generation driven by multimodal control signals and descriptors. In *Proceedings of the IEEE International Workshop on Multimedia Signal Processing*, pages 1–6, 2011.

[CZZM07] Rui Cai, Chao Zhang, Lei Zhang, and Wei-Ying Ma. Scalable music recommendation by search. In *Proceedings of the ACM International Conference on Multimedia*, pages 1065–1074, 2007.

[DBAH$^+$18] Cedric De Boom, Rohan Agrawal, Samantha Hansen, Esh Kumar, Romain Yon, Ching-Wei Chen, Thomas Demeester, and Bart Dhoedt. Large-scale user modeling with recurrent neural networks for music discovery on multiple time scales. *Multimedia Tools and Applications*, 77(12):15385–15407, 2018.

[DBSR18] Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 1383–1392, 2018.

[DDC$^+$19] Yashar Deldjoo, Maurizio Ferrari Dacrema, Mihai Gabriel Constantin, Hamid Eghbal-Zadeh, Stefano Cereda, Markus Schedl, Bogdan Ionescu, and Paolo Cremonesi. Movie genome: alleviating new item cold start in movie recommendation. *User Modeling and User-Adapted Interaction*, 29(2):291–343, 2019.

[Dea21] Brian Dean. Spotify user stats, 2021. `https://backlinko.com/spotify-Users`.

[DF10] Ricardo Dias and Manuel J Fonseca. Muvis: an application for interactive exploration of large music collections. In *Proceedings of the ACM International Conference on Multimedia*, pages 1043–1046, 2010.

[DGF16] Ricardo Dias, Daniel Gonçalves, and Manuel J Fonseca. Playlistcreator: An assisted approach for playlist creation. In

Proceedings of the ACM International Conference on Multimedia, pages 711–713, 2016.

[DGF17] Ricardo Dias, Daniel Gonçalves, and Manuel J Fonseca. From manual to assisted playlist creation: a survey. *Multimedia Tools and Applications*, 76(12):14375–14403, 2017.

[DK04] Mukund Deshpande and George Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems*, 22(1):143–177, 2004.

[DLN21] Seungheon Doh, Junwon Lee, and Juhan Nam. Music playlist title generation: A machine-translation approach. In *Proceedings of the Workshop on NLP for Music and Spoken Audio (NLP4MuSA)*, 2021.

[DLPN20] Seungheon Doh, Jongpil Lee, Tae Hong Park, and Juhan Nam. Musical word embedding: Bridging the gap between listening contexts and music. In *Proceedings of the Machine Learning for Media Discovery Workshop (ML4DL) at the International Conference on Machine Learning*, 2020.

[DLV20] Konstantinos Drossos, Samuel Lipping, and Tuomas Virtanen. Clotho: An audio captioning dataset. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pages 736–740. IEEE, 2020.

[DMV97] AM De Mooij and WFJ Verhaegh. Learning preferences for music playlists. *Artificial Intelligence*, 97(1-2):245–271, 1997.

[dONLF19] Igor de Oliveira Nunes, Gabriel Matos Cardoso Leite, and Daniel Ratton Figueiredo. 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*, pages 72–77. IEEE, 2019.

[DP02] Clemens Drews and Florian Pestoni. Virtual jukebox: reviving a classic. In *Proceedings of the Annual Hawaii International Conference on System Sciences*, pages 887–893, 2002.

[DSPK08] Markus Dopler, Markus Schedl, Tim Pohle, and Peter Knees. Accessing music collections via representative cluster prototypes in a hierarchical organization scheme. In *Proceedings of the In-*

ternational Society for Music Information Retrieval Conference, pages 179–184, 2008.

[DWLX15] Shuiguang Deng, Dongjing Wang, Xitong Li, and Guandong Xu. Exploring user emotion in microblogs for music recommendation. *Expert Systems with Applications*, 42(23):9284–9293, 2015.

[DWXG20] Yuanfei Dai, Shiping Wang, Neal N Xiong, and Wenzhong Guo. A survey on knowledge graph embedding: Approaches, applications and benchmarks. *Electronics*, 9(5):750, 2020.

[EJMS76] Honey S Elovitz, Rodney W Johnson, Astrid McHugh, and John E Shore. Automatic translation of english text to phonetics by means of letter-to-sound rules. Technical report, Naval Reseach Lab Washington DC, 1976.

[Ell06] Daniel PW Ellis. Extracting information from music audio. *Communications of the ACM*, 49(8):32–37, 2006.

[ET06] Greg T Elliott and Bill Tomlinson. Personalsoundtrack: context-aware playlists that adapt to user pace. In *Extended Abstracts of the CHI Conference on Human factors in Computing Systems*, pages 736–741, 2006.

[FBY$^+$18] Andres Ferraro, Dmitry Bogdanov, Jisang Yoon, KwangSeob Kim, and Xavier Serra. Automatic playlist continuation using a hybrid recommender system combining features from text and audio. In *Proceedings of the ACM Recommender Systems Challenge*, pages 1–5, 2018.

[FD21] Marco Furini and Sabrina D'arcangelo. Automatic and user-tailored playlist sequencing. In *Proceedings of the Conference on Information Technology for Social Good*, pages 321–324, 2021.

[FKL$^+$21] Andres Ferraro, Yuntae Kim, Soohyeon Lee, Biho Kim, Namjun Jo, Semi Lim, Suyon Lim, Jungtaek Jang, Sehwan Kim, and Xavier Serra. 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*, pages 536–540. IEEE, 2021.

[FL10] Ben Fields and Paul Lamere. Finding a path through the juke box: The playlist tutorial. In *Proceedings of the International Society for Music Information Retrieval Conference*, 2010.

[Fle14] Arthur Flexer. On inter-rater agreement in audio music similarity. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 245–250, 2014.

[FLR21] Arthur Flexer, Taric Lallai, and Katja Rašl. On evaluation of inter-and intra-rater agreement in music recommendation. *Proceedings of the International Society for Music Information Retrieval Conference*, 4(1), 2021.

[Fly16] Mathew Flynn. Accounting for listening: How music streaming has changed what it means to listen. *Kinephanos-Journal of Media Studies and Popular Culture*, 6(1), 2016.

[FM22] Marco Furini and Manuela Montangero. Automatic and personalized sequencing of music playlists. In *Proceedings of the IEEE International Conference on Distributed Computing Systems Workshops*, pages 205–208. IEEE, 2022.

[FMM19] Marco Furini, Jessica Martini, and Manuela Montangero. Automated generation of user-tailored and time-sensitive music playlists. In *Proceedings of the IEEE Annual Consumer Communications & Networking Conference*, pages 1–6. IEEE, 2019.

[FPA18] Guglielmo Faggioli, Mirko Polato, and Fabio Aiolli. Efficient similarity based methods for the playlist continuation task. In *Proceedings of the ACM Recommender Systems Challenge*, pages 1–6, 2018.

[FRCJ08] Benjamin Fields, Christophe Rhodes, Michael A Casey, and Kurt Jacobson. 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*, pages 559–564, 2008.

[Fre08] Ariana Moscote Freire. Remediating radio: Audio streaming, music recommendation and the discourse of radioness. *Radio*

Journal: International Studies in Broadcast & Audio Media, 5(2-3):97–112, 2008.

[FŞA+20] Dieter Fensel, Umutcan Şimşek, Kevin Angele, Elwin Huaman, Elias Kärle, Oleksandra Panasiuk, Ioan Toma, Jürgen Umbrich, and Alexander Wahler. Introduction: what is a knowledge graph? In *Knowledge Graphs*, pages 1–10. Springer, 2020.

