JOHANNES KEPLER
UNIVERSITY LINZ

Submitted by
Andreu Vall Portabella

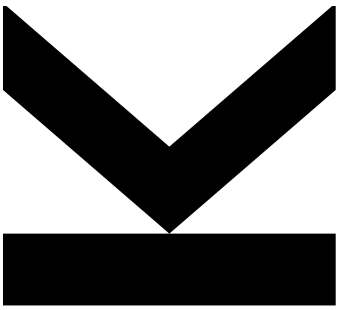
Submitted at
Institute of Computational Perception

Supervisor and
First Examiner
Markus Schedl

Second Examiner
Dietmar Jannach

December 2018

Machine Learning Approaches to Hybrid Music Recommender Systems



Doctoral Thesis

to obtain the academic degree of

Doktor der technischen Wissenschaften

in the Doctoral Program

Technische Wissenschaften

JOHANNES KEPLER
UNIVERSITY LINZ
Altenbergerstraße 69
4040 Linz, Österreich
www.jku.at
DVR 0093696

STATUTORY DECLARATION

I hereby declare that the thesis submitted is my own unaided work, that I have not used other than the sources indicated, and that all direct and indirect sources are acknowledged as references.

This printed thesis is identical with the electronic version submitted.

Linz, December 2018

Andreu Vall Portabella

ABSTRACT

Music catalogs in music streaming services, on-line music shops and private collections become increasingly larger and consequently difficult to navigate. Music recommender systems are technologies devised to support users accessing such large catalogs by automatically identifying and suggesting music that may interest them. This thesis focuses on the machine learning aspects of music recommendation with contributions at the intersection of recommender systems and music information retrieval: I investigate and propose recommender systems that observe and exploit the particularities of the music domain.

The thesis specializes in *hybrid* music recommender systems, so called because they combine two fundamentally different types of data: (1) user–music interaction histories (e.g., the music that users recently listened to, or "liked"), with (2) descriptions of the musical content (e.g., the genre, or acoustical properties of a song). The proposed hybrid music recommender systems integrate the strengths of these two types of data into enhanced standalone systems. This is in contrast to most previous approaches in the literature, where hybridization was achieved through the heuristic combination of music recommendations issued by independent systems. The proposed hybrid music recommender systems are thoroughly evaluated against competitive recommender system baselines, for different music recommendation tasks, and on different datasets. According to the conducted experiments, the proposed systems predict music recommendations comparably or more accurately than the considered baselines, with the improvements being largely explained by their superior ability to handle infrequent music items. In this way, the proposed hybrid music recommender systems provide means to alleviate the so-called "cold-start" problem for new releases and infrequent music and enable the discovery of music beyond the *charts* of popular music.

Special attention is paid to the particularities of the music domain. I focus on two important music recommendation tasks: music artist recommendation, focusing on general, stable user music preferences, and music playlist continuation, focusing on local relationships in short listening sessions. I exploit data sources abundant in the context of on-line music consumption: user listening histories, hand-curated music playlists, music audio signal, and social tags. I investigate challenges specific to modeling music playlists: the choice and the arrangement of songs within playlists, and the effectiveness of different types of music descriptions to identify songs that fit well together.

ZUSAMMENFASSUNG

Musikkataloge von Streamingplattformen und Online-Musikshops, sowie private Musiksammlungen werden zunehmend größer und lassen sich daher immer schwieriger effizient durchsuchen. Musikempfehlungssysteme unterstützen Benutzer darin, solch riesige Kataloge zu durchstöbern, indem sie automatisch Musik, die den Benutzern gefallen könnte identifizieren und vorschlagen. Diese Disseration befasst sich mit Aspekten des maschinellen Lernens im Kontext von Empfehlungssystemen. Ihr Beitrag liegt an der Schnittstelle der Forschungsfelder Recommender Systems und Music Information Retrieval, da die erforschten Empfehlungssysteme Charakteristika der Musikdomäne berücksichtigen.

Diese Arbeit erforscht *hybride* Empfehlungssysteme, welche zwei unterschiedliche Datentypen kombiniert: (1) Benutzer–Musik Interaktionen (z.B. zuletzt gehörte oder als Favorit gespeicherte Musik) und (2) deskriptive Musikbeschreibungen (z.B. Genre oder akustische Eigenschaften). Die erarbeiteten hybriden Musikempfehlungssysteme vereinen die Vorzüge beider Arten von Daten in einem Einzelsystem. Dies kontrastiert die früheren Ansätze, die Hybridisierung durch heuristische Kombination der Musikempfehlungen von mehreren Einzelsystemen erreicht haben. Die in dieser Arbeit erforschten hybriden Musikempfehlungssysteme werden umfassend gegen Baselines evaluiert, wobei unterschiedliche Empfehlungsszenarien und Datensets analysiert werden. Im Vergleich zu den Baselines, weisen die Empfehlungen der erforschten Systeme eine gleich gute oder bessere Akkuratheit auf. Die Verbesserungen werden vor allem durch eine spezielle Berücksichtigung seltener Musik erreicht. Dadurch sind unsere Ansätze insbesondere dazu geeignet, das “cold-start” Problem für neue Musikveröffentlichungen und selten gehörte Musik zu reduzieren, sowie in Folge die Musikexploration über die Pop-Musikcharts hinaus zu gewährleisten.

Diese Arbeit widmet sich den Besonderheiten der Musikdomäne. Zwei wichtige Musikempfehlungsaufgaben werden in den Fokus gerückt: Empfehlung von Musikkünstlern (was generelle, stabile Benutzermodelle über Musikpräferenzen voraussetzt) und Empfehlungen zur Erweiterung von Musikwiedergabelisten (was die Modellierung der Abfolge von Stücken in kurzen Hörsitzungen erfordert). In den erforschten Ansätzen werden vielfältige Daten des Online-Musikkonsums herangezogen und miteinander verknüpft: Hörsitzungen, kuratierte Wiedergabelisten, Audiosignale und kollaborativ generierte Beschreibungen. Des Weiteren berücksichtige ich spezifische Herausforderungen der automatischen Erweiterung von Wiedergabe-

listen: die Auswahl und Platzierung von Liedern in Wiedergabelisten sowie die Eignung unterschiedlicher Musikdeskriptoren zur Identifikation von Musikstücken die gut zusammen passen.

ACKNOWLEDGMENTS

I thank my family for all their support, and especially Rocío. During these years, patiently, you have accompanied me, read and proofread everything that I have written, given me your opinion, and designed each of the posters that I have presented. Gràcies.

I thank my friends and colleagues, at the institute and outside, with whom I have learned, discussed, played, discovered, and shared so much during this process.

I thank Markus Schedl (my supervisor) for having given me the opportunity to start my doctorate, and Gerhard Widmer (the head of the Institute of Computational Perception) for having helped me finish it. And I thank both for all the valuable feedback.

This research has received funding from the Austrian Science Fund (FWF) under the project P25655 (*Social Media Mining for Multimodal Music Retrieval*) and from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No 670035 (*Con Espressione*).

CONTENTS

1 INTRODUCTION

Music recommender systems support the interaction of users with large music catalogs by automatically identifying and suggesting music than can be of interest to the users. To this end, they strongly rely on machine learning and data mining methods to analyze data describing the users, the music items, the interaction of the users with the music items, and even the users' context during the interaction with the music items [121, Chapter 7]. The main techniques utilized in recommender systems are in principle applicable to any item domain, but music recommender systems should additionally take the following domain particularities into consideration:

- Music can be recommended using different granularity levels (e.g., songs, albums, artists, genres, or ready-made playlists).
- Music is often consumed in listening sessions, such as albums or playlists. The local context within such sessions must be taken into consideration.
- Music recommender systems should adapt their level of interference to the user needs, from just supporting the exploration of the music catalog, to providing a full *lean-back* experience.

Two main fields of research have contributed to music recommender systems: *music information retrieval* and *recommender systems*. Research in the field of music information retrieval has focused on content-based recommender systems. In such systems, music items are represented by feature vectors (typically extracted from the audio signal [47, 93, 116], social tags [99], or web content [74]), and recommendations are predicted on the basis of pairwise song similarities computed using the song feature vectors. On the other hand, research in recommender systems and data mining has concentrated on the so-called "collaborative filtering," that is, the extraction of underlying music taste patterns from usage data (such as listening logs [168], radio stations [2, 25], or hand-curated music playlists [17]). Content-based recommender systems deliver fair but limited performance because the relations derived from content-wise similarities tend to be simple. Collaborative recommender systems capture more abstract music taste patterns, but their performance depends on the availability of sufficiently dense usage data.

The works presented in this thesis are located at the intersection of these two fields of research. I investigate hybrid music recommender systems that integrate the strengths of content-based and collaborative

recommender systems. While the systems proposed throughout the thesis are valid for other item domains, I carefully take into consideration the particularities of recommending music: I address music recommendation tasks at different item granularity levels; I exploit collaborative and content-based data characteristic of the music domain; I investigate fundamental properties of music listening sessions and of the songs therein.

1.1 OUTLINE

The thesis is divided into six chapters. This introductory chapter provides a summary of the thesis and its contributions, and it describes the organization of the contents in the subsequent chapters. Chapter 2 provides the technical background necessary to contextualize and follow the main thesis chapters. This technical background does not intend to be exhaustive, and the interested reader will find pointers to additional references through the thesis. Chapters 3, 4 and 5 contain the contributions of the thesis. They are thematically divided into two parts, each addressing a music recommendation task. The first part, corresponding to Chapter 3, focuses on the task of music artist recommendation and investigates how to model users' general music preferences over an extended period of time. The second part, corresponding to Chapters 4 and 5, focuses on the task of music playlist continuation and investigates how to recommend next songs fitting the local characteristics of listening sessions. Chapter 6 draws conclusions, discusses limitations of the presented works, and outlines future research directions.

The thesis is organized as a mixture between a monograph and a collection of articles. The dissertation can be read as a unified document with a coherent flow. However, Chapters 3, 4 and 5, the main contributions of the thesis, stand as independent units and are strongly based on works that have already been published or that will be published soon. Consequently, the reader will find a certain degree of repetition among these chapters, especially regarding their motivation and the description of some systems and datasets. Each of these chapters is prefaced with additional relevant details, including the corresponding published articles or system implementations.

1.2 CONTRIBUTIONS

The main contributions of the thesis can be summarized as follows:

- I propose a hybrid extension to matrix factorization that combines listening and tagging histories. For music artist recom-

mendation, the proposed system yields competitive or higher recommendation accuracy, especially for infrequent artists.

- I study fundamental properties of music playlists, namely the song context, the song order and the song popularity, and their impact on the prediction of playlist continuations. The insights derived are crucial to design playlist continuation systems.
- I conduct an in-depth analysis of the expressiveness of song-level features (derived from audio, social tags, and user–item interactions data) to predict playlist continuations.
- I propose two hybrid recommender systems integrating collaborative information with any type of song features. For playlist continuations, the proposed systems yield competitive or higher recommendation accuracy, especially for infrequent songs.
- I share implementations of the proposed systems to facilitate the reproducibility of the results.
- I propose, in summary, hybrid recommender systems adapting to the particularities of the music domain, which consistently strengthen the representation of infrequent music items and thus facilitate the discovery of music beyond the popular music *charts*.

Below I provide a more extended description of the main contributions and findings of each of the core chapters of the thesis.

1.2.1 Music artist recommendation

I propose a hybrid matrix factorization model for collaborative filtering on implicit feedback datasets extending a well-established purely collaborative model. The proposed hybrid system jointly factorizes listening logs and social tags from music streaming services, which are abundant sources of usage data in the music domain. According to the conducted evaluation, the proposed hybrid system yields more accurate artist recommendations than its purely collaborative counterpart, and it also outperforms another hybrid matrix factorization baseline previously proposed in the literature. A dedicated analysis shows that the superior performance of the proposed hybrid system is explained by its improved representation of music artists for which few observations are available at training time. I extend the evaluation methodology previously followed in the literature by incorporating bootstrap confidence intervals to facilitate the comparison between the performance of the systems.

1.2.2 Music playlist continuation

Like previous works on music playlist modeling, this research is based on the exploitation of hand-curated music playlists as rich examples from which to learn music compilation patterns.

Playlist characteristics

I analyze fundamental playlist characteristics, namely the song context length, the song order and the song popularity, and their impact on effectively predicting next-song recommendations. By comparing the performance of four existing playlist continuation models of increasing complexity on two datasets of hand-curated music playlists, I derive insights into the importance of these playlist characteristics for the task of next-song recommendation. According to the conducted evaluation, considering a longer song context has a positive impact on next-song recommendations. Furthermore, the long-tailed nature of the playlist datasets (common in music collections) makes simple playlist continuation models and highly-expressive playlist continuation models appear to perform comparably. However, a detailed analysis reveals the advantage of using highly-expressive models, especially to deal with songs in the "long tail" (i.e., infrequent songs). The evaluation also suggests either that the song order is not crucial for next-song recommendations, or that even highly-expressive models are unable to exploit it. I further propose a variation of the standard retrieval-based evaluation methodology previously followed in the literature that provides a more complete view of the performance of the systems.

Hybrid playlist continuation systems

I propose two hybrid music recommender systems for the task of music playlist continuation. Building on the previous findings, I design hybrid systems considering the full playlist song context and hybridizing collaborative and content-based recommender systems, thus improving the representation of infrequent songs.

I first identify suitable song-level feature representations to model whether songs fit in music playlists. Features derived from the audio signal, social tags, and independent listening logs are investigated. According to the conducted evaluation, the features derived from independent listening logs are more expressive than those derived from social tags, which in turn outperform those derived from the audio signal. The combination of features from different modalities outperforms the individual features, suggesting that the different modalities indeed carry complementary information.

The first of the proposed hybrid systems can properly represent infrequent songs, but it is limited to extending playlists for which a profile has been computed at training time. The second of the proposed

hybrid systems regards playlist–song pairs exclusively in terms of feature vectors. Thus, it learns general playlist–song membership relationships, which not only make it robust to dealing with infrequent songs but also enable the extension of playlists not seen at training time. I provide an extensive evaluation comparing the proposed hybrid systems to purely collaborative and hybrid baselines. The proposed hybrid systems predict more accurate playlist continuations as a result of their better integration of content and collaborative information, especially for infrequent songs.

1.3 MAIN PUBLICATIONS

The thesis is based on the following publications:

- Andreu Vall, Marcin Skowron, Peter Knees, and Markus Schedl. "Improving music recommendations with a weighted factorization of the tagging activity." In: *Proc. ISMIR*. Málaga, Spain, 2015, pp. 65–71,
- Andreu Vall. "Listener-inspired automated music playlist generation." In: *Proc. RecSys*. Vienna, Austria, 2015, pp. 387–390,
- Andreu Vall, Hamid Eghbal-zadeh, Matthias Dorfer, and Markus Schedl. "Timbral and semantic features for music playlists." In: *Machine Learning for Music Discovery Workshop at ICML*. New York, NY, USA, 2016,
- Andreu Vall, Markus Schedl, Gerhard Widmer, Massimo Quadrana, and Paolo Cremonesi. "The importance of song context in music playlists." In: *RecSys Poster Proceedings*. Como, Italy, 2017,
- 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: *Proc. DLRS Workshop at RecSys*. Como, Italy, 2017, pp. 46–54,
- Andreu Vall, Matthias Dorfer, Markus Schedl, and Gerhard Widmer. "A hybrid approach to music playlist continuation based on playlist-song membership." In: *Proc. SAC*. Pau, France, 2018, pp. 1374–1382,
- Andreu Vall, Massimo Quadrana, Markus Schedl, and Gerhard Widmer. "The importance of song context and song order in automated music playlist generation." In: *Proc. ICMPC-ESCOM*. Graz, Austria, 2018,
- Andreu Vall and Gerhard Widmer. "Machine learning approaches to hybrid music recommender systems." In: *Proc. ECML PKDD*. Dublin, Ireland, 2018,

- Andreu Vall, Matthias Dorfer, Hamid Eghbal-zadeh, Markus Schedl, Keki Burjorjee, and Gerhard Widmer. "Feature-combination hybrid recommender systems for automated music playlist continuation." In: *User Modeling and User-Adapted Interaction* (2019, in press),
- Andreu Vall, Massimo Quadrana, Markus Schedl, and Gerhard Widmer. "Order, context and popularity bias in next-song recommendations." In: *International Journal of Multimedia Information Retrieval* (2019, in revision).

1.4 ADDITIONAL PUBLICATIONS

While not included in the thesis, I also contributed to the following publications in the fields of user modeling for recommender systems, machine learning methods for information retrieval, and operations research:

- Katayoun Farrahi, Markus Schedl, Andreu Vall, David Hauger, and Marko Tkalčič. "Impact of listening behavior on music recommendation." In: *Proc. ISMIR*. Taipei, Taiwan, 2014, 483–488,
- Markus Schedl, Andreu Vall, and Katayoun Farrahi. "User geospatial context for music recommendation in microblogs." In: *Proc. SIGIR*. Gold Coast, QLD, Australia, 2014, pp. 987–990,
- Alejandro Lago, Victor Martínez-de-Albéniz, Philip Moscoso, and Andreu Vall. "The role of quick response in accelerating sales of fashion goods." In: *Analytical modeling research in fashion business*. Springer Series in Fashion Business. Springer, 2016, pp. 51–78,
- Bruce Ferwerda, Mark P. Graus, Andreu Vall, Marko Tkalčič, and Markus Schedl. "The influence of users' personality traits on satisfaction and attractiveness of diversified recommendation lists." In: *EMPIRE Workshop at RecSys*. Boston, MA, USA, 2016, pp. 43–47,
- Bruce Ferwerda, Andreu Vall, Marko Tkalčič, and Markus Schedl. "Exploring music diversity needs across countries." In: *Proc. UMAP*. Halifax, Nova Scotia, Canada, 2016, pp. 287–288,
- Bruce Ferwerda, Mark P. Graus, Andreu Vall, Marko Tkalčič, and Markus Schedl. "How item discovery enabled by diversity leads to increased recommendation list attractiveness." In: *Proc. SAC*. Marrakech, Morocco, 2017, pp. 1693–1696,

- Matthias Dorfer, Jan Schlüter, Andreu Vall, Filip Korzeniowski, and Gerhard Widmer. "End-to-end cross-modality retrieval with CCA projections and pairwise ranking loss." In: *International Journal of Multimedia Information Retrieval* 7.2 (2018), pp. 117–128.

2 MUSIC RECOMMENDATION

Current on-line music platforms provide access to tens of millions of songs. Consequently, recommender systems have become a key technology to assist users to navigate such large music catalogs. This chapter describes the fundamental aspects of music recommender systems. It starts with an introduction to recommender systems, their components, and the main recommendation techniques considered in the thesis. This introduction is valid for any item domain. The chapter then continues with the particularities of dealing with music items, such as their representation, usual consumption patterns, or the data sources typically utilized to derive music taste patterns. The chapter concludes with an overview of the main approaches to evaluating music recommender systems.

2.1 RECOMMENDER SYSTEMS

Recommender systems are technologies devised to support the interaction of users with large collections of items,[1] specifically by suggesting relevant items to the users accessing the collection. The rapid development of recommender systems over the past years is to a good extent motivated by the increasingly larger availability of items in on-line content distributors [5]. Already in 2003, Linden, Smith, and York [89] described how recommender systems enabled personalized suggestions to Amazon customers. In 2007, the Netflix Prize [10] led to important advances in matrix factorization models to estimate the users' preferences for movies [76, 169], and similar models would be successfully applied to other item domains as well (e.g., recommending music [36], news [112], or digital television shows [60]).

Recommender systems encompass contributions from multiple, diverse disciplines. An important goal of recommender systems is to support the users' access to information of their interest. In this sense, recommender systems are regarded as specializations of more general tasks in the interconnected fields of information filtering and information retrieval [8, 52, 123]. Recommender systems strongly rely on the analysis of data describing the users, the items, the interaction of users with items, and even the users' context when interacting with the items. Thus, data mining and machine learning methods play an important role in recommender systems [121, Chapter 7].

1 The term "items" refers to any type of material goods (e.g., a CD or a vinyl record), digital goods (e.g., music streaming) or services (e.g., concerts in the users' proximity).

Beyond purely data-driven techniques, the field of human-computer interaction is gaining importance in recommenders systems. This responds to the observation that systems optimized to achieve high accuracy in simulated recommendation tasks (usually called "off-line experiments") do not necessarily provide the most useful recommendations to the end users [102]. Ultimately, the effectiveness of recommender systems also depends on additional factors such as the users' trust in the system [139], the system interface [29], and user-centered aspects such as personality and emotions [121, Chapter 21].