[FSGW08] Arthur Flexer, Dominik Schnitzer, Martin Gasser, and Gerhard Widmer. Playlist generation using start and end songs. In *Proceedings of the International Society for Music Information Retrieval Conference*, volume 8, pages 173–178, 2008.

[GB21a] Giovanni Gabbolini and Derek Bridge. Generating interesting song-to-song segues with dave. In *Proceedings of the ACM Conference on User Modeling, Adaptation and Personalization*, pages 98–107, 2021.

[GB21b] Giovanni Gabbolini and Derek Bridge. An interpretable music similarity measure based on path interestingness. In *Proceedings of the International Society for Music Information Retrieval Conference*, 2021.

[GB21c] Giovanni Gabbolini and Derek Bridge. Play it again, Sam! Recommending familiar music in fresh ways. In *Proceedings of the ACM Conference on Recommender Systems*, pages 697–701, 2021.

[GB22] Giovanni Gabbolini and Derek Bridge. A user-centered investigation of personal music tours. In *Proceedings of the ACM Conference on Recommender Systems*, pages 25–34, 2022.

[GB23] Giovanni Gabbolini and Derek Bridge. Predicting the listening contexts of music playlists using knowledge graphs. In *Advances in Information Retrieval: European Conference on IR Research*. Springer, 2023.

[GC13] Arthur Germain and Jacob Chakareski. Spotify me: Facebook-assisted automatic playlist generation. In *Proceedings of the IEEE International Workshop on Multimedia Signal Processing*, pages 025–028, 2013.

[GCC$^+$19] Alois Gruson, Praveen Chandar, Christophe Charbuillet, James McInerney, Samantha Hansen, Damien Tardieu, and Ben Carterette. Offline evaluation to make decisions about playlistrecommendation algorithms. In *Proceedings of the ACM International Conference on Web Search and Data Mining*, pages 420–428, 2019.

[GCS11] Fabien Gouyon, Nuno Cruz, and Luis Sarmento. A last.fm and youtube mash-up for music browsing and playlist edition. In *Proceedings of the Late-Dreaking Demo Session at the International Music Information Retrieval Conference*, 2011.

[GCW13] Darryl Griffiths, Stuart Cunningham, and Jonathan Weinel. A discussion of musical features for automatic music playlist generation using affective technologies. In *Proceedings of the Audio Mostly Conference*, pages 1–4, 2013.

[GCW16] Darryl Griffiths, Stuart Cunningham, and Jonathan Weinel. An interactive music playlist generator that responds to user emotion and context. *Electronic Visualisation and the Arts*, pages 275–276, 2016.

[GDS16] Roman B Gebhardt, Matthew EP Davies, and Bernhard U Seeber. Psychoacoustic approaches for harmonic music mixing. *Applied Sciences*, 6(5):123, 2016.

[GDW$^+$20] Yulong Gu, Zhuoye Ding, Shuaiqiang Wang, Lixin Zou, Yiding Liu, and Dawei Yin. Deep multifaceted transformers for multi-objective ranking in large-scale e-commerce recommender systems. In *Proceedings of the ACM International Conference on Information & Knowledge Management*, pages 2493–2500, 2020.

[GG05] Masataka Goto and Takayuki Goto. Musicream: New music playback interface for streaming, sticking, sorting, and recalling musical pieces. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 404–411, 2005.

[GGZ$^+$17] Lianli Gao, Zhao Guo, Hanwang Zhang, Xing Xu, and Heng Tao Shen. Video captioning with attention-based LSTM and semantic consistency. *IEEE Transactions on Multimedia*, 19(9):2045–2055, 2017.

[GH06] Liqiang Geng and Howard J. Hamilton. Interestingness measures for data mining: A survey. *ACM Computing Surveys*, 38(3):3, 2006.

[GHE22] Giovanni Gabbolini, Romain Hennequin, and Elena Epure. Data-efficient playlist captioning with musical and linguistic knowledge. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, December 2022.

[GKS07] Don Gartner, Florian Kraft, and Thomas Schaaf. An adaptive distance measure for similarity based playlist generation. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, volume 1, 2007.

[GKW08] Olga Goussevskaia, Michael Kuhn, and Roger Wattenhofer. Exploring music collections on mobile devices. In *Proceedings of the International Conference on Human Computer Interaction with Mobile Devices and Services*, pages 359–362, 2008.

[GLZ$^+$20] Longteng Guo, Jing Liu, Xinxin Zhu, Peng Yao, Shichen Lu, and Hanqing Lu. Normalized and geometry-aware self-attention network for image captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10327–10336, 2020.

[GMS10] Peter Grosche, Meinard Müller, and Craig Stuart Sapp. What Makes Beat Tracking Difficult? A Case Study on Chopin Mazurkas. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 649–654, 2010.

[Gre16] Jacek Grekow. Music emotion maps in arousal-valence space. In *Computer Information Systems and Industrial Management: IFIP TC8 International Conference*, pages 697–706. Springer, 2016.

[GSJ18] Anna Gatzioura, Miquel Sànchez-Marrè, and Alípio Mário Jorge. A Study on Contextual Influences on Automatic Playlist Continuation. In *Proceedings of the International Conference of the Catalan Association for Artificial Intelligence*, volume 308, pages 156–165, 2018.

[GSM17a] Anna Gatzioura and Miquel Sànchez-Marrè. A case-based reasoning framework for music playlist recommendations. In *Pro-*

ceedings of the International Conference on Control, Decision and Information Technologies, pages 0242–0247, 2017.

[GSM17b] Anna Gatzioura and Miquel Sànchez-Marrè. Using contextual information in music playlist recommendations. In *Proceedings of the International Conference of the Catalan Association for Artificial Intelligence*, pages 239–244, 2017.

[GVJSM19] Anna Gatzioura, João Vinagre, Alípio Mário Jorge, and Miquel Sanchez-Marre. A hybrid recommender system for improving automatic playlist continuation. *IEEE Transactions on Knowledge and Data Engineering*, 33(5):1819–1830, 2019.

[Hag15] Anja Nylund Hagen. The playlist experience: Personal playlists in music streaming services. *Popular Music and Society*, 38(5):625–645, 2015.

[Har90] Andrew C Harvey. *Forecasting, structural time Series models and the Kalman filter*. Cambridge University Press, 1990.

[HAS09] Jukka Holm, Antti Aaltonen, and Harri Siirtola. Associating colours with musical genres. *Journal of New Music Research*, 38(1):87–100, 2009.

[HC11] Jia-Lien Hsu and Shuk-Chun Chung. Constraint-based playlist generation by applying genetic algorithm. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pages 1417–1422, 2011.

[HCE$^+$17] Shawn Hershey, Sourish Chaudhuri, Daniel PW Ellis, Jort F Gemmeke, Aren Jansen, R Channing Moore, Manoj Plakal, Devin Platt, Rif A Saurous, and Bryan Seybold. CNN architectures for large-scale audio classification. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pages 131–135, 2017.

[HF01] David B Hauver and James C French. Flycasting: using collaborative filtering to generate a playlist for online radio. In *Proceedings of the International Conference on WEB Delivering of Music*, pages 123–130, 2001.

[HG09] Derek L Hansen and Jennifer Golbeck. Mixing it up: recommending collections of items. In *Proceedings of the CHI Confer-*

ence on Human Factors in Computing Systems, pages 1217–1226, 2009.

[HHKB06] Otmar Hilliges, Phillipp Holzer, Rene Klüber, and Andreas Butz. Audioradar: A metaphorical visualization for the navigation of large music collections. In *Proceedings of the International Symposium on Smart Graphics*, pages 82–92. Springer, 2006.