Recommender systems are applied to different item domains, including on-line shopping, music streaming, news reading, or job search. Therefore, domain-specific techniques, such as music signal processing [106] for music recommendation [74, 85, 109], or image processing [78, 140] for on-line shopping [55] or video recommendation [56], are also utilized, especially for the extraction of rich item representations.

2.1.1 Components

The design of each recommender system depends on the recommendation scenario, the expected interaction of the user with the system, the item domain, etc. Still, most recommender systems share some basic characteristics, illustrated in Figure 2.1. On the one hand, a provider distributes items, which may be of its own or belonging to a third party (e.g., a digital newspaper distributes its own contents, while a news aggregator distributes contents from other media). On the other hand, users interested in these items access the platform. As in information retrieval systems, it is assumed that the users have preferences [8]. The users' preferences possibly relate to regular interests which can be, at the same time, varied and conditional on particular needs and situations. For example, a user of a music streaming service is an enthusiast of electronic music. She listens to *industrial techno* to focus on work, to *deep house* while having dinner, and to *nu jazz* any time.[2]

Recommender systems derive summarizations or "profiles" of the users' preferences and of the items, which facilitate the use of computational approaches to assess the users' potential interest towards items. A common approach to recommend items to a user consists in directly comparing the user's preference profile to all item profiles. Then, the most relevant items are recommended. Other approaches involve comparing the user's preference profile to other user preference profiles, or comparing the profiles of items that the user consumed in the past to other item profiles. These comparisons enable recommendation approaches based on user–user similarities ("users like you liked Y") and item–item similarities ("if you like X you may like Y").

2 http://techno.org/electronic-music-guide

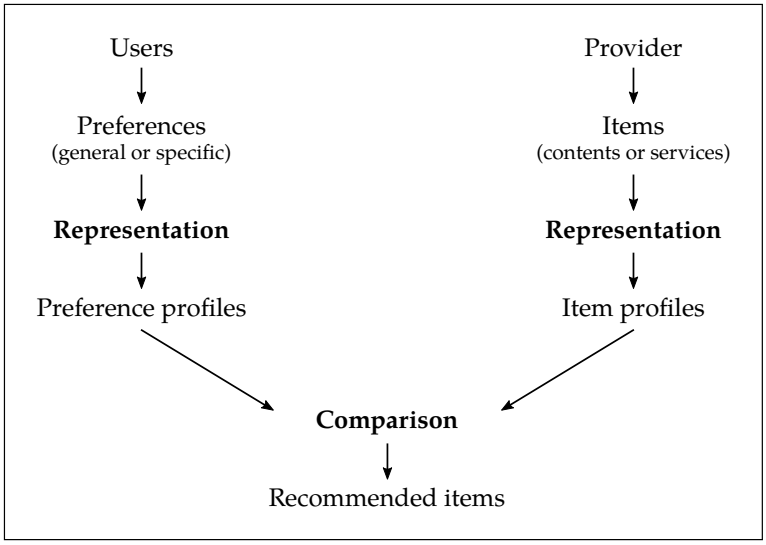


Figure 2.1: Basic components of a recommender system (adapted from Belkin and Croft [8], and Celma [21]).

2.1.2 Recommendation approaches

Recommender systems share essential components (Figure 2.1) but differ in the user and item information exploited, the representation techniques used to derive user preference and item profiles, and the procedure followed to assess which items should be recommended to a user. The user and item information considered in the first place are of particular importance because they define the meaning of the extracted profiles and of the subsequent profile comparisons. Recommender systems are classified into the following main approaches depending on the information they exploit: collaborative recommender systems (usually named "collaborative filtering"), content-based recommender systems, and hybrid recommender systems.

Content-based recommender systems

Content-based approaches utilize explicit descriptions of the item content (e.g., the audio signal in songs [47, 93] or the text in books [105]) to derive item profiles. User preferences are implicitly defined by the user's history of consumed items, from which a user preference profile is derived. The comparison of the user preference profile to the item profiles is indirect: items with a similar content to items that the user previously consumed are expected to be of interest to the user.

Similarly, demographic recommender systems assume that users' personal attributes (e.g., age, gender or language) can help determine their interests [113], and knowledge-based recommender systems match explicit user requirements to detailed item information (e.g., suggesting financial services constrained by the user willingness to take risks) [121, Chapter 5]. Even though demographic and knowledge-

based approaches are sometimes regarded separately [15, 121], they are strongly related to content-based recommender systems in that they rely on explicit user or item descriptions.

Collaborative recommender systems

Collaborative approaches assume that user preferences and items are implicitly characterized by the collective history of user–item interactions (e.g., music listening logs [168], bookmarked websites [112], or rated movies [104]) and exploit them to derive user preference profiles and item profiles. In contrast to content-based approaches, collaborative filtering disregards explicit descriptions of the user preferences and of the items. Depending on how the user preference profiles and the item profiles are built and compared, collaborative approaches are further categorized as neighbors-based or model-based.

Neighbors-based collaborative recommender systems profile the users by the items they consumed, and the items by the users who consumed them. They assume that two users have similar preferences if their histories of consumed items overlap, and two items are similar if they have been consumed by approximately the same group of users [121, Chapter 2]. The comparison of a user preference profile to item profiles is indirect: approaches based on user neighborhoods recommend items consumed by like-minded users [132], while approaches based on item neighborhoods recommend items similar to items that the user previously consumed [126].

Model-based collaborative recommender systems derive user preference profiles and item profiles by applying statistical models to the collective history of user–item interactions. This usually enables the direct comparison of user preference profiles to item profiles to decide which items to recommend. An extensively researched family of methods within this category are matrix factorization models. Models based on least-squares solutions provide competitive performance and scale well with the size of the data [60, 76, 112]. Yet further factorization models have been proposed, such as maximum-margin matrix factorization [137, 165], factorization machines [118], or probabilistic [2, 104, 120] and fully Bayesian [124] matrix factorization. Collaborative filtering is however not restricted to the matrix factorization family. Other collaborative recommender systems have been proposed based on, e.g., restricted Boltzmann machines [125], latent variable models [59, 168], logistic embeddings [25], or variational autoencoders [86].

Hybrid recommender systems

Collaborative recommender systems reveal abstract relations beyond content-wise item similarities, but their performance depends on the availability of rich user–item interactions. New users and new

items for which interactions are scarce suffer from the so-called "cold-start" problem: they are poorly represented and recommendations become unfeasible or not reliable [121, Chapter 8]. Content-based recommender systems identify simpler relations, but they are more robust to the lack of rich user–item interactions. Building the preference profile for a new user is still challenging, but new items do not suffer from the cold-start problem because they are profiled on the basis of an explicit item description. User preferences would also be explicitly profiled in demographic and knowledge-based recommender systems.

Content-based and collaborative recommender systems can be combined to take advantage of the strengths of each individual approach. Users and items with enough coverage in the history of user–item interactions mostly rely on the exploitation of collaborative patterns. However, poorly represented users and items benefit from additional explicit descriptions. There exist different hybridization approaches. An obvious one consists in combining independent recommender systems, for example by weighting their predicted scores [53], or by re-ranking the recommendations of one of the systems with information from the other [63]. Other approaches provide a tighter integration by incorporating content-based features into a collaborative recommender system [55, 109], or even tighter by fusing collaborative and content-based information into a standalone recommender system [56, 163].

Further information on the general aspects of recommender systems can be found in the introductory chapter of the *Recommender systems handbook* [121, Chapter 1], and in the surveys by Adomavicius and Tuzhilin [1], and Bobadilla et al. [15].

2.2 THE MUSIC DOMAIN

The principles of recommender systems described so far are generally valid regardless of the item domain. However, each item domain has inherent characteristics that determine the item availability, the expected consumption patterns, the relevant recommendation scenarios, and the sources of information available. These particularities must be considered in the design, development and evaluation of recommender systems.

We understand music recommender systems as tools that assist users to find music for casual situations and enjoyment [121, Chapter 13]. These should not be confused with music retrieval systems pursuing related but different goals, such as assisting music creators to retrieve recordings from large libraries [4, 73], or suggesting variations on a musical composition [28, 162].

2.2.1 Representation levels

Songs are the most fine-grained recommendable music items. However, more general musical entities such as albums or artists are often considered as well. Songs can even be arranged into more abstract and subjective entities such as ready-made playlists, genres, moods or activities. Different representation levels should be considered depending on the specific recommendation task. For example: playlist continuation systems must suggest songs to extend a given playlist [17]; however, users browsing a music collection may have a better experience if music is arranged into entities more general than songs, such as genres [24]; further, a recommender system with an emphasis on music discovery may be more effective by pointing users to artists that they do not know yet, rather than focusing on specific songs [98].

2.2.2 Sequential consumption

Compared to watching a movie, or reading a book, the time needed to listen to a song is short. Still, a listening session typically consists of several songs, such as an album, a ready-made playlist, or a playlist extended progressively like a radio stream. Music recommender systems must be aware of the sequential nature of listening sessions (e.g., by maintaining session profiles) and need to be responsive to the most recent songs played. Another particular consumption behavior is the recurrent listening of the same songs, which may be partly explained by the little effort necessary to listen to a song.

2.2.3 Recommendation scenarios

Music platforms equipped with recommender systems usually consider different scenarios depending on the users' engagement with the current listening session and on their willingness to interact with a recommender system (Figure 2.2). Users knowing which music they want to listen to can just search it using text-based retrieval tools (if the platform allows). This scenario does not require the intervention of a recommender system. Other users may be unsure of which music they want to listen to but still be interested in actively exploring the music collection. In such cases, the platform can provide not too intrusive support by arranging the music items into abstract representation levels, or by pointing the users to music items similar to those currently inspected (e.g., similar artists or related playlists) [72, 115]. For users more inclined to receive suggestions from a recommender system, a common approach (not only in the music domain) consists in letting a recommender system rank the items in the collection according to the user preferences. The most relevant items are shown to the users, who still examine them before making a decision [30, 108]. Another scenario

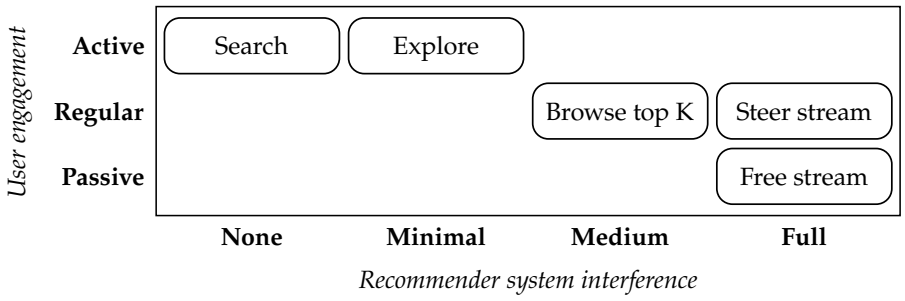


Figure 2.2: Recommendation scenarios.

(mostly in the music domain) consists in recommending playlists, or music streams. Users can steer such music streams by providing a starting music item [47, 116] or a partial playlist that should be continued [17, 53, 63], and also by interacting with the recommender system during the listening session [95, 111, 166]. Lastly, users can opt for a *lean-back* experience letting the recommender system freely extend the music stream.

2.2.4 Data sources

This section focuses on the types of data exploited by collaborative and content-based music recommender systems, which are the focus of this thesis.

Collaborative patterns in music

There exist different types of user–item interactions in the music domain from which to extract collaborative patterns. The most abundant source of user–item interactions are listening logs from on-line music streaming services, that describe which users listened to which music items [85, 86, 101, 168]. This is a form of "implicit" feedback, that is, feedback tracked from the users but not actively provided by the users [60]. Another rich but less abundant source of interactions are hand-curated music playlists [17, 53, 99, 100] and Internet radio streams [2, 25]. While a single playlist typically reflects individual preferences, a collection of playlists constitutes a source of collaborative information containing valuable information about which songs fit well together. Listening logs and curated music playlists are forms of "one-class" feedback, that is, feedback providing only positive opinions about items [112]. Implicit and one-class feedback contrast with "explicit" feedback, which is actively provided by the users and expresses both positive and negative opinions about items (e.g., ratings, or *thumbs up/down*). Explicit feedback is not abundant in the music domain, with the notable exception of the "Yahoo! Music Dataset" [36, 37].

Content of music items

The audio signal of songs is the most fundamental, low-level representation of the content of music items, and it can be mined using music signal processing techniques. A common approach to exploit the audio signal of a song starts by computing its spectrogram [106]. The spectrogram can be further processed to obtain Mel-frequency Cepstrum Coefficients (MFCCs) [92]. MFCCs have been widely used in the fields of music information retrieval and music recommendation in combination with Gaussian mixture models [6, 47, 98], vector quantization [58, 85, 97], or to extract "i-vectors" [39, 40]. More recently, spectrograms have been directly utilized for feature extraction and prediction using convolutional neural networks [27, 109].

Manual annotations also provide rich descriptions of the content of music items. The most simple example of such descriptions are editorial metadata, which can inform about a recording's release date, tempo, genre, or instrumentation [16, 63, 114]. A more complex type of manual annotations are unstructured text descriptions, such as music-related web content [74, 128, 145], or the *social tags* that users of on-line platforms assign to music items [46, 48, 51, 53]. Such text descriptions provide richer, more abstract representations than metadata, but they tend to be noisy and subjective. Song representations extracted from unstructured text descriptions have often been based on the vector space model with *tf-idf* weighting [96, Chapter 6], and more recently on word embedding techniques [103].

Further information on the specific characteristics of the music domain can be found in the *Recommender systems handbook* [121, Chapter 13], and in the survey by Schedl et al. [129], which are both specifically devoted to music recommender systems.

2.3 EVALUATION

2.3.1 On-line experiments

Recommender systems support the interaction of users with large item catalogs. Therefore, it is meaningful to have them evaluated by actual end users. On-line experiments, sometimes categorized as user studies or on-line studies [121, Chapter 8], seek such user feedback. User studies typically refer to small- or medium-scale experiments where participants are asked about different aspects of a recommender system, like usability, or satisfaction. On-line experiments typically refer to large-scale studies where users of an on-line platform interact with a recommender system, and the effectiveness of the system is measured with the usual metrics of interest of the platform.

User studies are essential before deploying a recommender system, but they can also be costly for initial developing stages. Especially chal-

lenging is hiring participants. For example, Liebman, Saar-Tsechansky, and Stone [87], Wang et al. [164], and Xing, Wang, and Wang [166] conducted user studies to evaluate music recommender systems based on reinforcement learning techniques, where receiving regular user feedback is essential. Kamehkhosh and Jannach [69] also conducted a user study to evaluate music playlist continuation systems. However, the participants of all these studies were invited undergradute and graduate students who may not constitute an unbiased sample of the actual end users. A recommended alternative to hire participants are crowd-sourcing platforms [121, Chapter 9], but also in this case it is important to select participants with good reputation and to introduce control questions to filter out careless contributions. For example, Ferwerda et al. [44, 45] conducted user studies on the "Amazon Mechanical Turk"[3] to investigate the effects of user personality and diversification on the attractiveness of the recommendations.

Large-scale on-line experiments provide the closest approximation to the final system performance. They of course require having access to an actual on-line platform, such as in the works by Li et al. [82], and Shani, Heckerman, and Brafman [131] (for news and book recommendations, respectively).

2.3.2 Off-line experiments

Historical user–item interactions enable the evaluation of recommender systems as predictive models. Off-line experiments compare the predictions of recommender systems to withheld user–item interactions, thus simulating the interaction with users and approximating the interest of the users towards the system recommendations. In the music domain, where the user feedback is mostly implicit, off-line experiments usually focus on the ability of the recommender systems to recover withheld music items known to be of interest to the users. That is, the recommendation task is approximated by a retrieval task.

The retrieval-based evaluation consists in repeating the following procedure for every user. A number of music items known to be of interest to the user are withheld. The music recommender system predicts the interest of the user towards all the items in the music collection. The items are ranked according to their predicted interest. A good recommender system is expected to place the withheld items in top positions of the ranked list (close to position 1). The performance of the recommender system is then typically summarized using rank-based metrics, such as precision [24, 53, 85], recall [17, 53, 63], mean average precision [24, 97, 110], mean percentile rank [60, 68], mean reciprocal rank [63, 64, 97], or normalized discounted cumulative gain [85, 86] (for each metric just a few works are cited as examples).

3 https://www.mturk.com

Rank-based metrics only capture the accuracy of recommender systems predicting the interest of users towards items. Additional metrics have been incorporated to off-line experiments trying to quantify other relevant factors beyond the accuracy of the predictions, such as the diversity, serendipity, novelty, and coverage of the recommendations [159, 167, 170]. While off-line experiments are in general more limited than on-line experiments to assess the ultimate user satisfaction, they have important advantages: they are inexpensive, and provide a controlled and reproducible framework to evaluate and systematically compare recommender systems.

Further information on the evaluation of recommender systems can be found in the *Recommender systems handbook* [121, Chapters 8 and 9], or in the works by Valcarce et al. [146] for an in-depth study of accuracy metrics, and Kaminskas and Bridge [70] for a survey on beyond-accuracy metrics.

Part I

MUSIC ARTIST RECOMMENDATION

3 MATRIX CO-FACTORIZATION

SUMMARY

Collaborative filtering systems for music recommendations are often based on implicit feedback derived from listening activity. Hybrid approaches further incorporate additional sources of information in order to improve the quality of the recommendations. In the context of a music streaming service, we present a hybrid model based on matrix factorization techniques that fuses the implicit feedback derived from the users' listening activity with the tags that users have given to music items. In contrast to existing work, we introduce a novel approach to exploit tags by performing a weighted factorization of the tagging activity. We evaluate the model for the task of artist recommendation, using the expected percentile rank as metric, extended with confidence intervals to enable the comparison between models. The proposed model consistently outperforms a pure collaborative filtering baseline, as well as a hybrid baseline that also combines listening logs and tagging activity. A closer analysis indicates that the improved performance of the proposed model can be explained by its ability to mitigate the cold-start problem for new artists.

REMARKS

This chapter focuses on the task of music artist recommendation as an approximation to modeling the users' general music preferences over an extended period of time. The works presented here correspond to early stages of my research and focus on understanding and extending matrix factorization models for implicit feedback data, which were, and still are, a strong recommender system baseline. Certainly, the integration of listening and tagging activity could be accomplished with other machine learning approaches, like factorization machines [118], or hybrid systems based on neural networks as presented in Chapter 5.

The chapter is based on the following publications:

- Andreu Vall, Marcin Skowron, Peter Knees, and Markus Schedl. "Improving music recommendations with a weighted factorization of the tagging activity." In: *Proc. ISMIR*. Málaga, Spain, 2015, pp. 65–71,
- Andreu Vall. "Listener-inspired automated music playlist generation." In: *Proc. RecSys*. Vienna, Austria, 2015, pp. 387–390,

The parts of both publications relevant to this chapter are combined and reproduced with minor revisions. I conceived and conducted the research presented in this chapter. Nevertheless, I maintain the use of the first-person plural ("we") and acknowledge the collaboration of the co-authors. Marcin Skowron contributed the careful preparation of the tagging activity, by processing and matching user and artist social tags. Peter Knees shared inspiring early research on the incorporation of metadata to matrix factorization models. Markus Schedl contributed to the data acquisition and provided valuable feedback.

Implementations of the proposed matrix co-factorization and baseline models are made available:

- Andreu Vall. *Matrix factorization models with social tags for music recommendation*. Original date: 2015-11-19. URL: https://github.com/andreuvall/WeightedTags-MF (visited on 12/12/2018).

3.1 INTRODUCTION AND RELATED WORK

We provide the motivation of our work together with a review of the relevant related work, divided into three parts. First, we introduce the types of user feedback under consideration. Then, we present the family of models we use to build recommender systems. Finally, we review the evaluation methodology.

3.1.1 Explicit, implicit and one-class feedback

The interactions between users and items provide a useful source of data to produce recommendations [121]. It is commonly accepted to distinguish between "explicit feedback" and "implicit feedback," depending on whether the user actively provides feedback about an item or this is tracked from the user's interaction with the system [1]. Examples of explicit feedback are rating a movie, giving a "like" to a blog post, or tagging an artist, because the user actively provides an opinion. In contrast, the listening histories of users in a music streaming service are an example of implicit feedback.