[HKBS19] Simao Herdade, Armin Kappeler, Kofi Boakye, and Joao Soares. Image captioning: Transforming objects into words. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[HKBT15] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

[HKV08] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the IEEE International Conference on Data Mining*, pages 263–272, 2008.

[HLE10] Jukka Holm, Arto Lehtiniemi, and Antti Eronen. Evaluating an avatar-based user interface for discovering new music. In *Proceedings of the International Conference on Mobile and Ubiquitous Multimedia*, pages 1–10, 2010.

[HLW19] Sophia Hadash, Yu Liang, and Martijn C Willemsen. 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*, pages 1–9, 2019.

[HMB12] Negar Hariri, Bamshad Mobasher, and Robin Burke. Context-aware music recommendation based on latenttopic sequential patterns. In *Proceedings of the ACM Conference on Recommender Systems*, pages 131–138, 2012.

[HO11] Yajie Hu and Mitsunori Ogihara. Nextone player: A music recommendation system based on user behavior. In *Proceedings of the International Society for Music Information Retrieval Conference*, volume 11, 2011.

[HPMA$^+$01] Jiawei Han, Jian Pei, Behzad Mortazavi-Asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Meichun Hsu. Prefixspan:

Mining sequential patterns efficiently by prefix-projected pattern growth. In *Proceedings of the International Conference on Data engineering*, pages 215–224. IEEE, 2001.

[HRH07] Maria Håkansson, Mattias Rost, and Lars Erik Holmquist. Gifts from friends and strangers: A study of mobile music sharing. In *Proceedings of the European Conference on Computer-Supported Cooperative Work*, pages 311–330. Springer, 2007.

[HRS10] Perfecto Herrera, Zuriñe Resa, and Mohamed Sordo. 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*, 2010.

[HSH21] Rujing Huang, Bob LT Sturm, and André Holzapfel. Decentering the West: East Asian Philosophies and the Ethics of Applying Artificial Intelligence to Music. In *Proceedings of the International Society for Music Information Retrieval Conference*, 2021.

[HSL17] Binbin Hu, Chuan Shi, and Jian Liu. Playlist recommendation based on reinforcement learning. In *Proceedings of the International Conference on Intelligence Science*, pages 172–182, 2017.

[HSSL19] MD Zakir Hossain, Ferdous Sohel, Mohd Fairuz Shiratuddin, and Hamid Laga. A comprehensive survey of deep learning for image captioning. *ACM Computing Surveys (CSUR)*, 51(6):1–36, 2019.

[HY13] Pitoyo Hartono and Ryo Yoshitake. Automatic playlist generation from self-organizing music map. *Journal of Signal Processing*, 17(1):11–19, 2013.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[IBZ$^+$19] Rosilde Tatiana Irene, Clara Borrelli, Massimiliano Zanoni, Michele Buccoli, and Augusto Sarti. Automatic playlist generation using convolutional neural networks and recurrent neural networks. In *Proceedings of the European Signal Processing Conference*, pages 1–5. IEEE, 2019.

[IEPR22] Karim M Ibrahim, Elena V Epure, Geoffroy Peeters, and Gaël Richard. Exploiting device and audio data to tag music with user-aware listening contexts. *arXiv preprint arXiv:2211.07250*, 2022.

[IFP22] IFPI. Global music report, 2022. `https://globalMusicreport.ifpi.org/`.

[IHT09] Hiromi Ishizaki, Keiichiro Hoashi, and Yasuhiro Takishima. 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*, pages 135–140, 2009.

[IOK18] Shobu Ikeda, Kenta Oku, and Kyoji Kawagoe. Music playlist recommendation using acoustic-feature transition inside the songs. In *Proceedings of the International Conference on Advances in Mobile Computing & Multimedia*, pages 216–219, 2018.

[Jan15] Mátyás Jani. Fast content independent playlist generation for streaming media. In *Proceedings of the IEEE/ACS International Conference of Computer Systems and Applications*, pages 1–6. IEEE, 2015.

[JDE07] M Cameron Jones, J Stephen Downie, and Andreas F Ehmann. Human similarity judgments: Implications for the design of formal evaluations. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 539–542, 2007.

[JGB$^+$16] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*, 2016.

[JJ09] Carles Fernandes Julià and Sergi Jordà. Songexplorer: A tabletop application for exploring large collections of songs. In *Proceedings of the International Society for Music Information Retrieval Conference*, 2009.

[JKB14] Dietmar Jannach, Iman Kamehkhosh, and Geoffray Bonnin. Analyzing the characteristics of shared playlists for music recommendation. In *Proceedings of the RSWeb Workshop at the ACM Recommender Systems Conference*, 2014.

[JKB16] Dietmar Jannach, Iman Kamehkhosh, and Geoffray Bonnin. Biases in automated music playlist generation: A comparison of next-track recommending techniques. In *Proceedings of the Conference on User Modeling Adaptation and Personalization*, pages 281–285, 2016.

[JKL17] Dietmar Jannach, Iman Kamehkhosh, and Lukas Lerche. Leveraging multi-dimensional user models for personalized next-track music recommendation. In *Proceedings of the Symposium on Applied Computing*, page 1635–1642, 2017.

[JL17] Dietmar Jannach and Malte Ludewig. When recurrent neural networks meet the neighborhood for session-based recommendation. In *Proceedings of the ACM Conference on Recommender Systems*, page 306–310, 2017.

[JLK15] Dietmar Jannach, Lukas Lerche, and Iman Kamehkhosh. Beyond "hitting the hits": Generating coherent music playlist continuations with the right tracks. In *Proceedings of the ACM Conference on Recommender Systems*, page 187–194, 2015.

[JLT14] Mátyás Jani, Gergely Lukács, and György Takács. Experimental investigation of transitions for mixed speech and music playlist generation. In *Proceedings of the Proceedings of International Conference on Multimedia Retrieval*, pages 392–398, 2014.

[JM09] Daniel Jurafsky and James H Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall, 2009.

[JMPS07] Claus Aage Jensen, Ester M Mungure, Torben Bach Pedersen, and Kenneth Sorensen. A data and query model for dynamic playlist generation. In *Proceedings of the IEEE International Conference on Data Engineering Workshop*, pages 65–74, 2007.

[KB18] Mesut Kaya and Derek Bridge. Automatic playlist continuation using subprofile-aware diversification. In *Proceedings of the ACM*

Recommender Systems Challenge, pages 1–6, 2018.

[KBBB18] Domokos M Kelen, Dániel Berecz, Ferenc Béres, and András A Benczúr. Efficient k-nn for playlist continuation. In *Proceedings of the ACM Recommender Systems Challenge*, pages 1–4, 2018.

[KBJ20] Iman Kamehkhosh, Geoffray Bonnin, and Dietmar Jannach. Effects of recommendations on the playlist creation behavior of users. *User Modeling and User-Adapted Interaction*, 30(2):285–322, 2020.

[KBM12] Mohsen Kamalzadeh, Dominikus Baur, and Torsten Möller. A survey on music listening and management behaviours. In *Proceedings of the International Society for Music Information Retrieval Conference*, 2012.

[KES16] Burak Köse, Süleyman Eken, and Ahmet Sayar. Playlist generation via vector representation of songs. In *Proceedings of the INNS Conference on Big Data*, pages 179–185, 2016.

[KGB14] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 655–665, 2014.

[KI15] James King and Vaiva Imbrasaitė. Generating Music playlists with hierarchical clustering and Q-learning. In *Proceedings of the European Conference on Information Retrieval*, pages 315–326. Springer, 2015.

[Kin80] Walter Kintsch. Learning from text, levels of comprehension, or: Why anyone would read a story anyway. *Poetics*, 9(1-3):87–98, 1980.

[KJ17] Iman Kamehkhosh and Dietmar Jannach. User perception of next-track music recommendations. In *Proceedings of the Conference on User Modeling, adaptation and personalization*, pages 113–121, 2017.

[KJB18] Iman Kamehkhosh, Dietmar Jannach, and Geoffray Bonnin. How automated recommendations affect the playlist creation behavior of users. In *Workshops of the ACM Conference on Intelligent User Interfaces*, 2018.

[KJL16] Iman Kamehkhosh, Dietmar Jannach, and Lukas Lerche. 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*, 2016.

[Kju16] Yngvar Kjus. Musical exploration via streaming services: The Norwegian experience. *Popular Communication*, 14(3):127–136, 2016.

[KKMS16] Mohsen Kamalzadeh, Christoph Kralj, Torsten Möller, and Michael Sedlmair. Tagflip: active mobile music discovery with social tags. In *Proceedings of the International Conference on Intelligent User Interfaces*, pages 19–30, 2016.

[KMM$^+$97] Joseph A Konstan, Bradley N Miller, David Maltz, Jonathan L Herlocker, Lee R Gordon, and John Riedl. Grouplens: Applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3):77–87, 1997.

[KMN$^+$20] Yuma Koizumi, Ryo Masumura, Kyosuke Nishida, Masahiro Yasuda, and Shoichiro Saito. A transformer-based audio captioning model with keyword estimation. In *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, pages 1977–1981, 2020.

[KNH$^+$22] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. *ACM Computing Surveys (CSUR)*, 54(10s):1–41, 2022.

[Koe04] Philipp Koehn. Statistical significance tests for machine translation evaluation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 388–395, 2004.

[Koh90] Teuvo Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[KPSW06] Peter Knees, Tim Pohle, Markus Schedl, and Gerhard Widmer. 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*, pages 147–154, 2006.

[KR12] Marius Kaminskas and Francesco Ricci. Contextual music information retrieval and recommendation: State of the art and challenges. *Computer Science Review*, 6(2-3):89–119, 2012.

[KRB22] Yehuda Koren, Steffen Rendle, and Robert Bell. Advances in collaborative filtering. In *Recommender Systems Handbook*, pages 91–142. Springer, 2022.

[KS16] Peter Knees and Markus Schedl. Introduction to music similarity and retrieval. In *Music Similarity and Retrieval: An Introduction to Audio- and Web-based Strategies*, pages 1–30. Springer, 2016.

[KSH17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[KSPW06] Peter Knees, Markus Schedl, Tim Pohle, and Gerhard Widmer. An innovative three-dimensional user interface for exploring music collections enriched. In *Proceedings of the ACM International Conference on Multimedia*, pages 17–24, 2006.

[KW15] Bart P Knijnenburg and Martijn C Willemsen. Evaluating recommender systems with user experiments. In *Recommender Systems Handbook*, pages 309–352. Springer, 2015.

[KWLH18] Jaehun Kim, Minz Won, Cynthia CS Liem, and Alan Hanjalic. Towards seed-free music playlist generation: Enhancing collaborative filtering with playlist title information. In *Proceedings of the ACM Recommender Systems Challenge*, pages 1–6, 2018.

[Lam08] Paul Lamere. Social tagging and music information retrieval. *Journal of New Music Research*, 37(2):101–114, 2008.

[LBE15] Zachary C Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

[LBM11] Jin Ha Lee, Bobby Bare, and Gary Meek. How similar is too similar?: Exploring users' perceptions of similarity in playlist evaluation. In *Proceedings of the International Society for Mu-*

sic Information Retrieval Conference, volume 11, pages 109–114, 2011.

[LCC18] Yi-Hsun Lin, Chia-Hao Chung, and Homer H Chen. Playlist-based tag propagation for improving music auto-tagging. In *Proceedings of the European Signal Processing Conference*, pages 2270–2274. IEEE, 2018.

[LD04] Jin Ha Lee and J Stephen Downie. Survey of music information needs, uses, and seeking behaviours: preliminary findings. In *Proceedings of the International Society for Music Information Retrieval Conference*, volume 2004, page 5th, 2004.

[LDHL12] Eva Lenz, Sarah Diefenbach, Marc Hassenzahl, and Sébastien Lienhard. Mo. shared music, shared moment. In *Proceedings of the Nordic Conference on Human-Computer Interaction: Making Sense Through Design*, pages 736–741, 2012.

[LE07] Paul Lamere and Douglas Eck. Using 3D Visualizations to Explore and Discover Music. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 173–174, 2007.

[Lea96] D.B. Leake. CBR in context: The present and future. In *Case-Based Reasoning: Experiences, Lessons, & Future Directions*, pages 3–30. MIT Press, 1996.

[Lew95] James R. Lewis. IBM Computer usability satisfaction questionnaires: Psychometric evaluation and instructions for use. *International Journal of Human–Computer Interaction*, 7(1):57–78, 1995.

[LH11] Arto Lehtiniemi and Jukka Holm. 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*, pages 94–99, 2011.

[LH18] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *Proceedings of the International Conference on Learning Representations*, 2018.

[LHR09] Hao Liu, Jun Hu, and Matthias Rauterberg. Music playlist recommendation based on user heartbeat and music preference. In

Proceedings of the International Conference on Computer Technology and Development, volume 1, pages 545–549, 2009.

[LHV08] Tuck Leong, Steve Howard, and Frank Vetere. Choice: abdicating or exercising? In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pages 715–724, 2008.

[Lil08] Anita Shen Lillie. *MusicBox: Navigating the space of your Music*. PhD thesis, Massachusetts Institute of Technology, 2008.

[Liu09] Tie-Yan Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

[LJ16] Gergely Lukács and Mátyás Jani. Analyzing speech and music blocks in radio channels: Lessons learned for playlist generation. In *Proceedings of the International Conference on Digital Information Management*, pages 179–184. IEEE, 2016.

[LK75] J. K. Lenstra and A. H. G. Rinnooy Kan. Some simple applications of the travelling salesman problem. *Journal of the Operational Research Society*, 26(4):717–733, 1975.

[LKLJ18] Malte Ludewig, Iman Kamehkhosh, Nick Landia, and Dietmar Jannach. Effective nearest-neighbor music recommendations. In *Proceedings of the ACM Recommender Systems Challenge*, pages 1–6, 2018.

[LL21] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing*, pages 4582–4597, 2021.

[LLG$^+$20] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, 2020.

[LN11] Adam J Lonsdale and Adrian C North. Why do we listen to music? a uses and gratifications analysis. *British Journal of Psychology*, 102(1):108–134, 2011.

[LO13] Arto Lehtiniemi and Jarno Ojala. Evaluating moodpic-a concept for collaborative mood music playlist creation. In *Proceedings of the International Conference on Information Visualisation*, pages 86–95. IEEE, 2013.

[Log02] Beth Logan. Content-based playlist generation: Exploratory experiments. In *Proceedings of the International Society for Music Information Retrieval Conference*, volume 2, pages 295–296, 2002.

[Log04] Beth Logan. Music recommendation from song sets. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 425–428, 2004.

[LOT16] Arto Lehtiniemi, Jarno Ojala, and Henri Toukomaa. Design and evaluation of mood pictures in social music discovery service. *Personal and Ubiquitous Computing*, 20(1):97–119, 2016.

[LOV17] Arto Lehtiniemi, Jarno Ojala, and Kaisa Väänänen. Socially augmented music discovery with collaborative playlists and mood pictures. *Interacting with Computers*, 29(3):416–437, 2017.