The standard approach to make use of implicit feedback is to count or aggregate all the interactions for each user–item pair [41, 60, 68], yielding a user–item–count table. In structure, this is identical to an explicit feedback user–item–rating table. We henceforth refer to such data structure as "user–item interactions matrix," regardless of the type of feedback (implicit or explicit).

In some cases a user–item interaction can express both positive and negative opinions, in other cases it only reflects positive (or active) examples. Ratings in a 1 to 5 scale conventionally range from strongly disliking an item to strongly liking it. However, tracking whether a user visited or not a website only provides binary feedback describing

action or inaction. Binary feedback is often referred to as "one-class feedback" [83, 112, 135], and examples of it can be found both in explicit and in implicit feedback. For example, a user–item interactions matrix (be it from explicit or implicit feedback) contains intrinsically a source of one-class feedback, revealing which user–item pairs were observed and which not.

Inaction must not be confused with a negative opinion, because a user may not have interacted with an item for a variety of reasons, not necessarily because of lack of interest. Social tags also exhibit this property, and treating this correctly will be a key point of the presented model.

3.1.2 Matrix factorization for collaborative filtering

Collaborative filtering is a widely used recommendation method which aims at recommending the most relevant items to a user based on relations learned from previous interactions between users and items [121, Chapter 3]. The factorization of the user–item interactions matrix into latent factors matrices is a well-established technique to implement collaborative recommender systems, both for explicit feedback and implicit feedback datasets [60, 76, 112]. Compared to other methods, it has the advantage of uncovering latent data structures by solving an optimization problem, instead of using problem-specific and manually designed features.

Specific collaborative systems for implicit feedback data based on matrix factorization techniques are presented by Hu, Koren, and Volinsky [60], and Pan et al. [112]. The key technique is to use appropriate weights in the low-rank approximation of the user–item interactions matrix. More specifically, even if the weighting schemes are different, both works propose to assign higher confidence to the observed user–item pairs and lower (but still positive) confidence to the unobserved user–item pairs. This is important to handle the uncertainty derived from the one-class property described before. We will insist on this point later, because our improved treatment of the tagging activity will rest on the same principle.

3.1.3 Hybrid recommender systems

In collaborative filtering implementations based on matrix factorization techniques, hybrid models can be based on the simultaneous factorization of the user–item interactions matrix, together with other data for users and items [41, 94]. The motivation for that is that latent factors summarizing user and item properties should be reinforced, or better described, if other data sources related to the same users and items are involved in the optimization problem.

The tags that users assign to music items — or other forms of textual data, like user profiles, or genre annotations for the items — are an obvious example of potentially useful additional information. In this line, the research presented by Fang and Si [41] is a valid starting point, dealing with implicit feedback data and hybridized with user and item profiles, built on the basis of *tf-idf* weights calculated for each user, each item, and each considered word in a dictionary.

Tagging information is an explicit source of feedback (because users actively provide it) that exhibits, at the same time, the one-class property described before; the tags assigned to music items are only positive examples (even if the meaning of a tag is semantically negative). A particular tag may not have been applied to a musical item, but this does not imply that the tag is not suited to describe that musical item. This property of social tags is also referred to as "weak labeling" [144]. It is reasonable to assume though, that the more often a tag has been applied to a musical item, the more it should be trusted. Similarly, if a user applies a tag very often, it may be assumed that the tag is to some extent relevant for her to describe music items. To address the uncertainty that arises from this wide range of possibilities, we propose to exploit the tagging activity with a weighted matrix factorization scheme similar to the one applied for collaborative filtering in implicit feedback datasets. Observed tags can be given higher confidence. Unobserved tags can be given lower confidence, but still positive, so that they are not ignored in the recommender system.

3.1.4 Evaluation of recommender systems

The Netflix Prize [10] has motivated an important progress in the domain of collaborative filtering, but probably due to the specific approach considered in the challenge, research has centered on attaining maximum levels of accuracy in the prediction of ratings. However, improvements in predictive accuracy do not always translate to improved user satisfaction [102].

To make the evaluation task more similar to a real use case (although still in an off-line experiment), Koren [75] evaluates different recommender systems on the basis of issued ranked lists of recommendations. A recommender able to rank first the relevant items should be considered better than a recommender that is not able to do so. An extension of this evaluation methodology to deal with implicit feedback datasets is proposed by Hu, Koren, and Volinsky [60] and applied by Li et al. [83] and Johnson [68]. It consists in a central tendency measure, called "expected percentile rank," assessing how good is the recommender at identifying relevant items.

The expected percentile rank is a valid metric to measure the average behavior of a single recommender system, but in order to compare the performance of different recommender systems, considering only

mean values can be inaccurate. We propose to use bootstrapping techniques to examine the distribution of the expected percentile rank and test for significant differences between models.

3.2 METHODOLOGY

This work is framed in the context of music streaming services in which users interact with music items, mainly listening to music, but also through the free input of text describing them. We focus on the task of artist recommendations. The listening data is aggregated at the artist level, obtaining a user–artist–count matrix of implicit feedback. The tagging activity yields a user–artist–tag matrix of one-class feedback, processed to obtain: a user–tag–count matrix, describing how many times a user applied a tag, and an artist–tag–count matrix, describing how many times a tag was applied to an artist. The proposed model is actually flexible regarding the tagging activity data. In our experiments, we successfully use a collection of top used tags (not a complete list of all the used tags) together with weights describing the tag relevance (instead of actual counts).

3.2.1 Matrix factorization models

We compare three recommender systems based on matrix factorization. The first is a standard collaborative filtering model for implicit feedback data. The second is a hybrid model incorporating textual data, that we modify for the specific task of using tags. Finally, we introduce a novel model, able to improve the quality of the recommendations through a weighted factorization of the tagging activity.

Matrix factorization ("MF")

We use the approach described by Hu, Koren, and Volinsky [60] to perform collaborative filtering on implicit feedback data. It consists in a weighted low-rank approximation of the user–artist–count matrix, adjusting the confidence of each user–artist pair as a function of the count. Given a system with N users and M artists, the counts for each user–artist pair are tabulated in a matrix $R \in \mathbb{N}^{N \times M}$, where users are stored row-wise and artists column-wise. A binary matrix $\widetilde{R}$ is defined, such that for each user u and each artist a

$$\widetilde{R}_{ua} = \begin{cases} 1 & \text{if } R_{ua} > 0 \\ 0 & \text{if } R_{ua} = 0, \end{cases} \tag{3.1}$$

and the following weight function is defined as

$$w(\eta, x) = 1 + \eta \log(1 + x). \tag{3.2}$$

Other weight functions can be defined and may better suit each specific problem and distribution of the data. We choose a logarithmic relation, and not the also common linear relation [41, 60, 68], to counteract the long-tailed distribution of the data, where a majority of users have a small percentage of the total observed interactions. However, the optimization of this function is not within the scope of this work.

Finally, the matrix factorization consists in finding two D-rank matrices $P \in \mathbb{R}^{N \times D}$ and $Q \in \mathbb{R}^{M \times D}$ (rows are latent features for users and artists, respectively) minimizing the following cost function:

$$J_{MF}(P, Q) = \sum_{ua \in R} w(\alpha, R_{ua}) \left(\widetilde{R}_{ua} - P_u Q_a^T\right)^2 + \lambda \left(\|P\|_F^2 + \|Q\|_F^2\right). \tag{3.3}$$

The matrix $\widetilde{R}$ is reconstructed using P and Q. $\widetilde{R}_{ua}$ is the entry of $\widetilde{R}$ corresponding to user u and artist a. P_u is the row of P corresponding to user u, and Q_a is the row of Q corresponding to artist a. The squared reconstruction error is weighted using a function of the actual counts in R_{ua} according to Equation (3.2), and it is summed over all the user–artist pairs.[1] The parameter α contributes to the weight function and is determined by grid search. A regularization term involving the Frobenius norm of P and Q is added to prevent the model from over-fitting. The regularization parameter λ is also determined by grid search.

Matrix factorization with tagging activity ("TMF")

Equation (3.3) is extended by Fang and Si [41] to incorporate textual information. We present a modification of their model to specifically deal with tags. Given a system where T tags have been used, the counts for each user–tag pair are stored in a matrix $T^U \in \mathbb{N}^{N \times T}$, where rows correspond to users and columns correspond to tags. The counts for each artist–tag pair are stored in a matrix $T^A \in \mathbb{N}^{M \times T}$, where rows correspond to artists and columns correspond to tags. The modified model factorizes together $\widetilde{R}$, T^U and T^A into three D-rank matrices $P \in \mathbb{R}^{N \times D}$, $Q \in \mathbb{R}^{M \times D}$ and $X \in \mathbb{R}^{T \times D}$ (rows are latent features for

1 This includes the zero entries of R as well [60].

users, artists and tags respectively) minimizing the following cost function:

$$
\begin{aligned}
J_{TMF}(P, Q, X) = & \sum_{ua \in R} w(\alpha, R_{ua}) \left(\widetilde{R}_{ua} - P_u Q_a^T\right)^2 \\
& + \mu_1 \sum_{ut \in T^U} \left(T^U_{ut} - P_u X_t^T\right)^2 \\
& + \mu_2 \sum_{at \in T^A} \left(T^A_{at} - Q_a X_t^T\right)^2 \\
& + \lambda \left(\|P\|_F^2 + \|Q\|_F^2 + \|X\|_F^2 \right).
\end{aligned}
\tag{3.4}
$$

The first term is identical as in Equation (3.3). The second and third terms account for the contribution of tags. X_t is the row of X corresponding to tag t. Matrices T^U and T^A are reconstructed using P, Q and X, and the squared reconstruction errors are summed over all user–tag pairs and artist–tag pairs. The parameters μ_1, μ_2 account for the contribution of each term to the cost function and are determined by grid search. The regularization term is analogous as in Equation (3.3).

This formulation modifies the one proposed by Fang and Si [41], in that it factorizes T^U and T^A using a single shared tags' factor matrix X, instead of two dedicated factor matrices. The tagging activity consists of user–artist–tag observations. Even if we use separated user–tag–count and artist–tag–count matrices as inputs for the model, the tags must be factorized in the same space of latent features.

This model factorizes the user–tag and artist–tag raw counts. If, for example, an artist–tag pair has never been observed, the model will try to fit a value of 0 counts for it. This seems an unsuited model, because we know that a tag that has not been applied may still be relevant.

Matrix factorization with weighted tagging activity ("WTMF")

We introduce a novel approach to improve the hybridization with tagging activity by using a weighted factorization scheme similar to the one used for implicit feedback data. The observed user–tag and artist–tag pairs are given high confidence and therefore have a higher contribution to the cost function. The unobserved user–tag and artist–tag pairs are given low confidence. They become less relevant in the cost function, and at the same time the model has more freedom to fit them. As the results in Section 3.3.3 demonstrate, this is a better approach to model the weak labeling property of social tags.

We define binary matrices $\widetilde{T}^U$ and $\widetilde{T}^A$, such that for each user u, each artist a and each tag t

$$\begin{aligned}\widetilde{T}^U_{ut} &= \begin{cases} 1 & \text{if } T^U_{ut} > 0 \\ 0 & \text{if } T^U_{ut} = 0, \end{cases} \\ \widetilde{T}^A_{at} &= \begin{cases} 1 & \text{if } T^A_{at} > 0 \\ 0 & \text{if } T^A_{at} = 0. \end{cases}\end{aligned} \tag{3.5}$$

We factorize together $\widetilde{R}$, $\widetilde{T}^U$ and $\widetilde{T}^A$ into three D-rank matrices $P \in \mathbb{R}^{N\times D}$, $Q \in \mathbb{R}^{M\times D}$ and $X \in \mathbb{R}^{T\times D}$ (rows are latent features for users, artists and tags respectively) minimizing the following cost function:

$$\begin{aligned} J_{WTMF}(P,Q,X) = &\sum_{ua\in R} w(\alpha, R_{ua}) \left(\widetilde{R}_{ua} - P_u Q_a^T\right)^2 \\ &+ \mu_1 \sum_{ut\in T^U} w(\beta, T^U_{ut}) \left(\widetilde{T}^U_{ut} - P_u X_t^T\right)^2 \\ &+ \mu_2 \sum_{at\in T^A} w(\gamma, T^A_{at}) \left(\widetilde{T}^A_{at} - Q_a X_t^T\right)^2 \\ &+ \lambda\left(\|P\|_F^2 + \|Q\|_F^2 + \|X\|_F^2\right). \end{aligned} \tag{3.6}$$

This is similar to Equation (3.4), but now all the terms involve a weighted factorization. Note that the second and third terms have specific weight coefficients β and γ, determined by grid search.

3.2.2 Parameter estimation

Alternating Least Squares (ALS) is usually the preferred method to minimize the objective functions of models based on matrix factorization [9, 41, 50, 60, 68, 112, 169]. ALS is an iterative method, where subsequently all but one of the factor matrices are kept fixed. This results in quadratic functions that approximate the original one. At each step, the cost value is expected to move closer to a local minimum, and the process is repeated until convergence. Since the approximated functions are quadratic, the exact solution for the factors can be computed in closed form.

For each of the presented models, we provide the exact solution for the factors of each user u stored in P_u, each artist a stored in Q_a, and each tag t stored in X_t. We introduce some additional notation. R_{r_u}, R_{c_a}, $T^U_{r_a}$, $T^U_{c_t}$, $T^A_{r_a}$, $T^A_{c_t}$ refer to the u^{th}, a^{th}, t^{th} row or column (r, c) of the corresponding matrix (R, T^U, T^A).[2] We also need to define the following matrices:

- $W^{r_u}_R \in \mathbb{R}^{M\times M}$ is a diagonal matrix with the weights computed for the u^{th} row of R in the diagonal,

2 T^U and T^A may be further transposed, reading T^{UT} and T^{AT}.

- $W_R^{c_a} \in \mathbb{R}^{N \times N}$ is a diagonal matrix with the weights computed for the a^{th} column of R in the diagonal,
- $W_{T^U}^{r_u} \in \mathbb{R}^{T \times T}$ is a diagonal matrix with the weights computed for the u^{th} row of T^U in the diagonal,
- $W_{T^U}^{c_t} \in \mathbb{R}^{N \times N}$ is a diagonal matrix with the weights computed for the t^{th} column of T^U in the diagonal,
- $W_{T^A}^{r_a} \in \mathbb{R}^{T \times T}$ is a diagonal matrix with the weights computed for the a^{th} row of T^A in the diagonal,
- $W_{T^A}^{c_t} \in \mathbb{R}^{M \times M}$ is a diagonal matrix with the weights computed for the t^{th} column of T^A in the diagonal.

Solution for J_{MF}

For each user u and artist a, the latent factors are given by

$$\begin{cases} P_u = \left(Q^T W_R^{r_u} Q + \lambda I\right)^{-1} \left(Q^T W_R^{r_u} R_{r_u}^T\right) \\ Q_a = \left(P^T W_R^{c_a} P + \lambda I\right)^{-1} \left(P^T W_R^{c_a} R_{c_a}^T\right). \end{cases} \tag{3.7}$$

Solution for J_{TMF}

For each user u, artist a and tag t, the latent factors are given by

$$\begin{cases} P_u = \left(Q^T W_R^{r_u} Q + \mu_1 X^T X + \lambda I\right)^{-1} \\ \qquad\qquad \left(Q^T W_R^{r_u} R_{r_u}^T + \mu_1 X^T T_{r_a}^{UT}\right) \\ Q_a = \left(P^T W_R^{c_a} P + \mu_2 X^T X + \lambda I\right)^{-1} \\ \qquad\qquad \left(P^T W_R^{c_a} R_{c_a}^T + \mu_2 X^T T_{r_a}^{AT}\right) \\ X_t = \left(\mu_1 P^T P + \mu_2 Q^T Q + \lambda\right)^{-1} \\ \qquad\qquad \left(\mu_1 P^T T_{c_t}^{UT} + \mu_2 Q^T T_{c_t}^{AT}\right). \end{cases} \tag{3.8}$$

Solution for J_{WTMF}

For each user u, artist a and tag t, the latent factors are given by

$$\begin{cases} P_u = \left(Q^T W_R^{r_u} Q + \mu_1 X^T W_{T^U}^{r_u} X + \lambda I\right)^{-1} \\ \qquad\qquad \left(Q^T W_R^{r_u} R_{r_u}^T + \mu_1 X^T W_{T^U}^{r_u} T_{r_a}^{UT}\right) \\ Q_a = \left(P^T W_R^{c_a} P + \mu_2 X^T W_{T^A}^{r_a} X + \lambda I\right)^{-1} \\ \qquad\qquad \left(P^T W_R^{c_a} R_{c_a}^T + \mu_2 X^T W_{T^A}^{r_a} T_{r_a}^{AT}\right) \\ X_t = \left(\mu_1 P^T W_{T^U}^{c_t} P + \mu_2 Q^T W_{T^A}^{c_t} Q + \lambda\right)^{-1} \\ \qquad\qquad \left(\mu_1 P^T W_{T^U}^{c_t} T_{c_t}^{UT} + \mu_2 Q^T W_{T^A}^{c_t} T_{c_t}^{AT}\right). \end{cases} \tag{3.9}$$

3.2.3 Producing recommendations

The technique employed to produce recommendations is the same for all the models. Once the factor matrices P, Q and X are learned, the user–artist preferences are predicted as $Z = PQ^T$. Note that the tags' factor matrix X is not directly involved in the prediction although it contributed to a better estimation of P and Q. The new matrix Z is expected to be a reconstruction of $\widetilde{R}$ for the observed user–artist pairs. For unobserved entries, Z is expected to reveal potential preferences on the basis of the learned user and artist factors. The closer a predicted user–artist preference is to 1, the more confidence we have that it corresponds to an interesting artist for the user. For each user u, a recommendation list is prepared showing the artists with higher predicted preference values in Z_u.

3.3 EXPERIMENTAL STUDY

3.3.1 Dataset

We compare the different models on a dataset of Last.fm listening histories, top tags used by users and top tags applied to artists, collected through the Last.fm API.[3] The combination of the standard Taste Profile Subset[4] with the Last.fm tags dataset[5] would seem a preferable choice, but the absence of users' tagging activity makes it unsuited.

The dataset is built as a stable subset of a running crawl of Last.fm listening events. The original crawl includes only users with non-empty country information, non-empty gender information and a value in the age field between 10 and 80 years, although such filtering is actually not needed. There is no constraint on the minimum or maximum number of artists a user has listened to. However, we only include users such that at least 95% of their listened artists have a valid MusicBrainz[6] identifier. This is to ensure that enough artists are properly identified and that we will be able to accurately crawl their tagging activity from the Last.fm. This does not bias the dataset towards popular artists because MusicBrainz is an open and collaborative platform that includes a wide variety of artists. The users' tagging activity is fetched with the Last.fm user names.

The dataset includes 21,852,559 listening events, relating to 2,902 users and 71,223 artists, yielding 687,833 non-zero user–artist–count entries. This corresponds to a matrix density of roughly 0.3%. Table 3.1

3 http://www.last.fm/api
4 http://labrosa.ee.columbia.edu/millionsong/tasteprofile
5 http://labrosa.ee.columbia.edu/millionsong/lastfm
6 https://musicbrainz.org

Table 3.1: Distribution of users per number of listened artists.

listened artists	users
1–10	64
11–20	84
21–30	122
31–40	77
41–50	96
50–100	466
101–2332	1993
total	2902

shows the distribution of users as a function of the number of artists they listened to.

The top tags for each user (if any) are provided together with a count variable describing how many times the user applied it. The top tags applied to an artist (if any) are provided together with a percentage relative to the most frequently applied tag [81]. Because the API functions only return the top tags, we only observe a partial set of the tagging activity. In addition, although the user is presented with previously used tags, she can always input free text. To overcome these limitations, we perform regularization and simplification operations to the tag strings, namely: replacements of genre abbreviations with their extended version, spelling corrections, removal of non-alphanumeric characters, and mapping of different spelling variants to a unique tag string, resulting in a unified set of tokens. After this process is applied to the fetched tags, we are left with 630 unique tags for 600 users and 12,902 unique tags for 67,332 artists, among which 494 unique tags are identified as identical between the user and the artist list. Note that tags were found for most of the artists, but only for 20% of the users. Probably, only a small subset of active users use the tagging functionality.

The whole matrix of user–artist counts is used, although not all users or artists have related tagging activity. Tags are a complement whenever they are available.