[LR08] KuanTing Liu and Roger Andersson Reimer. 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*, pages 403–406, 2008.

[LS01] Beth Logan and Ariel Salomon. A music similarity function based on signal analysis. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, pages 190–190, 2001.

[LSH20] Hsin-Wei Li, Sok-Ian Sou, and Hsun-Ping Hsieh. Room-based playlist arrangement system using group recommendation. In *Proceedings of the International Computer Symposium*, pages 50–54, 2020.

[LSTS15] Elad Liebman, Maytal Saar-Tsechansky, and Peter Stone. DJ-MC: A Reinforcement-Learning Agent for Music Playlist Recommendation. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, pages 591–599, 2015.

[LSTS19] Elad Liebman, Maytal Saar-Tsechansky, and Peter Stone. The right music at the right time: Adaptive personalized playlists based on sequence modeling. *MIS Quarterly*, 43(3), 2019.

[LT07] Stefan Leitich and Martin Topf. Globe of music-music library visualization using geosom. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 167–170, 2007.

[LTJ$^+$18] Qing Li, Qingyi Tao, Shafiq Joty, Jianfei Cai, and Jiebo Luo. Vqa-e: Explaining, elaborating, and enhancing your answers for visual questions. In *Proceedings of the European Conference on Computer Vision*, 2018.

[LVH05] Tuck W Leong, Frank Vetere, and Steve Howard. The serendipity shuffle. In *Proceedings of the Australian Conference on Computer-Human Interaction*, pages 1–4, 2005.

[LVH06] Tuck Wah Leong, Frank Vetere, and Steve Howard. Randomness as a resource for design. In *Proceedings of the Conference on Designing Interactive Systems*, pages 132–139, 2006.

[LWM$^+$09] Edith Law, Kris West, Michael I Mandel, Mert Bay, and J Stephen Downie. Evaluation of algorithms using games: The case of music tagging. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 387–392, 2009.

[MA98] Joseph F McCarthy and Theodore D Anagnost. MusicFX: an arbiter of group preferences for computer supported collaborative workouts. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, pages 363–372, 1998.

[MBQF21] Ilaria Manco, Emmanouil Benetos, Elio Quinton, and György Fazekas. Muscaps: Generating captions for music audio. In *Proceedings of the International Joint Conference on Neural Networks*, pages 1–8. IEEE, 2021.

[MCJT12] Joshua L Moore, Shuo Chen, Thorsten Joachims, and Douglas Turnbull. Learning to embed songs and tags for playlist prediction. In *Proceedings of the International Society for Music Information Retrieval Conference*, volume 12, pages 349–354, 2012.

[MD22] Judith Masthoff and Amra Delić. Group recommender systems: Beyond preference aggregation. In *Recommender Systems Handbook*, pages 381–420. Springer, 2022.

[MEDL09] François Maillet, Douglas Eck, Guillaume Desjardins, and Paul Lamere. Steerable playlist generation by learning song similarity from radio station playlists. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 345–350, 2009.

[MGL$^+$22] Cataldo Musto, Marco de Gemmis, Pasquale Lops, Fedelucio Narducci, and Giovanni Semeraro. Semantics and content-based recommendations. In *Recommender Systems Handbook*, pages 251–298. Springer, 2022.

[MGMS14] Daniel Müllensiefen, Bruno Gingras, Jason Musil, and Lauren Stewart. The musicality of non-musicians: An index for assessing musical sophistication in the general population. *PloS one*, 9(2), 2014.

[MHCV19] Martijn Millecamp, Nyi Nyi Htun, Cristina Conati, and Katrien Verbert. 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*, pages 397–407, 2019.

[Mil98] George A Miller. *WordNet: An electronic lexical Database*. MIT Press, 1998.

[MK18] YV Srinivasa Murthy and Shashidhar G Koolagudi. Content-based music information retrieval (cb-mir) and its applications toward the music industry: A review. *ACM Computing Surveys (CSUR)*, 51(3):1–46, 2018.

[ML11] Brian McFee and Gert RG Lanckriet. The natural language of playlists. In *Proceedings of the International Society for Music Information Retrieval Conference*, volume 11, pages 537–541, 2011.

[ML12] Brian McFee and Gert RG Lanckriet. Hypergraph models of playlist dialects. In *Proceedings of the International Society for*

Music Information Retrieval Conference, volume 12, pages 343–348, 2012.

[MPR$^+$18] Diego Monti, Enrico Palumbo, Giuseppe Rizzo, Pasquale Lisena, Raphaël Troncy, Michael Fell, Elena Cabrio, and Maurizio Morisio. An ensemble approach of recurrent neural networks using pre-trained embeddings for playlist completion. In *Proceedings of the ACM Recommender Systems Challenge*, pages 1–6, 2018.

[MRNT10] Scott Miller, Paul Reimer, Steven R Ness, and George Tzanetakis. Geoshuffle: Location-aware, content-based music browsing using self-organizing tag clouds. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 237–242, 2010.

[MSC$^+$13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, volume 26, 2013.

[MSJ$^+$15] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

[MUNS05] Fabian Mörchen, Alfred Ultsch, Mario Nöcker, and Christian Stamm. Databionic visualization of music collections according to perceptual distance. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 396–403, 2005.

[MvNL10] Bart Moens, Leon van Noorden, and Marc Leman. D-jogger: Syncing music with walking. In *Proceedings of the Sound and Music Computing Conference*, pages 451–456, 2010.

[NDR05] Robert Neumayer, Michael Dittenbach, and Andreas Rauber. Playsom and pocketsomplayer, alternative interfaces to large music collections. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 618–623, 2005.

[NNDK22] Athanasios N Nikolakopoulos, Xia Ning, Christian Desrosiers, and George Karypis. Trust your neighbors: a comprehensive survey of neighborhood-based methods for recommender systems. *Recommender Systems Handbook*, pages 39–89, 2022.

[OFM06] Nuria Oliver and Fernando Flores-Mangas. MPTrain: a mobile, music and physiology-based personal trainer. In *Proceedings of the Conference on Human-Computer Interaction with Mobile Devices and Services*, pages 21–28, 2006.

[OKS06] Nuria Oliver and Lucas Kreger-Stickles. Papa: Physiology and purpose-aware automatic playlist generation. In *Proceedings of the International Society for Music Information Retrieval Conference*, volume 2006, page 7th, 2006.

[OKT10] Ashley M Oudenne, Youngmoo E Kim, and Douglas S Turnbull. Meerkat: exploring semantic music discovery using personalized radio. In *Proceedings of the International Conference on Multimedia Information Retrieval*, pages 429–432, 2010.

[OLJ$^+$04] Kenton O'Hara, Matthew Lipson, Marcel Jansen, Axel Unger, Huw Jeffries, and Peter Macer. Jukola: democratic music choice in a public space. In *Proceedings of the Conference on Designing interactive Systems: processes, practices, Methods, and techniques*, pages 145–154, 2004.

[OMS16] Melissa Onori, Alessandro Micarelli, and Giuseppe Sansonetti. A comparative analysis of personality-based music recommender systems. In *Proceedings of the Empire Workshop at the ACM Conference on Recommender Systems*, pages 55–59, 2016.

[OON$^+$16] Sergio Oramas, Vito Claudio Ostuni, Tommaso Di Noia, Xavier Serra, and Eugenio Di Sciascio. Sound and music recommendation with knowledge graphs. *ACM Transactions on Intelligent Systems and Technology*, 8(2):1–21, 2016.

[PBdGS19] Marco Polignano, Pierpaolo Basile, Marco de Gemmis, and Giovanni Semeraro. 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*, pages 235–239, 2019.