3.3.2 Evaluation methodology

The most reliable evaluation method for a recommender system is an actual large-scale on-line experiment, where real users interact with the system [121, Chapter 8]. This requires a complex infrastructure which, unfortunately, is not within the scope of this work. Since we only have access to historical data, we can not measure how new

recommendations would be perceived by the users. Furthermore, in contrast to explicit feedback applications, accuracy metrics for predicted ratings are not meaningful for implicit feedback. Therefore, we adopt the evaluation approach proposed by Koren [75] and adapted by Hu, Koren, and Volinsky [60] to deal with implicit feedback datasets in a recall-oriented setting, and we additionally propose an extension to it.

The observed user–artist pairs are split into training and test sets to perform 5-fold cross validation, letting each user have approximately 80% of the listened artists in the training set and 20% in the test set. For each user–artist pair u, a assigned to the test set, a random list of artists (not including a) is drawn. The list is then ranked according to the preferences of user u, learned from the training set as explained in Section 3.2. Finally, a is inserted in the sorted list, and its percentile rank within the list is stored as rank_{ua}.[7] If a is ranked among the top positions of the list, then its percentile rank is close to 0%. If it is ranked in last positions, then its percentile rank is close to 100%.

After this process has been performed over all the splits, rank_{ua} is known for all the observed user–artist pairs in the dataset. Then, the *expected percentile rank* is defined as the weighted average of rank_{ua} with weights given by the user–artist counts [41, 60, 68]:

$$\overline{\mathrm{rank}} = \frac{\sum_{ua \in R} R_{ua} \mathrm{rank}_{ua}}{\sum_{ua \in R} R_{ua}}. \tag{3.10}$$

Correctly ranking a highly relevant artist is more important than correctly ranking a less relevant artist. Likewise, failing to recommend a highly relevant artist is worse than failing to recommend a less relevant one. Values of $\overline{\mathrm{rank}}$ close to 0% indicate that the recommender is able to correctly rank the relevant artists. Producing ranked lists uniformly at random results in an expected percentile rank of 50%. Ranking all the relevant items in the last position of the list results in an expected percentile rank of 100%.

We extend the evaluation methodology by building confidence intervals of $\overline{\mathrm{rank}}$. This allow us to test for significant differences in the performance of models. We use basic bootstrap confidence intervals, based on the bootstrap distribution of the expected percentile rank. For all the observed user–artist pairs in the dataset, random samples with replacement and with the same size as the dataset are drawn. For each sample of user–artist pairs, the expected percentile rank is computed. We repeat this step 1,000 times to obtain the bootstrap distribution of $\overline{\mathrm{rank}}$. We then build 95% confidence intervals of $\overline{\mathrm{rank}}$ using the basic bootstrap scheme [33].

7 Lists of any length may be prepared, and the percentile rank provides a unified scale. We use lists of 100 artists in our experiments. According to our experience, longer lists do not yield significant differences.

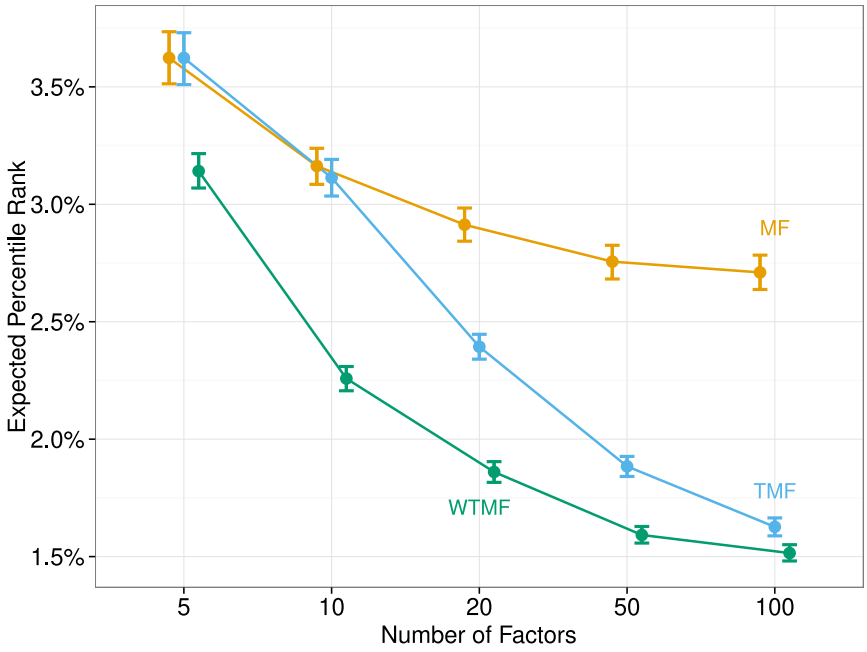


Figure 3.1: Model comparison for different number of latent factors. The dots correspond to $\overline{\text{rank}}$ and the error bars display 95% basic bootstrap confidence intervals. The different models are dodged to avoid overlapping. The top and center lines correspond to the baseline models. The lowest corresponds to the presented model.

3.3.3 Model comparison

The models are evaluated and compared for a varying number of latent factors D and for a varying number of training iterations. On the one hand, we fix the number of iterations to 10 and evaluate the models with 5, 10, 20, 50, 100 latent factors. On the other hand, we fix the number of factors to 10 and evaluate the models for 5, 10, 20, 50, 100 training iterations. We choose 10 factors and 10 training iterations as a basic setting, because they balance well performance and computational requirements.

For each model and each combination of factors and iterations, we tune the parameters α, β, γ, μ_1, μ_2 and λ by grid search. We choose the set of values that provides lowest expected percentile rank, computed by 5-fold cross validation as described in Section 3.3.2. Figures 3.1 and 3.2 show the results for different number of factors and iterations, respectively.

Note that all models, including the plain matrix factorization model, provide very good results, with values of expected percentile rank under 4%. This implies that, on average, the models are able to rank relevant artists among the top 4 positions of a list of 100 random artists.

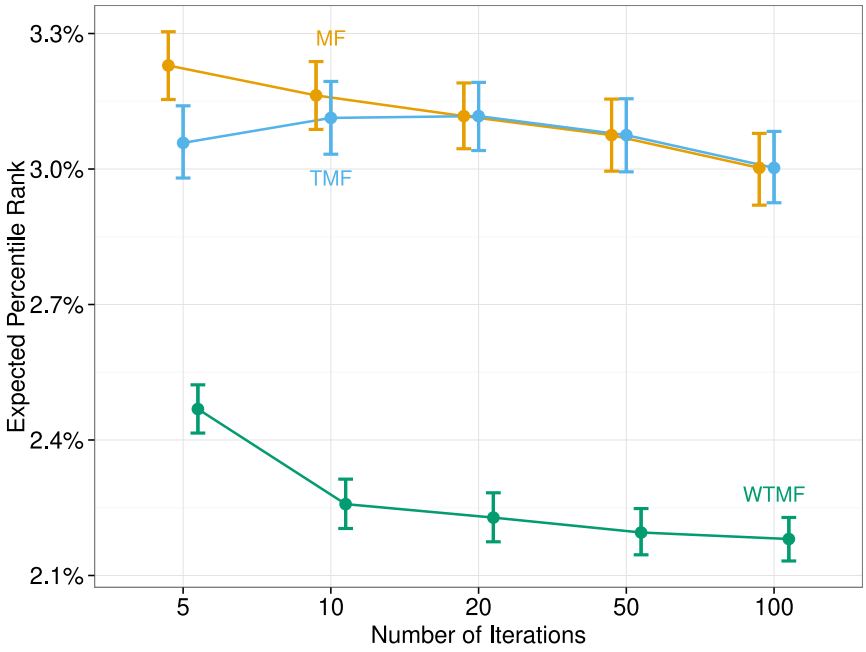


Figure 3.2: Model comparison for different number of training iterations. The dots correspond to $\overline{\text{rank}}$ and the error bars display 95% basic bootstrap confidence intervals. The different models are dodged to avoid overlapping. The top lines correspond to the baseline models. The lowest corresponds to the presented model.

The performance of TMF and WTMF improves significantly when more latent factors are used (Figure 3.1). The presented model outperforms the baselines, although for 100 factors the difference between TMF and WTMF is small. We examine this case. We compute a 95% basic bootstrap confidence interval for the difference of $\overline{\text{rank}}$ and it does not include 0. We conclude that the difference in performance is significant. For lower number of factors the differences between the presented model and the baselines are remarkable. Good performance at inexpensive computational requirements is a crucial property, especially if the recommender is implemented for large-scale applications.

Increasing the number of training iterations results in smaller improvements (Figure 3.2). TMF is not even monotonically decreasing. After 5 iterations, the cost function for TMF is already close to a local minimum and further minimization does not translate to significant differences in the expected percentile rank. For the experiments with 20 or more training iterations MF performs exactly as well as TMF. This is because the grid search process finds that discarding the tagging activity yields best results for those cases. Using only 10 factors as a basic setting, TMF is not able to exploit the tagging activity. Our model clearly outperforms the baselines in this set of experiments too, with a difference of 1% in expected percentile rank.

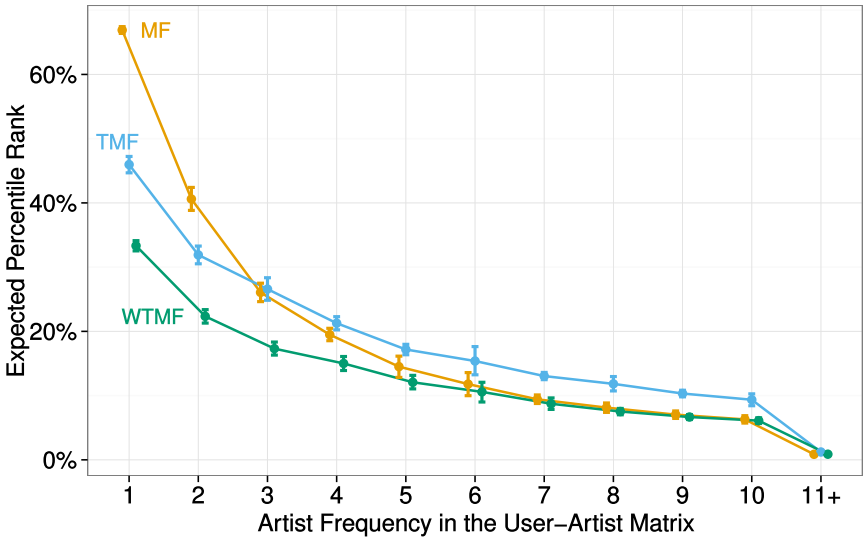


Figure 3.3: Model comparison as a function of the number of observed occurrences of the artists in the dataset. The dots correspond to $\overline{\mathrm{rank}}$ and the error bars display 95% basic bootstrap confidence intervals. The different models are dodged to avoid overlapping. The top lines correspond to the baseline model. The lower corresponds to the proposed model.

3.3.4 Cold start

We explore the effect of the number of observed occurrences of an artist in the dataset on the performance of the recommender systems. If an artist has only been listened by few users, plain collaborative filtering has little information on the relations between this artist and the users. Then, hybridizing with tags will prove advantageous.

Each user–artist pair in the dataset is assigned to a subset defined by the number of occurrences of the artist in the dataset. Equation (3.10) is computed for each subset, summing only over the corresponding user–artist pairs. Figure 3.3 shows the expected percentile rank for MF, TMF and WTMF for each subset. Artists with a single occurrence in the dataset are poorly recommended by MF. TMF obtains superior performance than MF for artists with one and two occurrences in the dataset, but its performance quickly decreases and becomes even slightly worse than the pure baseline MF. Thanks to the incorporation of the weighted tagging activity, WTMF clearly outperforms MF and TMF for artists with up to 5 occurrences in the dataset. The more often an artist is observed in the dataset, the smaller the difference of performance becomes.

We provide an example to give a qualitative explanation of the effect of using tags. We train MF, TMF and WTMF using 80% of the data. Table 3.2 shows the predicted preference z_{ua} of a selected user u for four different artists he or she listened to, but belong to the 20% of data withheld for test. MF is not able to identify that the

Table 3.2: Prediction of MF, TMF and WTMF for the preference Z_{ua} of a selected user for artists known to be relevant for the user but hidden from the training.

artist name	Z_{ua}^{MF}	Z_{ua}^{TMF}	Z_{ua}^{WTMF}
Feliu Ventura	0.00	0.09	0.61
Joan Colomo	0.35	0.58	0.75
Manos de Topo	0.23	0.44	0.64
Mazoni	0.00	0.22	0.69

first and the fourth artist were interesting for the user. The reason is that these artists were not observed at all in the training set. For the second and third artists, MF predicts low values. TMF has a weak belief on the artists without listening examples and becomes slightly more confident for the other two. WTMF is fairly confident about the four artists. Comparing the artist tags in the user's training data with the artist tags of these four artists we find several coincidences, like "Catalan," "indie pop" or "pop surrealista" (surreal pop). WTMF is able to identify these relations.

3.4 CONCLUSIONS

We have presented a novel matrix factorization model to incorporate tagging activity into implicit feedback recommender systems. Our approach consistently outperformed both pure and hybrid collaborative filtering baselines on experiments conducted with real data from Last.fm, a well-known music streaming service. Analyzing the results as a function of the artist frequency in the dataset, we have found that the proposed weighted scheme is able to deal with infrequent artists by establishing connections through shared social tags. We have extended the common evaluation methodology computing basic bootstrap confidence intervals for the expected percentile rank, allowing us to test for significant differences in the performance of models.

Part II

MUSIC PLAYLIST CONTINUATION

4 PLAYLIST CHARACTERISTICS

SUMMARY

The automated continuation of music playlists can be naturally regarded as a sequential task where a recommender system suggests a stream of songs that constitute a listening session. Instead of capturing general user preferences, the task of music playlist continuation focuses on revealing specific session preferences encoded in the most-recent user interactions. While the accuracy achieved in recommendations is important, in this work we shift our focus towards a deeper understanding of fundamental playlist characteristics, namely the song context length, the song order and the song popularity, and their relation to the recommendation of playlist continuations. We also propose an approach to assess the quality of the recommendations that mitigates known problems of off-line experiments for music recommender systems. Our results indicate that knowing a longer song context has a positive impact on next-song recommendations. We find that the long-tailed nature of the playlist datasets makes simple and highly-expressive playlist continuation models appear to perform comparably, but further analysis reveals the advantage of using highly-expressive models. Finally, our experiments suggest either that the song order is not crucial for next-song recommendations, or that even highly-expressive models are unable to exploit it.

REMARKS

This is the first of two chapters devoted to the task of music playlist continuation. The focus moves from understanding users' general music preferences to modeling local properties of listening sessions. Here I investigate the importance of fundamental playlist characteristics on next-song recommendations. The derived insights will be crucial in the upcoming Chapter 5 to determine which components should be build into music recommender systems for music playlist continuation. This research initially yielded the following publications:

- Andreu Vall, Markus Schedl, Gerhard Widmer, Massimo Quadrana, and Paolo Cremonesi. "The importance of song context in music playlists." In: *RecSys Poster Proceedings*. Como, Italy, 2017,
- Andreu Vall, Massimo Quadrana, Markus Schedl, and Gerhard Widmer. "The importance of song context and song order in

automated music playlist generation." In: *Proc. ICMPC-ESCOM*. Graz, Austria, 2018.

The current chapter extends these works and is considered for publication subject to minor revisions:

- Andreu Vall, Massimo Quadrana, Markus Schedl, and Gerhard Widmer. "Order, context and popularity bias in next-song recommendations." In: *International Journal of Multimedia Information Retrieval* (2019, in revision).

I conceived and conducted the research presented in this chapter. Nevertheless, I maintain the use of the first-person plural ("we") and acknowledge the collaboration of the co-authors. Massimo Quadrana contributed to the implementation of the experiments and provided continuous feedback and expertise on sequence-aware recommender systems. Markus Schedl and Gerhard Widmer provided valuable feedback.

4.1 INTRODUCTION

Automated music playlist continuation is a specific task in music recommender systems where the user sequentially receives song recommendations, producing a listening experience similar to traditional radio broadcasting. The sequential recommendation scenario is very natural in the music domain, where a listening session typically includes several songs.

According to interviews with practitioners and postings to a dedicated playlist-sharing website, Cunningham, Bainbridge, and Falconer [31] identified the choice of songs and the song order as important aspects of the playlist curation process. As we review in Section 4.2, some approaches to automated playlist continuation take into account the current and previous songs in the playlist (i.e., the *song context*[1]) and the order of the songs in the playlist to recommend the next song. However, to the best of our knowledge, previous works do not explicitly analyze the impact of exploiting the song context and the song order for next-song recommendations.

In this work, we compare four well-established and widely-used playlist continuation models: a popularity-based model, a song-based Collaborative Filtering (CF) model, a playlist-based CF model, and a Recurrent-Neural-Network-based model (RNN). These playlist continuation models are of increasing complexity and, by design, are able to exploit the song context and the song order to different extents. By

1 We refer to the current and previous songs in the playlist as the *song context* as it is commonly done in language models, but this should not be confused with the incorporation of general contextual information into recommender systems.

analyzing and comparing their performance on different off-line experiments, we derive insights regarding the impact of the song context and the song order for next-song recommendations and the necessity to be aware of the bias towards popular songs. For the evaluation of the off-line experiments, we propose to use metrics derived from complete recommendation lists, instead of from top K recommendations. This provides a more complete view on the performance of the playlist continuation models.

The remainder of this chapter is organized as follows. Section 4.2 reviews the related work on automated music playlist continuation. Section 4.3 introduces the guidelines for the off-line experiments conducted throughout this work. We describe the recommendation task that the playlist continuation models must fulfill and define the metrics employed to assess their performance on the task. Section 4.4 describes the four playlist continuation models considered. Section 4.5 presents the datasets of hand-curated music playlists on which we conduct the off-line experiments. Section 4.6 elaborates on the results of the off-line experiments and is divided into three parts, which discuss the impact of the song context, the popularity bias and the song order on next-song recommendations, respectively. Conclusions are drawn in Section 4.7.

4.2 RELATED WORK

A well-researched approach to automated music playlist continuation relies on the song content. Pairwise song similarities are computed on the basis of features extracted from the audio signal (possibly enriched with social tags and metadata) and used to enforce content-wise smooth transitions [47, 74, 93, 99, 116]. Recommendations based on content similarity are expected to yield coherent playlists. However, pure content-based recommendations can not capture complex relations and, in fact, it does not hold in general that the songs in a playlist should all sound similar [80].

Playlist continuation has also been regarded as a form of Collaborative Filtering (CF), making the analogy that playlists are equivalent to users' listening histories, on the basis of which songs should be recommended. Playlist-based nearest-neighbors CF models and factorization-based CF models exploit the full playlist context [2, 17, 53]. Song-based nearest-neighbors CF models [126] are not common in the playlist continuation literature. However, Hidasi et al. [57] show (for e-commerce and video streaming) that an item-based CF model that predicts the next item on the basis of the current item can effectively deal with *short* histories, such as music playlists. In general, CF models disregard the song order, but it is worth noting that the model presented by Aizenberg, Koren, and Somekh [2] accounts for neighbor-

ing songs, and the model introduced by Rendle, Freudenthaler, and Schmidt-Thieme [119] (for on-line shopping) is aware of sequential behavior.

The Latent Markov Embedding introduced by Chen et al. [25] models playlists as Markov chains. It projects songs into a Euclidean space such that the distance between two projected songs represents their transition probability. The importance of the direction of song transitions is evaluated by testing a model on actual playlists and on playlists with reversed transitions, yielding comparable performance in both cases. McFee and Lanckriet [100] also treat playlists as Markov chains, modeled as random walks on song hypergraphs, where the edges are derived from multimodal song features, and the weights are learned from hand-curated music playlists. The importance of modeling song transitions is assessed by learning the hypergraph weights again but treating the playlists as a collection of song singletons. When song transitions are ignored, the performance degrades. These works examine the importance of accounting for song transitions and their order, but the Markovian assumption implies that only adjacent songs are considered.

Hariri, Mobasher, and Burke [53] represent songs by latent topics extracted from song-level social tags. Sequential pattern mining is performed at the topic level, so that given seed songs, a next topic can be predicted. Re-ranking the results of a CF model with the predicted latent topics is found to outperform the plain CF model. This approach considers the ordering but only at the topic level, which is more abstract than the song level.

Hidasi et al. [57] propose an approach to sequential recommendation based on the combination of Recurrent Neural Networks (RNNs) with ranking-aware loss functions (for e-commerce and video streaming). This approach has gained attention and it has been further improved and extended [56, 142]. Jannach and Ludewig [64] have applied it to the task of automated music playlist continuation in a study that compares the performance of RNN models and session-based nearest-neighbors models for sequential recommendation. Among other analyses, Jannach and Ludewig question whether the computational complexity of RNN models is justified. Recommendation models based on RNNs consider the full context of the sequence and are also aware of the order.