[PBS$^+$01] John Platt, Christopher J Burges, Steven Swenson, Christopher Weare, and Alice Zheng. Learning a gaussian process prior for automatically generating music playlists. In *Advances in Neural Information Processing Systems*, volume 14, 2001.

[PDM$^+$22] Savvas Petridis, Nediyana Daskalova, Sarah Mennicken, Samuel F Way, Paul Lamere, and Jennifer Thom. Tastepaths: Enabling deeper exploration and understanding of personal preferences in recommender systems. In *Proceedings of the International Conference on Intelligent User Interfaces*, page 120–133, 2022.

[PE02] Steffen Pauws and Berry Eggen. Pats: Realization and user evaluation of an automatic playlist generator. In *Proceedings of the International Society for Music Information Retrieval Conference*, volume 2, pages 222–230, 2002.

[PFC12] Geoffroy Peeters and Joachim Flocon-Cholet. Perceptual tempo estimation using GMM-regression. In *Proceedings of the Second International ACM Workshop on Music Information Retrieval With User-Centered and Multimodal Strategies*, pages 45–50, 2012.

[PGG19] Lorenzo Porcaro and Emilia Gómez Gutiérrez. 20 years of playlists: A statistical analysis on popularity and diversity. In *Proceedings of the International Society for Music Information Retrieval Conference*, 2019.

[PGS$^+$11] Geoffroy Peeters, Bruno L Giordano, Patrick Susini, Nicolas Misdariis, and Stephen McAdams. The timbre toolbox: Extracting audio descriptors from musical signals. *The Journal of the Acoustical Society of America*, 130(5):2902–2916, 2011.

[PHG$^+$17] Luciana Fujii Pontello, Pedro HF Holanda, Bruno Guilherme, João Paulo V Cardoso, Olga Goussevskaia, and Ana Paula Couto Da Silva. Mixtape: using real-time user feedback to navigate large media collections. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 13(4):1–22, 2017.

[PK21] So Yeon Park and Blair Kaneshiro. Social music curation that works: Insights from successful collaborative playlists. *Proceedings of the ACM on Human-Computer Interaction*, 5:1–27, 2021.

[PK22] So Yeon Park and Blair Kaneshiro. User perspectives on critical factors for collaborative playlists. *PloS one*, 17(1):e0260750, 2022.

[PKS$^+$07] Tim Pohle, Peter Knees, Markus Schedl, Elias Pampalk, and Gerhard Widmer. "reinventing the wheel": a novel approach to music player interfaces. *IEEE Transactions on Multimedia*, 9(3):567–575, 2007.

[PKW$^+$20] Ayush Patwari, Nicholas Kong, Jun Wang, Ullas Gargi, Michele Covell, and Aren Jansen. Semantically meaningful attributes from co-listen embeddings for playlist exploration and expansion. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 527–533, 2020.

[PL21] So Yeon Park and Sang Won Lee. Lost in co-curation: uncomfortable interactions and the role of communication in collaborative music playlists. *Proceedings of the ACM Conference on Human-Computer Interaction*, 5:1–24, 2021.

[PLLK19] So Yeon Park, Audrey Laplante, Jin Ha Lee, and Blair Kaneshiro. Tunes together: Perception and experience of collaborative playlists. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 723–730, 2019.

[PNP$^+$18] Jordi Pons, Oriol Nieto, Matthew Prockup, Erik Schmidt, Andreas Ehmann, and Xavier Serra. End-to-end learning for music audio tagging at scale. In *Proceedings of the International Society for Music Information Retrieval Conference*, 2018.

[Por80] M. Porter. An algorithm for suffix stripping. *Program*, 14:130–137, 1980.

[PP12] George Popescu and Pearl Pu. 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*, pages 1673–1678, 2012.

[PPW05] Tim Pohle, Elias Pampalk, and Gerhard Widmer. Generating similarity-based playlists using traveling salesman algorithms. In *Proceedings of the International Conference on Digital Audio Effects*, pages 220–225, 2005.

[PRBK22] So Yeon Park, Emily Redmond, Jonathan Berger, and Blair Kaneshiro. 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*, pages 1–16, 2022.

[PRC00] François Pachet, Pierre Roy, and Daniel Cazaly. A combinatorial approach to content-based music selection. *IEEE Multimedia*, 7(1):44–51, 2000.

[PRM02] Elias Pampalk, Andreas Rauber, and Dieter Merkl. Content-based organization and visualization of music archives. In *Proceedings of the ACM International Conference on Multimedia*, pages 570–579, 2002.

[PRWZ02] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[PS19] Jordi Pons and Xavier Serra. Musicnn: Pre-trained convolutional neural networks for music audio tagging. In *Proceedings of the International Society for Music Information Retrieval Conference*, 2019.

[PvdW05] Steffen Pauws and Sander van de Wijdeven. User evaluation of a new interactive playlist generation concept. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 638–643, 2005.

[PVV06] Steffen Pauws, Wim Verhaegh, and Mark Vossen. Fast generation of optimal music playlists using local search. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 138–143, 2006.

[PVV08] Steffen Pauws, Wim Verhaegh, and Mark Vossen. Music playlist generation by adapted simulated annealing. *Information Sciences*, 178(3):647–662, 2008.

[PZS16] Martin Pichl, Eva Zangerle, and Günther Specht. Understanding playlist creation on music streaming platforms. In *Proceedings* *of the IEEE International Symposium on Multimedia*, pages 475–480. IEEE, 2016.

[PZS17] Martin Pichl, Eva Zangerle, and Günther Specht. Understanding user-curated playlists on spotify: A machine learning approach. *International Journal of Multimedia Data Engineering and Management*, 8(4):44–59, 2017.

[RBBC07] Gordon Reynolds, Dan Barry, Ted Burke, and Eugene Coyle. Towards a personal automatic music playlist generation algorithm: the need for contextual information. In *Proceedings of the International Audio Mostly Conference*, pages 84–89, 2007.

[RBH05] Robert Ragno, Christopher JC Burges, and Cormac Herley. Inferring similarity between music objects with application to playlist generation. In *Proceedings of the ACM SIGMM International Workshop on Multimedia Information Retrieval*, pages 73–80, 2005.

[RKS11] Neil Rubens, Dain Kaplan, and Masashi Sugiyama. Active learning in recommender systems. In *Recommender Systems Handbook*, pages 735–767. Springer, 2011.

[RKV$^+$18] Vasiliy Rubtsov, Mikhail Kamenshchikov, Ilya Valyaev, Vasiliy Leksin, and Dmitry I Ignatov. A hybrid two-stage recommender system for automatic playlist continuation. In *Proceedings of the ACM Recommender Systems Challenge*, pages 1–4, 2018.

[RLHTM18] Jimena Royo-Letelier, Romain Hennequin, Viet-Anh Tran, and Manuel Moussallam. Disambiguating music artists at scale with audio metric learning. In *Proceedings of the International Society for Music Information Retrieval Conference*, 2018.

[RM87] Richard A Roberts and Clifford T Mullis. *Digital Signal Processing*. Addison-Wesley, 1987.

[RM06] Sasank Reddy and Jeff Mascia. Lifetrak: Music in tune with your life. In *Proceedings of the ACM International Workshop on Human-centered Multimedia*, pages 25–34, 2006.

[RN10] Stuart Russell and Peter Norvig. Beyond classical search. In *Artificial Intelligence: A Modern Approach*, pages 120–160. Prentice Hall, Upper Saddle River, NJ, 2010.

[RRS22] Francesco Ricci, Lior Rokach, and Bracha Shapira. Recommender systems: Techniques, applications, and challenges. In *Recommender Systems Handbook*, pages 1–35. Springer, 2022.