For a comprehensive survey on automated music playlist continuation, we point the interested reader to Bonnin and Jannach [17] and Ricci, Rokach, and Shapira [121, Chapter13].

We conducted preliminary studies preceding this work analyzing the importance of the song order and the song context in music playlists [154, 156]. This chapter covers these topics and further extends them by incorporating a playlist-based CF model, additional exper-

iments on the song order, a complete analysis of the impact of the popularity bias, and the detailed configurations of the playlist models.

4.3 EXPERIMENTAL DESIGN

We compare four well-established and widely-used playlist continuation models: a popularity-based model, a song-based CF model, a playlist-based CF model, and an RNN model (Section 4.4). We assess their ability to recover withheld playlist continuations following the off-line evaluation methodology proposed by Hidasi et al. [57] (Section 4.3.1). By comparing the performance of the different playlist continuation models on the same experiment, or the performance of the same model on different experiments, we reason about the importance of considering the song context and the song order for next-song recommendations. Furthermore, we study the impact of the song popularity on the performance of the different models.

Off-line evaluation approaches estimate the models' behavior on the recommendation task. However, they might not be able to estimate the final user satisfaction. The aim of this work is to provide insights on the importance of the order, the context and the popularity bias on next-song recommendations. The off-line evaluation methodology followed in this work serves this analysis well, because it allows a systematic comparison of the playlist continuation models under controlled conditions.

4.3.1 Evaluation methodology

A collection of music playlists is split into training and test playlists. A trained playlist continuation model (Section 4.4) is evaluated by repeating the following procedure over all the test playlists, which for clarity, we describe alongside the example depicted in Figure 4.1. We consider a test playlist (e.g., $p = (s_3, s_5, s_2)$). In the first step, we show the model the first song in the playlist (s_3). The model ranks all the songs in the dataset according to their likelihood to be the second song in the playlist. We keep track of the rank attained by the actual second song in the playlist (s_5 attains rank 2). We also keep track of the fact that this is a prediction for a song in second position. In the second step, we show the model the first and the second actual songs in the playlist (s_3, s_5). The model ranks all the songs in the dataset according to their likelihood to be the third song in the playlist. We keep track of the rank attained by the actual third song in the playlist (s_2 attains rank 3), etc. In this way, we progress until the end of the playlist, always keeping track of the rank attained by the actual next song in the playlist and the position in the playlist for which the prediction is made.

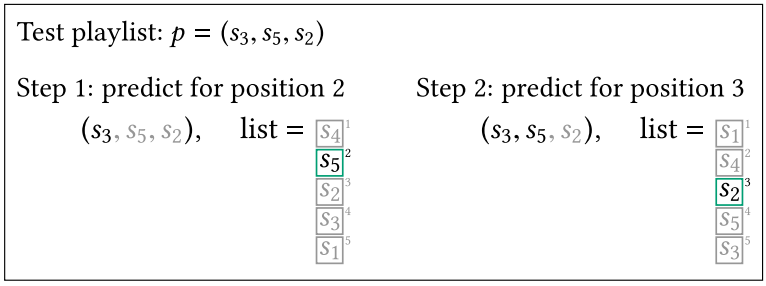


Figure 4.1: Illustration of the evaluation methodology. The playlist continuation model is evaluated on the test playlist $p = (s_3, s_5, s_2)$. It progresses through p and ranks all the songs in the dataset according to their likelihood to be the next song. The actual second song, s_5, attains rank 2. The actual third song, s_2, attains rank 3.

We index the ordered list of next-song candidates from 1 (most likely) until N (least likely), where N is the number of unique songs in the dataset. A good playlist continuation model is expected to rank the actual next song in top positions (small rank values). On the other hand, a poor model would rank the actual next song on bottom positions (large rank values). A random model would, on average, rank the actual next song on positions around $N/2$.

4.3.2 Assessing the quality of the recommendations

Previous research in automated music playlist continuation has summarized the distribution of attained ranks using metrics derived from the top K positions of the ordered lists of next-song candidates. For example, the recall at K (also named "hit rate" at K) is defined as the proportion of times that the actual next songs in the test playlists attain a rank lower than K [17, 53, 63, 64]. The rationale behind fixing the length K is that, in practice, only the top K results are of interest to the end user of the recommender system. However, Platt et al. [114] and McFee and Lanckriet [99] pointed out the following limitation of off-line evaluation approaches for playlist continuation models: a playlist may be extended by a number of potentially relevant songs, but off-line retrieval experiments only accept the exact match to the actual next song. The rank attained by the actual next song can be overly pessimistic, because the ordered list of next-song candidates can contain relevant results in better positions.

We propose to summarize the distribution of attained ranks using metrics derived from the whole ordered lists of next-song candidates, as opposed to focusing only on the top K positions of the lists. We believe that this provides a more complete view on the performance of the playlist continuation models. We report the distribution of attained ranks by means of boxplots representing the first quartile, median, and third quartile rank values (see e.g., Figure 4.2). Alternatively, we

Table 4.1: Summary of the playlist continuation models. The context length is the number of songs considered by the model to predict the next song (n means all the songs shown to the model). Order awareness indicates if the model regards the order of songs in playlists.

playlist model	context length	order awareness
Song Popularity	0	✗
Song-based CF	1	✗
Playlist-based CF	n	✗
RNN	n	✓

report only the median rank value if this facilitates the interpretation of the results (Figure 4.4).

4.4 PLAYLIST MODELS

We describe the four well-established and widely-used playlist continuation models involved in our experiments. By design, the models are of increasing complexity and are able to exploit the song context and the song order to different extents (Table 4.1). Hyperparameter tuning, if necessary, is performed on validation playlists withheld from the training playlists.

4.4.1 Song popularity

This is a unigram model that computes the popularity of a song s according to its relative frequency in the training playlists, i.e.,

$$\text{pop}(s) = \frac{|P_{tr}(s)|}{|P_{tr}|}, \tag{4.1}$$

where P_{tr} is the set of training playlists, $P_{tr}(s)$ is the subset of training playlists that contain the song s, and $|\cdot|$ denotes the number of playlists in each set. Given a test playlist, the next-song candidates are ranked by their popularity, disregarding the previous songs and their order. Despite its simplicity, the popularity-based model is a competitive playlist continuation model [17, 25].

4.4.2 Song-based collaborative filtering

This is a CF model based on song-to-song similarities. A song s is represented by a binary vector $\mathbf{p}_s$ that indicates the training playlists

to which it belongs. The similarity of a pair of songs s, t is computed as the cosine between $\mathbf{p}_s$ and $\mathbf{p}_t$, i.e.,

$$\text{sim}(s, t) = \cos(\mathbf{p}_s, \mathbf{p}_t) = \frac{\mathbf{p}_s \cdot \mathbf{p}_t}{\|\mathbf{p}_s\|\|\mathbf{p}_t\|}.$$

Two songs are similar if they co-occur in training playlists, regardless of the positions they occupy in the playlists. We follow Hidasi et al. [57] and implement the song-based CF model such that next-song candidates are ranked according to their similarity to the current song in the playlist only, ignoring previous songs. This approach is relatively simple, but Hidasi et al. show its competitive performance for sequential recommendation on short sessions.

4.4.3 Playlist-based collaborative filtering

This is a CF model based on playlist-to-playlist similarities. A playlist p is represented by a binary vector $\mathbf{s}_p$ indicating the songs that it includes. The similarity of a pair of playlists p, q is computed as the cosine between $\mathbf{s}_p$ and $\mathbf{s}_q$, i,.e.,

$$\text{sim}(p, q) = \cos(\mathbf{s}_p, \mathbf{s}_q) = \frac{\mathbf{s}_p \cdot \mathbf{s}_q}{\|\mathbf{s}_p\|\|\mathbf{s}_q\|}. \tag{4.2}$$

The score assigned to a song s as a candidate to extend a test playlist p is computed as

$$\text{score}(s, p) = \sum_{q \in P_{tr}(s)} \text{sim}(p, q), \tag{4.3}$$

where $P_{tr}(s)$ is the subset of training playlists that contain the song s. This model considers a song to be a suitable continuation for playlist p if it has occurred in training playlists that are similar to p. The similarity between playlists (4.2) and the score of a song to extend a playlist (4.3) depend on the full playlist p, i.e., on the full song context, but they disregard the song order.

Playlist-based CF has proven to be a competitive playlist continuation model [17, 53, 63, 64]. It usually has an additional parameter defining the number of most-similar training playlists on which the score (4.3) is calculated. We use all the training playlists because we find that this yields best performance in our experiments (Appendix 4.A.3).

4.4.4 Recurrent neural networks

Recurrent Neural Networks (RNNs) are a class of neural network models particularly suited to learn from sequential data. They have a hidden state that accounts for the input at each time step while recurrently incorporating information from previous hidden states.

We point the interested reader to Lipton and Berkowitz [91] for a review of RNN models.

We adopt the approach and implementation[2] proposed by Hidasi et al. [57], where an RNN model with one layer of Gated Recurrent Units (GRU) [26] is combined with a loss function designed to optimize the ranking of next-item recommendations. The model hyperparameters and architecture are detailed in Appendix 4.A.4.

Given a test playlist, the RNN model considers the full song context and the song order and outputs a vector of song scores used to rank the next-song candidates.

4.5 DATASETS

We evaluate the four playlist continuation models on two datasets of hand-curated music playlists derived from the on-line playlist-sharing platforms "Art of the Mix"[3] and "8tracks."[4] Both platforms allow music aficionados to publish their playlists on-line. Moreover, the Art of the Mix platform hosted forums and blogs for discussion about playlist curation, as well as social functionalities such as favoriting, or providing direct feedback to a user.[5] The 8tracks platform also provides social functionalities, such as following users, liking, or commenting on specific playlists. Previous works in the automated music playlist continuation literature have chosen to work with collections derived from the Art of the Mix and the 8tracks databases because of their presumably careful curation process [17, 53, 63, 99, 100]. As an illustration of the users' engagement, we refer the interested reader to the study presented by Cunningham, Bainbridge, and Falconer [31], that analyzes posts to the Art of the Mix forums requesting advice on, e.g., the choice of songs, or ordering rules.

The "AotM-2011" dataset [100] is a publicly available playlist collection derived from the Art of the Mix database. Each playlist is represented by song titles and artist names, linked to the corresponding identifiers of the Million Song Dataset[6] (MSD) [12], where available. The "8tracks" dataset is a private playlists collection derived from 8tracks. Each playlist is represented by song titles and artist names. Since we find multiple spellings for the same song-artist pairs, we use fuzzy string matching to resolve the song titles and artist names against the MSD, adapting the code released by Jansson, Raffel, and Weyde [65] for a very similar task.

2 https://github.com/hidasib/GRU4Rec
3 http://www.artofthemix.org
4 https://8tracks.com
5 Publishing playlists and interacting with individual users are still active services on the Art of the Mix, but the forums and blogs seem to be discontinued.
6 https://labrosa.ee.columbia.edu/millionsong

Table 4.2: Descriptive statistics of the filtered AotM-2011 and 8tracks playlist collections. We report the distribution of playlist lengths, number of artists per playlist, and song frequency in the dataset (i.e., the number of playlists in which each song occurs).

dataset	statistic	min	1q	med	3q	max
AotM-2011	Playlist length	5	6	7	8	34
	Artists per playlist	3	5	7	8	34
	Song frequency	1	8	12	20	249
8tracks	Playlist length	5	5	6	7	46
	Artists per playlist	3	5	6	7	41
	Song frequency	1	9	15	30	2320

We use the MSD as a common name space to correctly identify song-artist pairs. In both datasets, the songs that could not be resolved against the MSD are discarded, with one of two possible approaches. The first approach consists in simply removing the non-matched songs. The original playlists are preserved but with skips within them, which we ignore. The second approach consists in breaking up the original playlists into segments of consecutive matched songs, yielding shorter playlists without skips. We base our analysis on the first approach, but experiments on the second approach yield the same conclusions.

We keep only the playlists with at least 3 unique artists and with a maximum of 2 songs per artist. This is to discard artist- or album-themed playlists, which may correspond to a careless compilation process. For example, such a playlist could be the result of saving a full album as a playlist, which does not involve any curation effort. We also keep only the playlists with at least 5 songs to ensure a minimum playlist length. Songs occurring in less than 10 playlists are removed to ensure that the models have sufficient observations for each song.

We randomly assign 80% of the playlists for training and the remaining 20% for test. As in any recommendation task blind to item content, the songs that occur only in test playlists need to be removed because they can not be modeled at training time. This affects the final playlist length and song frequency.

The filtered AotM-2011 dataset has 17,178 playlists with 7,032 unique songs by 2,208 artists. The filtered 8tracks dataset has 76,759 playlists with 15,649 unique songs by 4,290 artists. Table 4.2 reports the distribution of unique songs per playlist, unique artists per playlist and song frequency in the datasets.

4.6 RESULTS

We elaborate on the off-line experiments and the derived findings on the importance of the song context, the popularity bias and the song order for next-song recommendations. As a reference, all the results include the performance of a random model that assigns scores to next-song candidates uniformly at random, yielding random ranks.

4.6.1 Song context

We compare the four playlist continuation models described in Section 4.4. The popularity-based model predicts the next song disregarding the current and previous songs, i.e., it has no context. The song-based CF model predicts the next song on the basis of the current song but disregards the previous ones, i.e., it has a context of 1 song. The playlist-based CF and the RNN models predict the next song on the basis of the full playlist, i.e., they have full context.

Figure 4.2 reports the distribution of ranks attained by the actual next songs in the test playlists, given the predictions of the four playlist continuation models. The distributions of attained ranks are split by the position in the playlist for which the next-song prediction is made.[7] We consider only predictions up to position 8, which represent roughly the 90% of all the next-song predictions made in the AotM-2011 and the 8tracks datasets. From position 9 onwards the number of predictions quickly decreases and the results become less reliable.

The results in Figure 4.2 show that the popularity-based model and the song-based CF model do not improve their predictions as they progress through the playlists. This is the expected result because the popularity-based model has no context, and the song-based CF model has a context of 1 song. Their distributions of attained ranks remain stable with only small fluctuations easily explained by the fact that at each position the models deal with different songs. On the other hand, the playlist-based CF model and the RNN model are aware of the full song context. The results in Figure 4.2 show that the performance of the playlist-based CF model and the RNN model improves as they progress through the playlists, indicating that these models benefit from increasingly longer song contexts.

In terms of absolute model performance, the song-based CF model is the least competitive model, slightly better but not significantly different than the random reference model. The popularity-based model and the RNN model show the most competitive overall performances. The playlist-based CF model has difficulties when the song context

7 The position in the playlist for which the next-song prediction is made must not be confused with the song context length of the playlist continuation model. For example, making a next-song prediction for a song in position 5, the playlist-based CF model has a context of 4 songs, while the song-based CF still has a context of 1 song (Table 4.1).

is short, but it consistently improves as it gains more context, until eventually it outperforms the popularity-based model.

Summary of main observations:

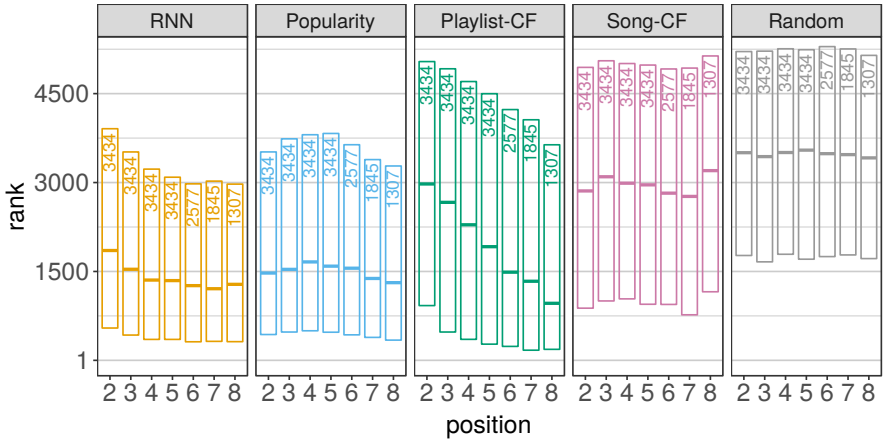
- *The playlist-based CF model and the RNN model, aware of the song context, improve their performance as they obtain increasingly longer contexts.*
- *Despite its simplicity, the popularity-based model performs comparably to the RNN model and, except for long contexts, better than the playlist-based CF model.*
- *The song-based CF model exhibits a poor performance.*

4.6.2 Popularity bias

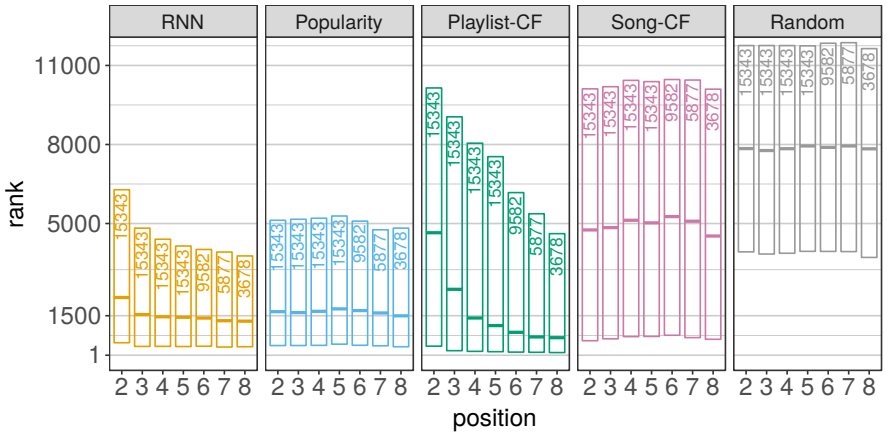
The previous results pose an apparent contradiction: the performance of the popularity-based model, unaware of the song context, is comparable to the performance of the RNN model and, overall, slightly superior to the performance of the playlist-based CF model, both aware of the full playlist context. Is it then important or not to exploit the song context? Furthermore, as discussed by Jannach and Ludewig [64], do marginal performance gains of the RNN model over the playlist-based CF model and the popularity-based model justify its higher computational complexity?

To shed light on these two questions we deem it important to analyze the possible impact of song-popularity effects, ubiquitous in the music consumption domain [21]. Within our off-line experiments, we identify the popularity of a song with its frequency in the datasets, i.e., with the number of playlists in which it occurs. Figure 4.3 and Table 4.2 show the song frequency distribution of the AotM-2011 and the 8tracks datasets. The AotM-2011 and the 8tracks datasets present a clear popularity bias, with a vast majority of songs occurring in few playlists and a few songs occurring in many playlists.

We consider again the performance of the four playlist continuation models in Figure 4.4 but this time distinguishing whether the actual next songs in the test playlists were popular or not. Precisely, we define the popularity of a song as its relative frequency in the training playlists, as given by Equation (4.1). The left panels in Figure 4.4 report the median rank attained by the actual next songs in the test playlists considering all the next-song predictions. The central panels in Figure 4.4 report the median rank attained by the actual next songs in the test playlists when the actual next songs belong to the 10% most popular songs in the datasets. The right panels in Figure 4.4 report the median rank attained by the actual next songs in the test playlists when the actual next songs belong to the 90% least popular songs in the datasets (which we refer to as the "long tail"). Figure 4.4 reports only the median rank to obtain a more compact graph that

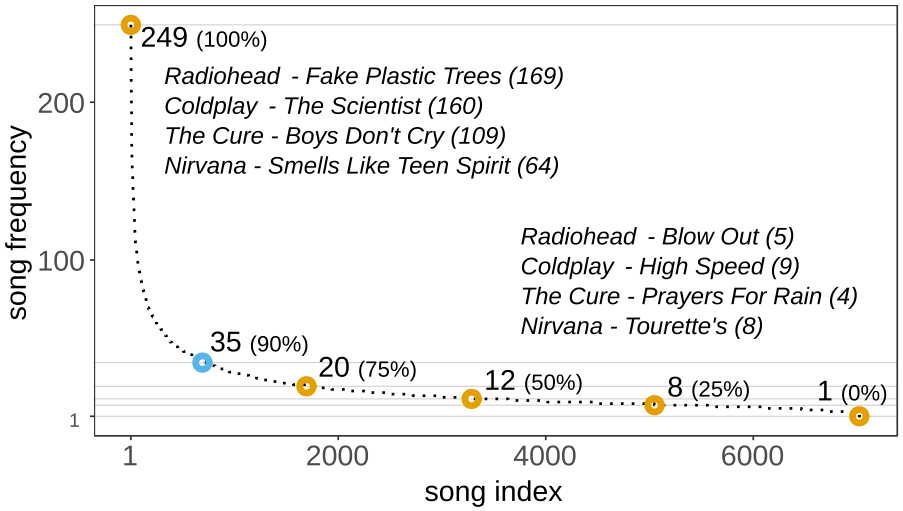


(a) AotM-2011 dataset

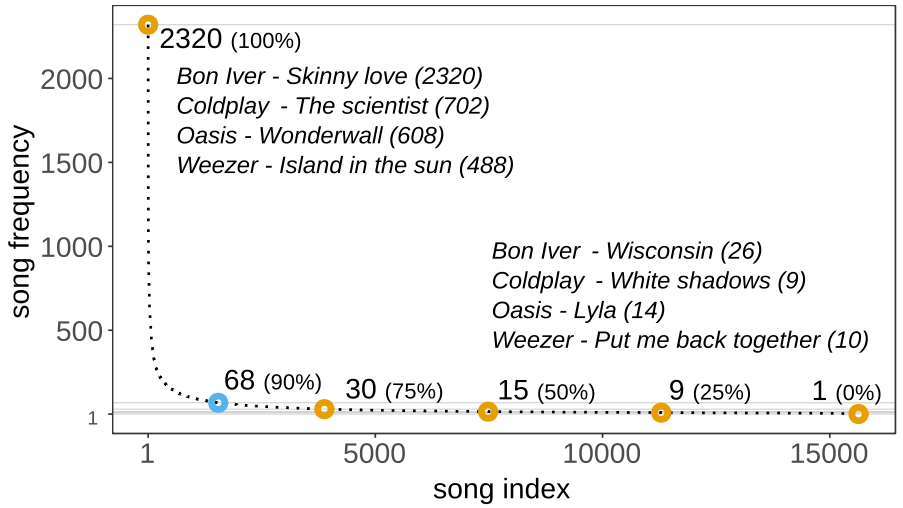


(b) 8tracks dataset

Figure 4.2: Song context length experiments. Distribution of ranks attained by the actual next songs in the test playlists (lower is better) for the AotM-2011 and the 8tracks datasets. Each panel corresponds to a playlist continuation model. The **x**-axis indicates the position in the playlist for which a prediction is made. The **y**-axis indicates the attained ranks and its scale relates to the number of songs in each dataset. The boxplots summarize the distribution of attained ranks by their first quartile, median and third quartile values. The number of rank values at every position is annotated.



(a) AotM-2011 dataset



(b) 8tracks dataset

Figure 4.3: Unique songs in the AotM-2011 and the 8tracks dataset, sorted by frequency, i.e., by the number of playlists in which they occur. The colored dots correspond to songs at percentile positions, with the song frequency and the percentile annotated (the latter in parentheses). Examples of frequent (popular) and infrequent (non-popular) songs in the datasets are provided, with the frequency annotated in parentheses.

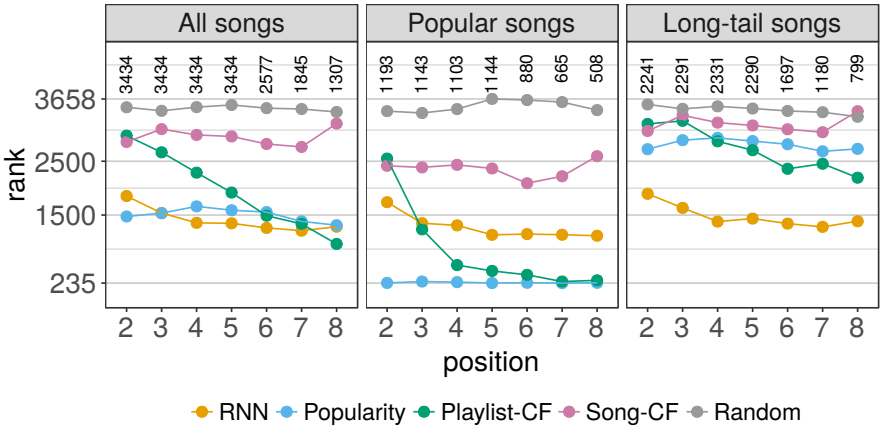
facilitates the comparison of the playlist continuation models over the song-popularity levels.

The results in Figure 4.4 show that the popularity-based model performs outstandingly well on the most popular songs, but it makes poor predictions for songs in the long tail. This is the natural consequence of the very design of the popularity-based model (Section 4.4.1). The playlist-based CF model performs reasonably well on the most popular songs and shows a quick improvement as it gains song context. Its performance on long-tail songs is poorer, but it shows a slight improvement as it gains song context, until given a context of at least 5 songs, it outperforms the popularity-based model. The good performance of the playlist-based CF model on popular songs is not surprising because the scoring Equation (4.3) favors songs occurring in many training playlists. However, the rather poor performance on long-tail songs is less expected, especially if we remember that our implementation of the playlist-based CF model considers all the training playlists as neighbors, which could help to counteract the large amount of non-popular songs in the AotM-2011 and the 8tracks datasets (Section 4.4.3). The song-based CF model also performs better on popular songs than on non-popular songs, especially in the 8tracks dataset, where the bias is stronger. The RNN model is reasonably competitive. Furthermore, in contrast to the other playlist continuation models, the performance of the RNN model is largely unaffected by the popularity of the actual next songs.

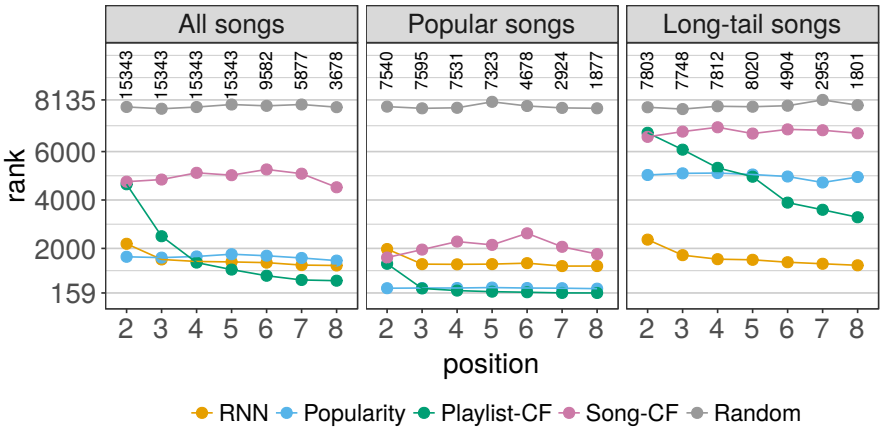
Focusing on the performance of the playlist continuation models on all next-song predictions (left panels in Figure 4.4), the popularity-based model seems comparable to the more sophisticated RNN model. Given enough song context, the playlist-based CF model also seems to compete with the RNN model. However, as we have just seen, the overall strong performance of the popularity-based model and the playlist-based CF model is the result of aggregating the accurate predictions made for a few popular songs (central panels in Figure 4.4) with the rather poor predictions made for a vast majority of non-popular songs (right panels in Figure 4.4). On the contrary, the performance of the RNN model is not affected by the song popularity. This observation must be taken into consideration to judge whether the higher computational complexity of the RNN model is justified, also considering the particular use case and target users of the recommender system. For example, the robustness of the RNN model to the popularity bias would be crucial to assist users interested in discovering non-popular music.

Summary of main observations:

- *The RNN model has a reasonably good performance, which is not affected by the popularity of the actual next songs.*



(a) AotM-2011 dataset



(b) 8tracks dataset

Figure 4.4: Popularity effects experiments. Median rank attained by the actual next songs in the test playlists (lower is better) for the AotM-2011 and the 8tracks datasets. Left: All songs are considered. Center: Only the 10% most popular songs in the dataset are considered. Right: Only the 90% least popular (long-tail) songs in the dataset are considered. The x-axis indicates the position in the playlist for which a prediction is made. The y-axis indicates the attained ranks and its scale relates to the number of songs in each dataset. The number of rank values at every position is annotated.

- *The popularity-based model, the song-based CF model and the playlist-based CF model exhibit a significant performance gap depending on the popularity of the actual next songs.*
- *Despite its overall poor performance on non-popular songs, the playlist-based CF model can exploit the song context to eventually outperform the popularity-based model.*

4.6.3 Song order

The RNN playlist continuation model is the most complex one among the four models considered and the only one aware of the song order. Furthermore, we have shown its good performance and robustness predicting playlist continuations. We now investigate the importance of considering the song order on next-song recommendations by comparing the performance of the RNN model when it is shown original playlists or playlists where the song order has been manipulated.

We devise three different song order manipulation experiments. Firstly, we train the RNN model on original playlists, but we evaluate it on shuffled playlists. This can be regarded as a *weak* check, because the RNN model can still potentially exploit the song order at training time. We refer to this setting as "shuffled test." Secondly, we train the RNN model on shuffled playlists and evaluate it on original playlists. This is a *strong* check, because we now make sure that the RNN model can not exploit the song order at training time. We refer to this setting as "shuffled training." Finally, we train and evaluate the RNN model on shuffled playlists. We refer to this experiment as "shuffled training and test." For the last two settings we re-tune the hyperparameters of the RNN model to make sure that the performance is not compromised as a consequence of modifying the training playlists. However, we keep the same architecture and hyperparameters because other configurations do not yield significantly better performance.

Figure 4.5 reports the distribution of ranks attained by the actual next songs in the test playlists given the predictions of the RNN model under each song order randomization experiment. The distributions of attained ranks are split by the position in the playlist for which the next-song prediction is made. As before, we consider only predictions up to position 8, which represent roughly the 90% of all the next-song predictions made in the AotM-2011 and the 8tracks datasets (Section 4.6.1). As a reference, we also include the performance of the RNN model trained and evaluated on original playlists. Surprisingly, the distribution of ranks is comparable for all the song order randomization experiments, regardless of whether the song order is maintained, broken at test time, or broken at training time. This result provides an indication that the song order may not be an essential feature for next-song recommendations. Alternatively, even

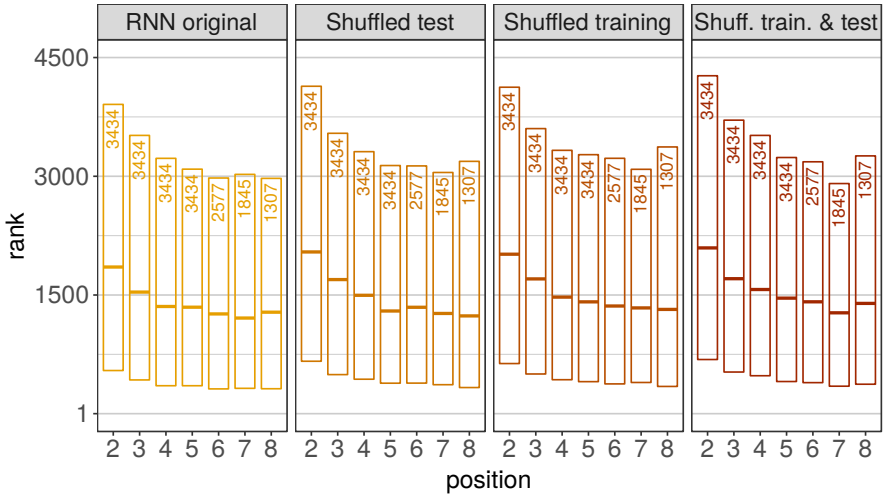
though RNN models are the state of the art in many sequential tasks, the result could be explained by the incapability of the considered RNN model to properly exploit the song order. Further investigation is required.

Analyzing closely Figure 4.5 we observe that, especially for the 8tracks dataset, the distribution of attained ranks when the song context is short (positions 2 and 3) is slightly worse on randomized playlists than on original playlists. However, even if the song order has been randomized, the performance of the RNN model improves as it is given more context. From position 5 onwards, the distributions of attained ranks on randomized playlists are close to those obtained on original playlists. Even though further experiments are required to fully understand this result, a possible explanation could be the following. Given a very short song context, the RNN model is uncertain about the playlist it is extending. Thus, it can not properly identify the relevant next-song candidates and resorts to "memorized" relations between songs, resulting in next-song predictions sensitive to the manipulation of the song order. On the other hand, given a longer song context, the RNN model is well informed about the playlist it is extending and it can successfully narrow down a relevant set of next-song candidates, regardless of the exact order in which the songs occurred in the playlist.

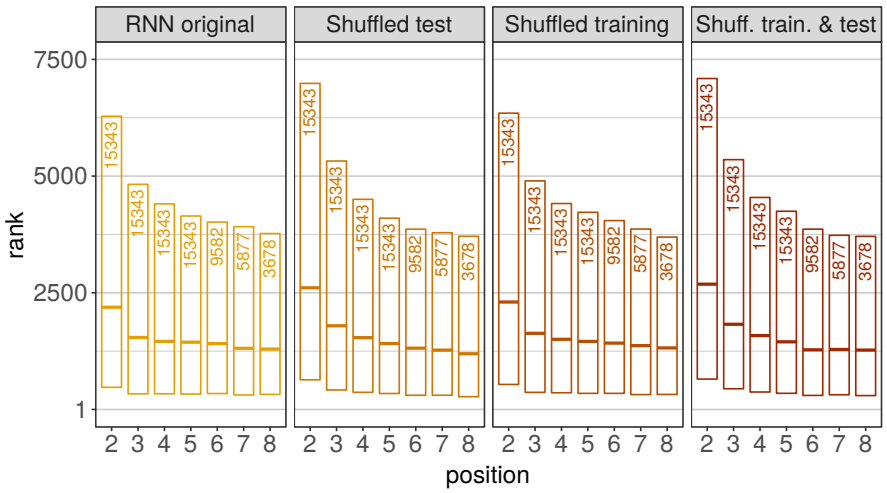
We further conduct a related set of experiments consisting in reversing the song order instead of randomizing it. A similar experiment was proposed by Chen et al. [25] to investigate the importance of the "directionality" in next-song recommendations. Chen et al. found only small performance differences evaluating the Latent Markov Embedding model on original and reversed playlists. We replicate the different settings from our previous experiments: we first train the RNN model on original playlists and evaluate it on reversed playlists. Then, we train the RNN model on reversed playlists and evaluate it on original playlists. Finally, we train and evaluate the RNN model on reversed playlists. Figure 4.6 reports the distribution of ranks attained by the actual next songs in the test playlists given the predictions of the RNN model under each reversed song order experiment. As expected, the results are not significantly different from those reported in Figure 4.5. That is, the distribution of ranks is comparable for all the reversed song order experiments.

Summary of main observations:

- *The RNN model achieves comparable performance on original playlists and on playlists where the song order has been reversed or randomized.*
- *The previous result suggests that either the song order is not a crucial feature for next-song recommendations, or the RNN model is unable to exploit the song order.*

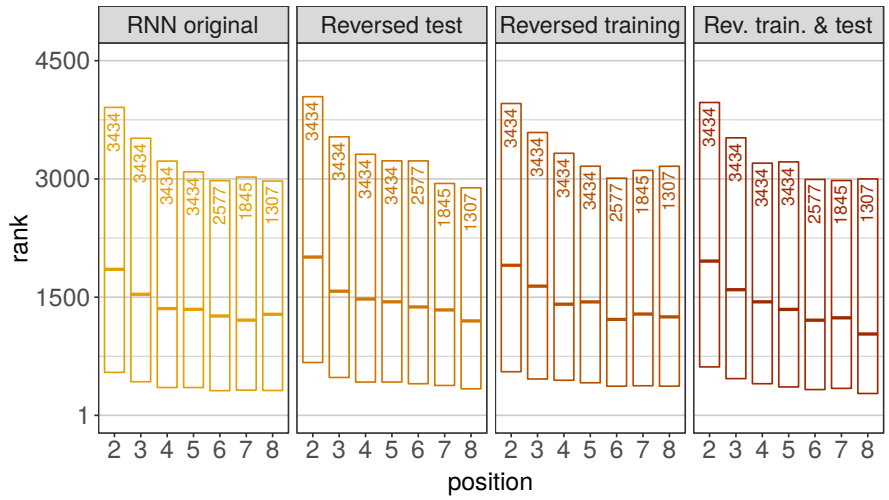


(a) AotM-2011 dataset

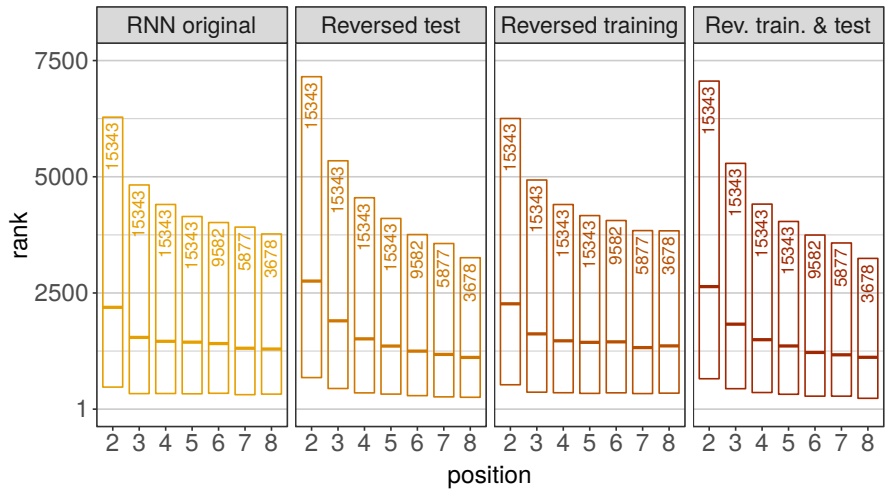


(b) 8tracks dataset

Figure 4.5: Randomized song order experiments. Distribution of ranks attained by the actual next songs in the test playlists (lower is better) for the AotM-2011 and the 8tracks datasets. The panels include the predictions of the RNN on the original playlists and on the different song order randomization experiments. The x-axis indicates the position in the playlist for which a prediction is made. The y-axis indicates the attained ranks and its scale relates to the number of songs in each dataset. The boxplots summarize the distribution of attained ranks by their first quartile, median and third quartile values. The number of rank values at every position is annotated.



(a) AotM-2011 dataset



(b) 8tracks dataset

Figure 4.6: Reversed song order experiments. Distribution of ranks attained by the actual next songs in the test playlists (lower is better) for the AotM-2011 and the 8tracks datasets. The panels include the predictions of the RNN on the original playlists and on the different reversed song order experiments. The x-axis indicates the position in the playlist for which a prediction is made. The y-axis indicates the attained ranks and its scale relates to the number of songs in each dataset. The boxplots summarize the distribution of attained ranks by their first quartile, median and third quartile values. The number of rank values at every position is annotated.

- *Even when the song order is reversed or randomized, the RNN model improves its performance as it gains song context.*

4.7 CONCLUSION

We have explicitly investigated the impact of the song context length, the song order and the song popularity in music playlists for the task of predicting next-song recommendations. We have conducted dedicated off-line experiments on two datasets of hand-curated music playlists comparing the following playlist continuation models: a popularity-based model, a song-based CF model, a playlist-based CF model, and an RNN model. These models are well-established and widely-used and exploit the song context and the song order to different extents. Our results indicate that the playlist-based CF model and the RNN model, which can consider the full song context, do benefit from increasingly longer song contexts. However, we observe that a longer song context does not necessarily translate into outperforming the simpler popularity-based model, which is unaware of the song context. This is explained by the popularity bias in the datasets, i.e., the coexistence of few, popular songs with many, non-popular songs. Failing to take into account the popularity bias masks significant performance differences: the popularity-based model, the song-based CF model and the playlist-based CF model exhibit significantly different performance depending on the popularity of the actual next songs in the test playlists. On the contrary, the more complex RNN model has a stable performance regardless of the song popularity. This effect must be taken into account in the design of playlist continuation models for specific use cases and target users. The RNN model is the only of the considered playlist continuation models aware of the song order. We have found that its performance on original, shuffled and reversed playlists is comparable, suggesting either that the song order is not crucial for next-song recommendations in such playlists, or that the RNN model, despite being one of the strongest statistical model for sequential tasks, is unable to fully exploit it. We have proposed an approach to assess the quality of the recommendations that observes the complete recommendation lists instead of focusing on the top K recommendations. Doing so provides a more complete view on the performance of the playlist continuation models.