[RSR+20] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67, 2020.

[SACH06] Vegard Sandvold, Thomas Aussenac, Oscar Celma, and Perfecto Herrera. Good vibrations: Music discovery through personal musical concepts. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 322–323, 2006.

[SB13] Ingo Schmädecke and Holger Blume. High performance hardware architectures for automated music classification. In *Algorithms from and for Nature and Life*, pages 539–547. Springer, 2013.

[SB18] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT Press, 2018.

[SBE+21] Harald Steck, Linas Baltrunas, Ehtsham Elahi, Dawen Liang, Yves Raimond, and Justin Basilico. Deep Learning for Recommender Systems: A Netflix Case Study. *AI Magazine*, 42:7–18, 2021.

[SBI14] Markus Schedl, Georg Breitschopf, and Bogdan Ionescu. Mobile Music Fenius: Reggae at the Beach, Metal on a Friday Night? In *Proceedings of International Conference on Multimedia Retrieval*, pages 507–510, 2014.

[SC12] Andy M Sarroff and Michael Casey. Modeling and predicting song adjacencies in commercial albums. In *Proceedings of the Sound and Music Computing Conference*, 2012.

[SC18] Shun-Yao Shih and Heng-Yu Chi. Automatic, personalized, and flexible playlist generation using reinforcement learning. In *Proceedings of the International Society for Music Information Retrieval Conference*, 2018.

[SCB$^+$22] Matteo Stefanini, Marcella Cornia, Lorenzo Baraldi, Silvia Cascianelli, Giuseppe Fiameni, and Rita Cucchiara. From show to tell: a survey on deep learning-based image captioning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):539–559, 2022.

[SCBG08] Mohamed Sordo, Oscar Celma, Martin Blech, and Enric Guaus. The quest for musical genres: Do the experts and the wisdom of crowds agree? In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 255–260, 2008.

[Sch79] Roger C Schank. Interestingness: Controlling inferences. *Artificial Intelligence*, 12:273–297, 1979.

[Sch98] Eric D Scheirer. Tempo and beat analysis of acoustic musical signals. *The Journal of the Acoustical Society of America*, 103(1):588–601, 1998.

[SG18] Anastasia Shimorina and Claire Gardent. Handling rare items in data-to-text generation. In *Proceedings of the International Conference on Natural Language Generation*, pages 360–370, 2018.

[SHB16] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 1715–1725, 2016.

[Shn92] Ben Shneiderman. Tree visualization with tree-maps: 2-d space-filling approach. *ACM Transactions on Graphics*, 11(1):92–99, 1992.

[SK05] Richard Stenzel and Thomas Kamps. Improving content-based similarity measures by training a collaborative model. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 264–271, 2005.

[SKMB22] Markus Schedl, Peter Knees, Brian McFee, and Dmitry Bogdanov. Music recommendation systems: Techniques, use cases, and challenges. In *Recommender Systems Handbook*, pages 927–971. Springer, 2022.

[SLPL18] Louis Spinelli, Josephine Lau, Liz Pritchard, and Jin Ha Lee. 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*, pages 671–677, 2018.

[SLT07] Dionisios N Sotiropoulos, Aristomenis S Lampropoulos, and George A Tsihrintzis. Evaluation of modeling music similarity perception via feature subset selection. In *International Conference on User Modeling*, pages 288–297. Springer, 2007.

[SM11] Simone Stumpf and Sam Muscroft. When users generate music playlists: When words leave off, music begins? In *Proceedings of the IEEE International Conference on Multimedia and Expo*, pages 1–6. IEEE, 2011.

[SPCS21] Harald Victor Schweiger, Emilia Parada-Cabaleiro, and Markus Schedl. Does track sequence in user-generated playlists matter?. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 618–625, 2021.

[SPKW07] Dominik Schnitzer, Tim Pohle, Peter Knees, and Gerhard Widmer. One-touch access to music on mobile devices. In *Proceedings of the International Conference on Mobile and ubiquitous Multimedia*, pages 103–109, 2007.

[Spo22] Spotify. Explore spotify catalogue, 2022.

[STOH20] Keigo Sakurai, Ren Togo, Takahiro Ogawa, and Miki Haseyama. Music playlist generation based on reinforcement learning using acoustic feature map. In *Proceedings of the Global Conference on Consumer Electronics*, pages 942–943. IEEE, 2020.

[STOH21] Keigo Sakurai, Ren Togo, Takahiro Ogawa, and Miki Haseyama. Music playlist generation based on graph exploration using reinforcement learning. In *Proceedings of the Global Conference on Life Sciences and Technologies*, pages 53–54. IEEE, 2021.

[STOH22] Keigo Sakurai, Ren Togo, Takahiro Ogawa, and Miki Haseyama. Controllable music playlist generation based on knowledge graph and reinforcement learning. *Sensors*, 22(10):3722, 2022.

[SVC$^+$18] Ganeshsiva Subramaniam, Janhavi Verma, Nikhil Chandrasekhar, KC Narendra, and Koshy George. Generating playlists on the basis of emotion. In *IEEE Symposium Series on Computational Intelligence*, pages 366–373. IEEE, 2018.

[SW06] Malcolm Slaney and William White. Measuring playlist diversity for recommendation systems. In *Proceedings of the ACM Workshop on Audio and Music Computing Multimedia*, pages 77–82, 2006.

[SW07] Malcolm Slaney and William White. Similarity based on rating data. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 479–484, 2007.

[SWT08] David Sprague, Fuqu Wu, and Melanie Tory. Music selection using the partyvote democratic jukebox. In *Proceedings of the Working Conference on Advanced visual Interfaces*, pages 433–436, 2008.

[SZC$^+$18] Markus Schedl, Hamed Zamani, Ching-Wei Chen, Yashar Deldjoo, and Mehdi Elahi. Current challenges and visions in music recommender systems research. *International Journal of Multimedia Information Retrieval*, 7(2):95–116, 2018.

[TCC$^+$15] Roberto Turrin, Andrea Condorelli, Paolo Cremonesi, Roberto Pagano, and Massimo Quadrana. Large scale music recommendation. In *Proceedings of the Workshop on Large-Scale Recommender Systems at the ACM Recommender Systems Conference*, 2015.

[THA04] Marc Torrens, Patrick Hertzog, and Josep Lluis Arcos. Visualizing and exploring personal music libraries. In *Proceedings of the International Society for Music Information Retrieval Conference*, 2004.

[TLL17] Nava Tintarev, Christoph Lofi, and CS Liem. Sequences of diverse song recommendations. *User Modelling, Adaptation and Personalization*, 2017.

[TSL19] Thanh Tran, Renee Sweeney, and Kyumin Lee. 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*, pages 245–254, 2019.

[UKM18] Seiji Ueda, Atsushi Keyaki, and Jun Miyazaki. A contextual random walk model for automated playlist generation. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, pages 367–374. IEEE, 2018.

[VA15] Felipe Vieira and Nazareno Andrade. Evaluating conflict management mechanisms for online social jukeboxes. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 190–196, 2015.

[VDSW18] Andreu Vall, Matthias Dorfer, Markus Schedl, and Gerhard Widmer. A hybrid approach to music playlist continuation based on playlist-song membership. In *Proceedings of the Annual ACM Symposium on Applied Computing*, pages 1374–1382, 2018.

[VEZD$^+$17] Andreu Vall, Hamid Eghbal-Zadeh, Matthias Dorfer, Markus Schedl, and Gerhard Widmer. 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*, pages 46–54, 2017.

[VEzDS16] Andreu Vall, Hamid Eghbal-zadeh, Matthias Dorfer, and Markus Schedl. Timbral and semantic features for music playlists. In *Proceedings of the Machine Learning for Music Discovery Workshop at the Internation Conference on Machine Learning*, 2016.

[VGMS18] Iacopo Vagliano, Lukas Galke, Florian Mai, and Ansgar Scherp. Using adversarial autoencoders for multi-modal automatic playlist continuation. In *Proceedings of the ACM Recommender Systems Challenge*, pages 1–6, 2018.

[VGV05] Rob Van Gulik and Fabio Vignoli. Visual playlist generation on the artist map. In *Proceedings of the International Society for Music Information Retrieval Conference*, volume 5, pages 520–523, 2005.

[vGVvdW04] Rob van Gulik, Fabio Vignoli, and Huub van de Wetering. Mapping music in the palm of your hand, explore and discover your collection. In *Proceedings of the International Society for Music Information Retrieval Conference*, 2004.

[VLZP15] Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4566–4575, 2015.

[VM21] A Helen Victoria and G Maragatham. Automatic tuning of hyperparameters using bayesian optimization. *Evolving Systems*, 12(1):217–223, 2021.

[vNdV18] Timo van Niedek and Arjen P de Vries. Random walk with restart for automatic playlist continuation and query-specific adaptations. In *Proceedings of the ACM Recommender Systems Challenge*, pages 1–6, 2018.

[Voo06] Michael Voong. *Generating Music playlists using colour (Bachelor degree thesis)*. University of Birmingham, 2006.

[VP05] Fabio Vignoli and Steffen Pauws. A music retrieval system based on user driven similarity and its evaluation. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 272–279, 2005.

[VQS$^+$17] Andreu Vall, Massimo Quadrana, Markus Schedl, Gerhard Widmer, and Paolo Cremonesi. The importance of song context in music playlists. In *Proceedings of the ACM Conference on Recommender Systems*, 2017.

[VQSW18] Andreu Vall, Massimo Quadrana, Markus Schedl, and Gerhard Widmer. The importance of song context and song order in automated music playlist generation. *arXiv preprint arXiv:1807.04690*, 2018.

[VQSW19] Andreu Vall, Massimo Quadrana, Markus Schedl, and Gerhard Widmer. Order, context and popularity bias in next-song recommendations. *International Journal of Multimedia Information Retrieval*, 8(2):101–113, 2019.

[VRC$^+$18] Maksims Volkovs, Himanshu Rai, Zhaoyue Cheng, Ga Wu, Yichao Lu, and Scott Sanner. Two-stage model for automatic playlist continuation at scale. In *Proceedings of the ACM Recommender Systems Challenge*, pages 1–6, 2018.

[VSC16] Flavian Vasile, Elena Smirnova, and Alexis Conneau. Meta-prod2vec: Product embeddings using side-information for recommendation. In *Proceedings of the ACM Conference on Recommender Systems*, pages 225–232, 2016.

[VSP$^+$17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[Wat89] Christopher John Cornish Hellaby Watkins. *Learning from delayed rewards*. PhD thesis, University of Cambridge, 1989.

[WBDB17] Xander Wilcke, Peter Bloem, and Victor De Boer. The knowledge graph as the default data model for learning on heterogeneous knowledge. *Data Science*, 1(1-2):39–57, 2017.

[WCNS20] Minz Won, Sanghyuk Chun, Oriol Nieto, and Xavier Serra. Data-driven harmonic filters for audio representation learning. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, pages 536–540, 2020.

[Web20] Jack Webster. Taste in the platform age: Music streaming services and new forms of class distinction. *Information, Communication & Society*, 23(13):1909–1924, 2020.

[WFBS20] Minz Won, Andres Ferraro, Dmitry Bogdanov, and Xavier Serra. Evaluation of CNN-based automatic music tagging models. *arXiv preprint arXiv:2006.00751*, 2020.

[WGI14] Morgan K Ward, Joseph K Goodman, and Julie R Irwin. The same old song: The power of familiarity in music choice. *Marketing Letters*, 25(1):1–11, 2014.

[WLS19] Chenguang Wang, Mu Li, and Alexander J Smola. Language models with transformers. *arXiv preprint arXiv:1904.09408*, 2019.

[WMH$^+$21] Justin D Weisz, Michael Muller, Stephanie Houde, John Richards, Steven I Ross, Fernando Martinez, Mayank Agarwal, and Kartik Talamadupula. Perfection not required? human-ai partnerships in code translation. In *Proceedings of the Interna-*

tional Conference on Intelligent User Interfaces, pages 402–412, 2021.

[XLSZ09] Linxing Xiao, Lie Lu, Frank Seide, and Jie Zhou. 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*, pages 1885–1888. IEEE, 2009.

[YBK21] Ali Yürekli, Alper Bilge, and Cihan Kaleli. Exploring playlist titles for cold-start music recommendation: an effectiveness analysis. *Journal of Ambient Intelligence and Humanized Computing*, 12(11):10125–10144, 2021.

[YJCL18] Hojin Yang, Yoonki Jeong, Minjin Choi, and Jongwuk Lee. MMCF: Multimodal collaborative filtering for automatic playlist continuation. In *Proceedings of the ACM Recommender Systems Challenge*, pages 1–6, 2018.

[YKB21] Ali Yürekli, Cihan Kaleli, and Alper Bilge. Alleviating the cold-start playlist continuation in music recommendation using latent semantic indexing. *International Journal of Multimedia Information Retrieval*, 10(3):185–198, 2021.

[ZGMRMF10] Elena Zheleva, John Guiver, Eduarda Mendes Rodrigues, and Nataša Milić-Frayling. Statistical models of music-listening sessions in social media. In *Proceedings of the International Conference on World wide web*, pages 1019–1028, 2010.

[ZHJ$^+$18] Lin Zhu, Bowen He, Mengxin Ji, Cheng Ju, and Yihong Chen. Automatic music playlist continuation via neighbor-based collaborative filtering and discriminative reweighting/reranking. In *Proceedings of the ACM Recommender Systems Challenge*, pages 1–6, 2018.

[ZK06] Yongwei Zhu and Mohan S Kankanhalli. Precise pitch profile feature extraction from musical audio for key detection. *IEEE Transactions on Multimedia*, 8(3):575–584, 2006.

[ZKW$^+$20] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. BERTScore: Evaluating Text Generation with

BERT. In *Proceedings of the International Conference on Learning Representations*, 2020.

[ZML16] Fan Zhang, Hongying Meng, and Maozhen Li. Emotion extraction and recognition from music. In *Proceedings of the International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery*, pages 1728–1733, 2016.

[ZSCH18] Xing Zhao, Qingquan Song, James Caverlee, and Xia Hu. Trailmix: An ensemble recommender system for playlist curation and continuation. In *Proceedings of the ACM Recommender Systems Challenge*, pages 1–6, 2018.

[ZSLC19] Hamed Zamani, Markus Schedl, Paul Lamere, and Ching-Wei Chen. An analysis of approaches taken in the acm recsys challenge 2018 for automatic music playlist continuation. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(5):1–21, 2019.

[ZSLS19] Sanqiang Zhao, Piyush Sharma, Tomer Levinboim, and Radu Soricut. Informative image captioning with external sources of information. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 6485–6494, 2019.

[ZWF$^+$21] Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models. In *Proceedings of the International Conference on Machine Learning*, pages 12697–12706, 2021.

[ZZS01] Fang Zheng, Guoliang Zhang, and Zhanjiang Song. Comparison of different implementations of MFCC. *Journal of Computer Science and Technology*, 16(6):582–589, 2001.