4.A MODEL CONFIGURATIONS

4.A.1 Song popularity

This model computes the popularity of all the unique songs in the dataset. That is 7,032 songs for the AotM-2011 dataset and 15,649 songs for the 8tracks dataset.

4.A.2 Song-based collaborative filtering

This model computes pairwise similarities for all the unique songs in the dataset. That is 7,032 songs for the AotM-2011 dataset and 15,649 songs for the 8tracks dataset.

4.A.3 Playlist-based collaborative filtering

This model computes the similarity of each test playlist to all the training playlists in the dataset. That is, 13,744 playlists for the AotM-2011 dataset and 61,416 playlists for the 8tracks dataset. We also experimented using 100, 500 and 1000 training playlists but did not achieve better results.

4.A.4 Recurrent neural networks

We experiment with different loss functions, namely categorical cross-entropy, Bayesian Pairwise Ranking (BPR) [120] and TOP-1 [57]. The RNN is optimized using AdaGrad [38] with momentum and L2-regularization. We also experiment with dropout [138] in the recurrent layer, but none of the final configurations use it. We tune the number of units, the learning rate, the batch size, the amount of momentum, the L2-regularization weight and the dropout probability on a withheld validation set, by running 100 random search experiments [11] for each of the loss functions mentioned above. The final model configuration is chosen according to the validation recall at 100 (i.e., the proportion of times that the actual next song is included within the top 100 ranked candidates), which we consider a proxy of the model's ability to rank the actual next songs in top positions. The number of training epochs is chosen on the basis of the validation loss.

For the AotM-2011 dataset, the final model uses the TOP-1 loss and it has 200 hidden units. It is trained on mini-batches of 16 playlists, with a learning rate of 0.01, a momentum coefficient of 0.5 and an L2-regularization weight of 0.1. For the 8tracks dataset, the final model uses the TOP-1 loss and it has 200 hidden units. It is trained on mini-batches of 64 playlists, with a learning rate of 0.025, a momentum coefficient of 0.3 and an L2-regularization weight of 0.02. For both datasets, the hyperparemeters and architecture of the RNN models

trained on shuffled and reversed playlists were re-tuned, but since other configurations did not yield clearly better results, we decided to use the same settings for consistency.

5 HYBRID PLAYLIST SYSTEMS

SUMMARY

Music recommender systems have become a key technology to support the interaction of users with the increasingly larger music catalogs of on-line music streaming services, on-line music shops, and personal devices. An important task in music recommender systems is the automated continuation of music playlists, that enables the recommendation of music streams adapting to given (possibly short) listening sessions. Previous works have shown that applying collaborative filtering to collections of curated music playlists reveals underlying playlist–song co-occurrence patterns that are useful to predict playlist continuations. However, most music collections exhibit a pronounced long-tailed distribution. The majority of songs occur only in few playlists and, as a consequence, they are poorly represented by collaborative filtering. We introduce two feature-combination hybrid recommender systems that extend collaborative filtering by integrating the collaborative information encoded in curated music playlists with any type of song feature vector representation. We conduct off-line experiments to assess the performance of the proposed systems to recover withheld playlist continuations, and we compare them to competitive pure and hybrid collaborative filtering baselines. The results of the experiments indicate that the introduced feature-combination hybrid recommender systems can more accurately predict fitting playlist continuations as a result of their improved representation of songs occurring in few playlists.

REMARKS

This is the second chapter devoted to the task of music playlist continuation. Building on the conclusions drawn in the previous chapter, here I propose and thoroughly evaluate two hybrid recommender systems for music playlist continuation which (1) consider the full song context and (2) incorporate content information to improve the representation of very infrequent songs. This research initially yielded the following publications:

- Andreu Vall, Hamid Eghbal-zadeh, Matthias Dorfer, and Markus Schedl. “Timbral and semantic features for music playlists.” In:

Machine Learning for Music Discovery Workshop at ICML. New York, NY, USA, 2016,

- 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: *Proc. DLRS Workshop at RecSys*. Como, Italy, 2017, pp. 46–54,

- Andreu Vall, Matthias Dorfer, Markus Schedl, and Gerhard Widmer. "A hybrid approach to music playlist continuation based on playlist-song membership." In: *Proc. SAC*. Pau, France, 2018, pp. 1374–1382.

The current chapter extends these works and has been accepted for publication:

- Andreu Vall, Matthias Dorfer, Hamid Eghbal-zadeh, Markus Schedl, Keki Burjorjee, and Gerhard Widmer. "Feature-combination hybrid recommender systems for automated music playlist continuation." In: *User Modeling and User-Adapted Interaction* (2019, in press).

I conceived and conducted the research presented in this chapter. Nevertheless, I maintain the use of the first-person plural ("we") and acknowledge the collaboration of the co-authors. Matthias Dorfer contributed to the design and implementation of the second of the proposed recommender systems and contributed the extraction of audio-based song features using convolutional neural networks. Hamid Eghbal-zadeh contributed to the extraction of audio-based song features using the "i-vectors" framework. Keki Burjorjee provided feedback and expertise on matrix factorization models. Markus Schedl and Gerhard Widmer provided valuable feedback.

Implementations of the proposed hybrid recommender systems are made available:

- Andreu Vall and Matthias Dorfer. *Hybrid recommender systems for music playlist continuation*. Original date: 2017-04-20. URL: `https://github.com/andreuvall/HybridPlaylistContinuation` (visited on 12/12/2018).

5.1 INTRODUCTION

Music recommender systems have become an important component of music platforms to assist users to navigate increasingly larger music collections. Recommendable items in the music domain may correspond to different entities such as songs, albums, or artists [24,

121, Chapter 13], and music streaming services even organize music in more abstract categories, like genre or activity.

As a consequence of the relatively short time needed to listen to a song (compared to watching a movie or reading a book) a user session in an on-line music streaming service typically involves listening to, not one, but several songs. Thus, modeling and understanding music playlists is a central research goal in music recommender systems. As in other item domains, music recommender systems often provide personalized lists of suggestions based on the users' general music preferences. This approach may work to recommend music entities such as albums, artists, or ready-made listening sessions (like curated playlists or charts) because it can be useful to provide the users with a wide choice range. However, recommendations based on the users' general music preferences may be too broad for the task of automated music playlist continuation, where it is crucial to recommend individual songs that specifically adapt to the most-recent songs played.

A common approach to explicitly address the automated continuation of music playlists consists in applying Collaborative Filtering (CF) to curated music playlists, revealing specialized playlist–song co-occurrence patterns [2, 17]. While this approach works fairly well, it has an important limitation: the performance of any CF system depends on the availability of sufficiently dense training data [1]. In particular, songs occurring in few playlists can not be properly modeled by CF because they are hardly related to other playlists and songs. Music collections generally exhibit a bias towards few, popular songs [21]. In the case of collections of curated music playlists, this translates into a vast majority of songs occurring only in very few playlists. This majority of infrequent songs is poorly represented by CF.

To overcome this limitation, we observe that songs occurring rarely in the context of curated playlists are not necessarily completely unknown to us. We can often gather rich song-level side information from, e.g., the audio signal, text descriptions from social-tagging platforms, or even listening logs from music streaming services. Such additional song descriptions can be leveraged to make CF robust to infrequent songs by means of hybridization [1, 19].

We introduce two feature-combination hybrid recommender systems that integrate curated music playlists with any type of song feature vector derived from song descriptions. The curated music playlists provide playlist–song co-occurrence patterns as in CF approaches. The song features make the proposed systems robust to data scarcity problems. In contrast to previous hybrid playlist continuation approaches, the proposed systems are *feature-combination* hybrids [19], having the advantage that the collaborative information and the song features are implicitly fused into standalone enhanced recommender

systems. The proposed systems can be used to play and sequentially extend music streams, resulting in a *lean-back* listening experience similar to traditional radio broadcasting, or to assist users to find fitting songs to extend their own music playlists, stimulating their engagement.

5.1.1 Contributions of the chapter

- We provide a unified view of music playlist continuation as a matrix completion and expansion problem, encompassing
  - pure CF systems, solely exploiting curated playlists,
  - hybrid systems integrating curated playlists and song feature vectors.

- We introduce two feature-combination hybrid recommender systems
  - readily applicable to automated music playlist continuation,
  - able to exploit any type of song feature vectors.

- Still, the proposed systems are domain-agnostic. They can generally leverage
  - collaborative implicit feedback data from any domain,
  - item feature vectors from any domain and modality.

- A thorough off-line evaluation comparing to pure and hybrid state-of-the-art CF baselines shows that, having access to comparable data, the proposed systems
  - compete to CF when sufficient training data is available,
  - outperform CF when training data is scarce,
  - compete to, or outperform the hybrid baseline.

- The proposed systems further improve their performance by considering richer song feature vectors, e.g., concatenating features from different modalities.

- The evaluation also provides a complete comparison of
  - a widely-used matrix factorization CF system [60],
  - its audio-based hybrid extension [109],
  - a popular playlist-neighbors CF system [17].

- The appendices extend the evaluation of the proposed systems with a detailed analysis of the contribution of each type of song feature vector, showing
  - the standalone performance of each type of song feature vector,
  - the incremental gains of stepwise combinations of song feature vectors.

5.1.2 Scope of the chapter

Compiling a music playlist is a complex task. According to interviews with practitioners and postings to a playlist-sharing website, Cunningham, Bainbridge, and Falconer [31] found that the playlist curation process is influenced by factors such as mood, theme, or purpose. They also observed a lack of agreement on curation rules, except for loose and subjective guidelines. Krause and North [77] studied music listening in situations. Among other conclusions, they found that participants of their study, when asked to compile playlists for specific situations, selected music seeking to comply with perceived social norms defining what music ought to be present in each situation.

The scope of our work is restricted to machine learning approaches to music recommender systems. We focus on the exploitation of data describing playlists and the songs therein, in order to identify patterns useful to recommend playlist continuations. We acknowledge the complexity of the playlist curation process, and we are aware of the possible limitations of a pure machine-learning perspective.

5.1.3 Organization of the chapter

The remainder of the chapter is organized as follows. Section 5.2 reviews previous works on music playlist continuation. Section 5.3 formulates music playlist continuation as a matrix completion and expansion problem. Sections 5.4 and 5.5 describe the proposed systems and the baselines for music playlist continuation, respectively. The evaluation methodology is presented in Section 5.6. Section 5.7 describes the datasets of curated playlists and song features used in our experiments. Section 5.8 elaborates on the results. Finally, conclusions are drawn in Section 5.9. Additional details of each playlist continuation system, additional song feature types, and additional results are provided in Appendices 5.A, 5.B and 5.C, respectively.

5.2 RELATED WORK

Content-based recommender systems for automated music playlist continuation generally compute pairwise song similarities on the basis of previously extracted song features and use these similarities to enforce content-wise smooth transitions. Such systems have typically relied on audio-based song features [47, 93, 116], possibly combined with features extracted from social tags [99] or web-based data [74]. While this approach is expected to yield coherent playlists, Lee, Bare, and Meek [80] actually found that recommending music with stronger audio similarity does not necessarily translate to higher user satisfaction. This limitation relates to the so-called *semantic gap* in music information retrieval, that is, the distance between the raw audio signal of a song and a listener's perception of the song [22].

Collaborative Filtering (CF) has been proven successful to reveal underlying structure from user-item interactions [1, 121]. In particular, CF has been applied to music playlist continuation by considering collections of hand-curated playlists and regarding each playlist as a user's listening history on the basis of which songs should be recommended. Previous research has mostly focused on playlist-neighbors CF systems [17, 53, 63], but Aizenberg, Koren, and Somekh [2] also presented a latent-factor CF model tailored to mine Internet radio stations, accounting for song, artist, time of the day, and song adjacency. An important limitation of most latent-factor and playlist-neighbors CF systems is that they need to profile the playlists at training time in order to extend them, by computing their latent factors or finding their nearest neighbors. As a consequence, such systems can not extend playlists unseen at training time. To circumvent this issue, Aizenberg, Koren, and Somekh [2] replaced the latent factors of unseen playlists by the latent factors of their songs, and Jannach and Ludewig [64] showed how to efficiently implement a playlist-neighbors CF system able to extend unseen playlists in reasonable time, even for large datasets. Song-neighbors CF systems have also been investigated [154, 158], and Bonnin and Jannach [17] proposed a successful variation consisting in computing similarities between artists instead of between songs, even when the ultimate recommendations were at the song level. Their system also incorporated song popularity. A common limitation of all pure CF systems is that they are only aware of the songs occurring in training playlists. Thus, songs that never occurred in training playlists, to which we refer as "out-of-set" songs, can not be recommended in an informed manner. Furthermore, songs that do occur in training playlists, but seldom, are not properly modeled by CF because they lack connections to other playlists and songs.

Other collaborative systems (i.e., systems based on the exploitation of playlist–song interactions) have been presented. Zheleva et al. [168] proposed to adapt Latent Dirichlet Allocation (LDA) [14] to modeling

listening sessions. They found that a variation of LDA that specifically considers the sessions provided better recommendations than plain LDA. Chen et al. [25] presented the Latent Markov Embedding, a model that exploits radio playlists to learn an embedding of songs into a Euclidean space such that the distance between embedded songs relates to their transition probability in the training playlists. Both systems can extend playlists unseen at training time, but can only make informed recommendations for songs occurring in training playlists.

Hybrid systems combining collaborative and content information are a common approach to mitigate the difficulties of CF to represent infrequent songs. Hariri, Mobasher, and Burke [53] represented the songs in hand-curated playlists by topic models derived from social tags and then mined frequent sequential patterns at the topic level. The recommendations predicted by a playlist-neighbors CF system were re-ranked according to the next topics predicted. The approach proposed by Jannach, Lerche, and Kamehkhosh [63] pre-selected suitable next songs for a given playlist using a weighted combination of the scores yielded by a playlist-neighbors CF system and a content-based system. The candidate songs were then re-ranked to match some characteristic of the playlist being extended. In both cases the hybridization followed from the combination of independently obtained scores, by means of weighting heuristics or re-ranking. For songs occurring in few training playlists, the recommendations predicted by CF could be boosted with content information. However, the recommendations for out-of-set songs would solely rely on the content-based component of the hybrid systems.

McFee and Lanckriet [100] proposed a hybrid system integrating collaborative and content information more closely. The system was based on a weighted song hypergraph, that is, a song graph where edges can join multiple songs, and weights define the similarity between the (possibly many) songs connected by an edge. The edges were defined by assigning the songs in hand-curated playlists to possibly overlapping song sets, which had been previously obtained through clustering of extracted multimodal song features. The weights were found in a second step as the best possible fit given a collection of training playlists and the hypergraph edges. This system could better deal with out-of-set songs, as it would only need to assign them to appropriate edges. Not strictly applied to music playlist continuation but to music understanding and recommendation in general, Oord, Dieleman, and Schrauwen [109] introduced the use of convolutional neural networks to estimate song factors from a latent-factor model, given the log-compressed mel-spectrogram of the audio signal of songs. Such networks can be essentially regarded as feature extraction tools, but further combined with latent-feature models they enable the informed recommendation of infrequent or out-of-set songs. This

approach was further combined with semantic features derived from artist biographies by Oramas et al. [110].

Our own previous works on music playlist continuation focused on two main research lines. On the one hand we studied the importance of three main playlist characteristics (length, song order, and popularity of the songs included) in music playlist continuation systems [154, 156, 158]. On the other hand, more related to the current work, we analyzed the extent to which multimodal features can capture playlist–song relationships, and we designed two feature-combination hybrid recommender systems for music playlist continuation [152, 153, 155]. In the current work we consolidate the second line of research. We present the two feature-combination hybrid systems in full detail. We conduct an extensive evaluation, comparing the proposed systems to four competitive playlist continuation baselines, and incorporating uncertainty estimation by means of bootstrap confidence intervals. We analyze additional audio-based features extracted applying convolutional neural networks on song spectrograms. Finally, the evaluation further provides new, insightful comparisons between well-established pure and hybrid CF systems, namely the matrix factorization model proposed by Hu, Koren, and Volinsky [60], its hybrid extension proposed by Oord, Dieleman, and Schrauwen [109], and the playlist-neighbors CF system [17, 53, 63].

5.3 PROBLEM FORMULATION

Let P be a collection of music playlists. Let S be the universe of songs available, including at least the set S_P of songs occurring in the playlists of the collection P, but possibly more (i.e., $S \supseteq S_P$). A playlist $p \in P$ is regarded as a set of songs, where the song order is ignored.[1] Playlists may have different lengths.

Since playlists are seen as song sets, any playlist p is a subset of the universe of songs S. Thus, the set difference $S \setminus p$ represents the songs that do not belong to the playlist p. A song s can be regarded as playlist of one song, i.e., as the singleton set $\{s\}$. Given a song s in a playlist p, the set difference $p \setminus \{s\}$ removes the song s from the playlist p. The length of a playlist p is denoted by its cardinality $|p|$.

We refer to any playlist $p \in P$ as an "in-set" playlist, and to any song $s \in S_P$ as an "in-set" song. In contrast, we refer to any playlist $p \notin P$ as an "out-of-set" playlist, and to any song $s \notin S_P$ as an "out-of-set" song.

1 Even though the process of listening to a playlist is inherently sequential, we found in Chapter 4 that considering the song order in curated music playlists is actually not crucial to extend such playlists. While more research is required to fully understand the impact of the song order in music playlists, we feel confident that disregarding the song order does not harm the contribution of the current work.

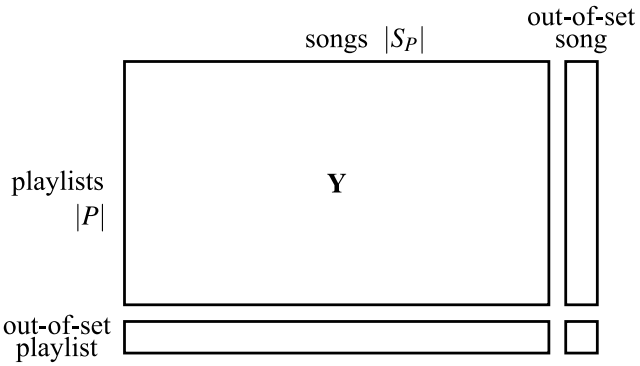


Figure 5.1: Playlist continuation as a matrix completion and expansion problem. The matrix $\mathbf{Y}$ encodes the playlist collection P. CF systems discover in-set potential positive playlist–song interactions by "completing" the matrix $\mathbf{Y}$. Hybrid systems can further "expand" the matrix $\mathbf{Y}$ towards out-of-set songs by incorporating external song descriptions. Systems not specializing in the playlists of P can expand the matrix towards out-of-set playlists, possibly at the cost of slightly lower performance.

5.3.1 Playlist continuation as matrix completion

While a single playlist typically reflects individual preferences, a collection of playlists constitutes a source of collaborative implicit feedback [60, 112] encoding rich playlist–song co-occurrence patterns. Similar to other recommendation tasks, music playlist continuation can be regarded as a matrix completion problem. The playlist collection P is arranged into a binary matrix $\mathbf{Y} \in \{0,1\}^{|P| \times |S_P|}$ of playlist–song interactions, with as many rows as playlists and as many columns as unique songs in the playlists (Figure 5.1). The interaction between a playlist p and a song s indicates whether the song occurs ($y_{p,s} = 1$) or not ($y_{p,s} = 0$) in the playlist. The matrix $\mathbf{Y}$ is typically very sparse and thus it can be stored efficiently by keeping only the positive interactions (e.g., the playlist collections introduced in Section 5.7 have both a density rate of 0.08%).

"Completing" the matrix $\mathbf{Y}$ generally refers to discovering new potential positive playlist–song interactions. Songs identified as potential positive interactions to a given playlist are then recommended as candidates to extend the playlist. That is the approach followed by CF systems, both neighborhood-based [17, 154, 158] and model-based [2].

We evaluate two CF baselines, one model-based and one neighbors-based. For the model-based system, we adapt the matrix factorization model for implicit feedback datasets by Hu, Koren, and Volinsky [60] for the task of music playlist continuation (Section 5.5.1). For the neighbors-based system, we modify the playlist-neighbors CF system [17, 53, 63] to adapt to the challenging sparsity of the considered playlist collections (Section 5.5.3).

5.3.2 Playlist continuation as matrix expansion

The matrix completion framework is limited to playlists and songs within the matrix. However, common use cases may require extending not-yet-seen playlists, or considering candidate songs that do not occur in any of the playlists of the matrix.

Out-of-set songs

CF systems rely solely on playlist–song co-occurrence patterns. Therefore, they are unable to recommend out-of-set songs as candidates to extend playlists, precisely because out-of-set songs do not co-occur with the playlists in the collection. Hybrid extensions to CF overcome this limitation by incorporating external song descriptions seeking to compensate for the lack of playlist–song co-occurrences. Hybrid systems can not only enable the recommendation of out-of-set songs (Figure 5.1) but also strengthen the representation of in-set but infrequent songs.

The feature-combination hybrid recommender systems proposed in this work handle out-of-set and in-set but infrequent songs by fusing any type of song feature vectors with collaborative patterns derived from hand-curated music playlists (Sections 5.4.1 and 5.4.2). We also evaluate the hybrid CF system proposed by Oord, Dieleman, and Schrauwen [109], which predicts song latent factors from the audio signal and passes them to a matrix factorization model. We extend this latter approach by additionally considering song latent factors derived from independent listening logs (Section 5.5.2).

Out-of-set playlists

Model-based CF systems relying on matrix factorization are not generally able to extend playlists unseen at training time. However, we see that this limitation can be overcome for the matrix factorization model considered in this work [60], and we show how to predict continuations for out-of-set playlists (Figure 5.1) provided that latent song factors are available (Section 5.5.1).

Neighbors-based CF systems can generally extend out-of-set playlists. Still, playlist-neighbors CF systems require a careful implementation to efficiently compute the similarity between out-of-set playlists and large training playlist collections [17, Appendix A.1]. This computation can be accelerated by sampling a subset of the training playlists [64]. The moderate size of the playlist collections considered in this work, however, does not make it necessary to apply such sampling (Section 5.5.3).

The first of the hybrid systems proposed in this work specializes towards the collection of training playlists (Section 5.4.1). It achieves a very competitive performance, but it is not readily able to extend

out-of-set playlists. The second hybrid system is designed to generally model whether any playlist and any song fit together (Section 5.4.2). It achieves slightly lower performance but it can handle out-of-set playlists.

5.3.3 Recommending playlist continuations

A playlist continuation system has to be able to predict a score quantifying the fitness between a playlist p and a candidate song s. This score may be interpreted as a probability (e.g., in the proposed systems) or as a similarity measure (e.g., in neighbors-based CF systems). After assessing the fitness between a playlist and multiple song candidates, we select the most suitable song recommendations to extend the playlist.

5.4 PROPOSED SYSTEMS

We introduce two hybrid feature-combination recommender systems. The feature-combination hybridization scheme integrates collaborative and content information treating the collaborative information as an additional feature associated to each playlist–song pair [19]. The hybridization results in an enhanced, standalone system, informed about both types of information. This is in contrast to other hybridization schemes that simply combine the predictions of independent systems.

5.4.1 Profiles-based playlist continuation ("Profiles")

This system specializes towards a playlist collection by means of a song-to-playlist classifier. As a consequence of this specialization, Profiles achieves very competitive performance, but it is not readily able to extend out-of-set playlists. This system should be used to recommend songs to stable user playlists. If new playlists needed to be considered, the song-to-playlist classifier could be extended using incremental training techniques [84]. Profiles can deal with out-of-set songs.

Model definition

A song s is represented by a feature vector $\mathbf{x}_s \in \mathbb{R}^D$. We are interested in the probability of song s fitting each of the playlists of a collection P.

The system is based on a song-to-playlist classifier implemented by a neural network $\mathbf{c}\colon \mathbb{R}^D \to \mathbb{R}^{|P|}$. The network takes the song feature $\mathbf{x}_s$ as input. The output $\mathbf{c}(\mathbf{x}_s) \in \mathbb{R}^{|P|}$ is pointwise passed through logistic

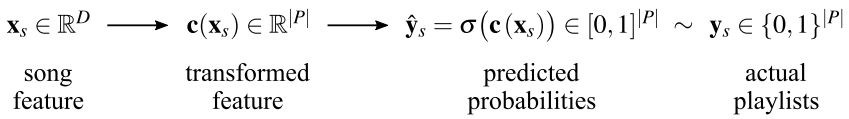


Figure 5.2: Sketch of the Profiles system. A song feature vector is taken as input and processed through the network to decide the playlists of P that the song fits. The model is trained on labeled song-to-playlist examples.

activation functions,[2] yielding a vector $\hat{\mathbf{y}}_s = \sigma(\mathbf{c}(\mathbf{x}_s)) \in [0,1]^{|P|}$ that indicates the predicted probability of song s fitting each of the playlists in the collection P (Figure 5.2).

The song-to-playlist classifier depends on a set of learnable weights θ_c (omitted so far for simplicity). The weights are adjusted on the basis of training examples $\{\mathbf{x}_s, \mathbf{y}_s\}$ (Section 5.4.1) by comparing the model's predicted probabilities to the actual labels $\mathbf{y}_s \in \{0,1\}^{|P|}$. Precisely, the weights θ_c are estimated to minimize the following binary cross-entropy cost function

$$\mathcal{L}_{\text{Profiles}}(\theta_c \mid \{\mathbf{x}_s, \mathbf{y}_s\}) = \\ -\sum_{p,s} y_{p,s} \log(\hat{y}_{p,s}) + (1 - y_{p,s}) \log(1 - \hat{y}_{p,s}). \quad (5.1)$$

The terms $y_{p,s}$ and $\hat{y}_{p,s}$ denote the components of $\mathbf{y}_s$ and $\hat{\mathbf{y}}_s$ corresponding to playlist p, respectively. The dimensionality of the set of parameters θ_c depends on the network architecture. The summation is done over all the possible playlist–song pairs, both occurring ($y_{p,s} = 1$) and non-occurring ($y_{p,s} = 0$) in the training playlists. We experimented with different weighting schemes for occurring and non-occurring pairs, as suggested by Hu, Koren, and Volinsky [60] or Pan et al. [112], but none yielded superior performance than using equal weights.

Song-to-playlist training examples

S_P is the set of unique songs in the playlists of P. For each song $s \in S_P$, $\mathbf{y}_s \in \{0,1\}^{|P|}$ is the column of the playlist–song interactions matrix $\mathbf{Y}$ corresponding to song s, which indicates the playlists of P to which the song s belongs. The training set consists of all the pairs of song features and playlist-indicator binary vectors $\{\mathbf{x}_s, \mathbf{y}_s\}_{s \in S_P}$.

5.4.2 Membership-based playlist continuation ("Membership")

This system generally models playlist–song membership relationships, that is, whether a given playlist and a given song fit together. This ap-

2 The song-to-playlist classifier makes as many independent decisions as playlists in the collection P. We also experimented with a softmax activation function yielding a probability distribution over playlists, but using sigmoids provided better results according to the followed evaluation methodology (Section 5.6).

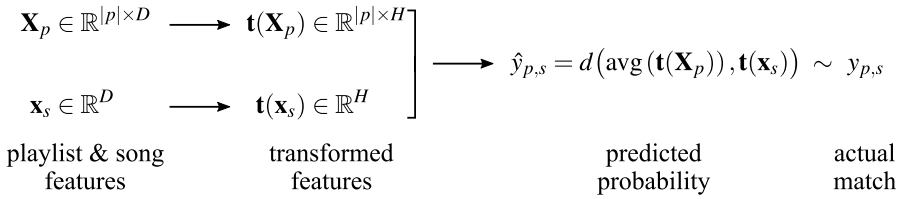


Figure 5.3: Sketch of the Membership system. Given any playlist–song pair, its feature matrix and vector are transformed into hidden representations that are then used to decide if the playlist–song pair fits together. The model is trained on labeled playlist–song pairs derived from Algorithm 1.

proach is related to the Profiles system, but here we seek to discourage the specialization towards specific playlists by generally representing any playlist by the feature vectors of the songs that it contains. In this way, Membership can deal with out-of-set playlists and out-of-set songs.

Model definition

A playlist p is represented by a feature matrix $\mathbf{X}_p \in \mathbb{R}^{|p| \times D}$ that contains, in each row, the feature vector of each song in the playlist. A song s is represented by a feature vector $\mathbf{x}_s \in \mathbb{R}^D$. A playlist–song pair (p, s) is then represented by the features $(\mathbf{X}_p, \mathbf{x}_s)$.

The system is based on a deep neural network with a "feature-transformation" component $\mathbf{t}\colon \mathbb{R}^D \to \mathbb{R}^H$ and a "match-discrimination" component $d\colon \mathbb{R}^{2H} \to [0, 1]$. The playlist feature matrix $\mathbf{X}_p$ is transformed song-wise (i.e., row-wise) into a hidden matrix representation $\mathbf{t}(\mathbf{X}_p) \in \mathbb{R}^{|p| \times H}$ (where we slightly abuse notation for $\mathbf{t}$). This matrix is averaged over songs yielding a summarized playlist feature vector $\text{avg}(\mathbf{t}(\mathbf{X}_p)) \in \mathbb{R}^H$. The song feature vector $\mathbf{x}_s$ is also transformed into a hidden representation $\mathbf{t}(\mathbf{x}_s) \in \mathbb{R}^H$. Both hidden representations are passed through the match-discrimination component that predicts the probability of the playlist–song pair fitting together, $\hat{y}_{p,s} = d\big(\text{avg}(\mathbf{t}(\mathbf{X}_p)), \mathbf{t}(\mathbf{x}_s)\big) \in [0, 1]$ (Figure 5.3).

The transformation and match-discrimination components depend on sets of learnable weights θ_t and θ_d, respectively (omitted so far for simplicity). These are adjusted on the basis of training examples $\{(\mathbf{X}_p, \mathbf{x}_s), y_{p,s}\}$ (Section 5.4.2) by comparing the model's predicted probability for a pair (p, s) to the actual label $y_{p,s} \in \{0, 1\}$. Precisely, the sets of weights θ_t, θ_d are estimated to minimize the following binary cross-entropy cost function

$$\mathcal{L}_{\text{Membership}}\Big(\theta_t, \theta_d \;\Big|\; \{(\mathbf{X}_p, \mathbf{x}_s), y_{p,s}\}\Big) = \\ -\sum_{p,s} y_{p,s} \log\left(\hat{y}_{p,s}\right) + \left(1 - y_{p,s}\right) \log\left(1 - \hat{y}_{p,s}\right). \tag{5.2}$$

The dimensionalities of the sets of parameters θ_t and θ_d depend on the network architecture.

Playlist–song training examples

We assume that any playlist $p \in P$ implicitly defines matches to each of its own songs. That is, each song $s \in p$ matches the shortened playlist $p_s = p \setminus \{s\}$. We further assume that any song not occurring in the playlist p is a "mismatch" to the shortened playlist p_s.[3] Thus, we can obtain a mismatch by randomly drawing a song from $S \setminus p$.

Following this procedure, Algorithm 1 details how to derive a training set with as many matching as mismatching playlist–song pairs given a playlist collection P and a universe of available songs S.

Algorithm 1 Derive playlist–song matches and mismatches.

Input:
- P ▷ playlist collection
- S ▷ universe of songs

Output:
- matches ▷ list of playlist–song matches
- mismatches ▷ list of playlist–song mismatches

```
matches = []                                   ▷ initialize empty lists
mismatches = []
for each p ∈ P do
    for each s ∈ p do
        p_s = p \ {s}                          ▷ remove s from p
        s_+ = s                                ▷ s is a match to p_s
        s_− = sample(S \ p)                    ▷ draw a mismatch to p_s
        matches.append((X_{p_s}, x_{s_+}), 1)  ▷ store training examples
        mismatches.append((X_{p_s}, x_{s_−}), 0)
    end for
end for
return matches, mismatches
```

Sampling strategy

The Membership system can be utilized as we have described so far. However, we find that applying the following sampling strategy before we derive the training playlist–song pairs and at recommendation time is necessary to obtain competitive results.

We set a fix playlist length n given by the length of the shortest playlist in a collection P. Given a playlist $p \in P$, we derive all the sub-playlists p' that result from drawing n songs from p without replacement. However, the number of possible draws can be large. To keep the approach computationally tractable, if the number of possible

3 In the context of implicit feedback, the term "no-match" may be preferable to "mismatch" because missing feedback does not necessarily reflect negative feedback. However, we keep the latter for simplicity.

draws is larger than $|p|$, we select only $|p|$ sub-playlists by randomly drawing n songs from p without replacement $|p|$ times.[4]

We apply this procedure to each playlist of P, thus obtaining a modified playlist collection P' with many more, but shorter fix-length playlists. Then, we apply Algorithm 1 to the modified collection P' to derive training playlist–song pairs.

Once Membership is trained, we also apply the sampling strategy to predict the match probability of an unseen playlist–song pair (p, s). We derive no more than $|p|$ sub-playlists out of p as described above. We let Membership predict the match probability of (p', s) for each derived sub-playlist p'. Then we average the probabilities.

5.5 BASELINE SYSTEMS

5.5.1 Matrix factorization ("MF")

This is a purely collaborative system based on the weighted matrix factorization model proposed by Hu, Koren, and Volinsky [60]. As any pure CF system, it is unable to recommend out-of-set songs. In principle it is also unable to extend out-of-set playlists, but we see how to overcome this limitation with a fast one-step factorization update (Section 5.5.1).

Model definition

We factorize the matrix of playlist–song interactions $\mathbf{Y} \in \{0,1\}^{|P|\times|S_P|}$ into two low-rank matrices $\mathbf{u} \in \mathbb{R}^{|P|\times D}$, $\mathbf{v} \in \mathbb{R}^{|S_P|\times D}$ of playlist and song latent factors, respectively, where D is the depth of the factorization and the product $\hat{\mathbf{Y}} = \mathbf{u} \cdot \mathbf{v}^T$ approximately reconstructs the original matrix $\mathbf{Y}$. Precisely, the latent factors are estimated to minimize the following weighted least squares cost function

$$\mathcal{L}_{MF}(\mathbf{u}, \mathbf{v} \mid \mathbf{Y}) = \sum_{p,s} w_{p,s} \left(y_{p,s} - \mathbf{u}_p \cdot \mathbf{v}_s^T\right)^2, \tag{5.3}$$

where $w_{p,s}$ is the weight assigned to the playlist–song pair (p, s). Following Hu, Koren, and Volinsky [60], we define the weights by $w_{p,s} = 1 + \alpha y_{p,s}$, where α is a parameter adjusted on a validation set. However, since the matrix $\mathbf{Y}$ is binary, the weighting scheme is reduced to

$$w_{p,s} = \begin{cases} w_1 & \text{if } y_{p,s} = 1 \\ 1 & \text{if } y_{p,s} = 0, \end{cases}$$

and the weight w_1 is adjusted on a validation set.

4 The number of possible sub-playlists is $\binom{|p|}{n}$. For example, we could sample 2,002 sub-playlists of length 5 out of a playlist of length 14. In this case, we would randomly draw 14 sub-playlists of length 5.

Minimization via Alternating Least Squares

The cost function (5.3) is minimized via Alternating Least Squares (ALS), an iterative optimization procedure consisting in subsequently keeping one of the factor matrices fixed while the other is updated. The initial factor matrices $\mathbf{u}^0$, $\mathbf{v}^0$ are set randomly. At iteration k, the song factors $\mathbf{v}^k$ are obtained by minimizing an approximation of the original cost function where the playlist factors have been fixed to $\mathbf{u}^k$:

$$\widetilde{\mathcal{L}}_{\mathrm{MF}}\left(\mathbf{v} \mid \mathbf{u} = \mathbf{u}^k, \mathbf{Y}\right) = \sum_{p,s} w_{p,s} \left(y_{p,s} - \mathbf{u}_p^k \cdot \mathbf{v}_s^T\right)^2 . \tag{5.4}$$

The playlist factors $\mathbf{u}^{k+1}$ for the next iteration are obtained analogously, by minimizing the approximate cost function where the song factors have been fixed to $\mathbf{v}^k$:

$$\widetilde{\mathcal{L}}_{\mathrm{MF}}\left(\mathbf{u} \mid \mathbf{v} = \mathbf{v}^k, \mathbf{Y}\right) = \sum_{p,s} w_{p,s} \left(y_{p,s} - \mathbf{u}_p \cdot \mathbf{v}_s^{kT}\right)^2 . \tag{5.5}$$

The approximate cost functions (5.4) and (5.5), where one of the factor matrices has been fixed, become quadratic on the other, unknown factor matrix. Thus, they have a unique minimum and it can be found exactly. At each iteration, the original cost function (5.3) is expected to move closer to a local minimum and the procedure is repeated until convergence.

Extension of out-of-set playlists

In principle, CF systems based on matrix factorization can only extend in-set playlists, for which latent playlist factors have been precomputed at training time. However, we observe that ALS enables a fast procedure to obtain reliable playlist factors for out-of-set playlists.

Firstly, we have to insist that ALS is an iterative optimization procedure whose updates are solved exactly. Given the song factors matrix $\mathbf{v}^*$, one update solving for cost function (5.5) yields the playlist factors matrix $\mathbf{u}^*$ deterministically. As an example, imagine two independent optimization processes factorizing the same matrix but initialized differently. If, by chance, both processes reached the same song factors matrix $\mathbf{v}^*$ at whichever iteration, then both processes would derive $\mathbf{u}^*$ as the next playlist factors matrix, regardless of when and how they had arrived at $\mathbf{v}^*$ in the first place. A simple corollary of this observation is that, given the song factors matrix $\mathbf{v}^*$, the playlist factors matrix $\mathbf{u}^*$ derived next is always an equally good solution, regardless of how many ALS iterations had occurred before arriving at $\mathbf{v}^*$.

Assume that the playlist collections P and P' are disjoint. We are interested in predicting continuations for the playlists in the collection P', but at training time we only have access to the collection P. Even though the playlist collections are disjoint, the songs within them are likely not. We arrange the collections P and P' into respective matrices

$\mathbf{Y}$ and $\mathbf{Y}'$ of playlist–song interactions. We factorize the matrix $\mathbf{Y}$ until convergence and keep only the song factors $\mathbf{v}^*$. We can now perform one ALS update solving for the following cost function, which is similar to cost function (5.5) but combines the song factors $\mathbf{v}^*$ (derived from $\mathbf{Y}$) with the matrix $\mathbf{Y}'$:

$$\widetilde{\mathcal{L}}_{\text{Out-of-set}}\left(\mathbf{u} \mid \mathbf{v} = \mathbf{v}^*, \mathbf{Y}'\right) = \sum_{p,s} w_{p,s} \left(y'_{p,s} - \mathbf{u}_p \cdot \mathbf{v}_s^{*T}\right)^2. \tag{5.6}$$

This yields playlist factors $\mathbf{u}'$ for the playlists in P'. We can finally predict extensions for the playlists in P' by reconstructing $\hat{\mathbf{Y}}' = \mathbf{u}' \cdot \mathbf{v}^{*T}$.

Even though the playlist factors are the result of a single ALS update, they are as reliable as the song factors used to derive them. This follows from the reasoning presented above, together with the condition that matrix $\mathbf{Y}'$ is not much more sparse than $\mathbf{Y}$, as this could degrade the results.

5.5.2 Hybrid matrix factorization ("Hybrid MF")

This is a hybrid extension to the just-presented weighted matrix factorization model for implicit feedback datasets (Section 5.5.1). It is based on the exploitation of song latent factors derived from sources other than the playlist–song matrix $\mathbf{Y}$, and it is enabled by an appropriate application of the ALS procedure. This is a hybrid system because the song latent factors are derived from independent song descriptions, such as independent listening logs (Section 5.7.2) or the audio signal (Section 5.7.2).

Let $\mathbf{v}^e$ be a song factors matrix corresponding to the songs in $\mathbf{Y}$ but derived from external song descriptions. We perform one ALS update solving for cost function (5.5) but replacing $\mathbf{v}^k$ by $\mathbf{v}^e$. This yields a playlist factors matrix $\mathbf{u}$ corresponding to the playlists in $\mathbf{Y}$. We can then predict recommendations by reconstructing $\hat{\mathbf{Y}} = \mathbf{u} \cdot \mathbf{v}^{eT}$.

Using this system is not advised when the matrix $\mathbf{Y}$ contains sufficient training data. However, it can be helpful to deal with infrequent songs (poorly represented by pure CF), and it enables the recommendation of out-of-set songs. The distinction between in-set and out-of-set playlists is not meaningful for this system because the song latent factors are derived independently from any playlist collection. Then, one ALS iteration adapts to whichever playlist collection is being considered.

5.5.3 Playlist neighbors-based playlist continuation ("Neighbors")

This is a CF system based on playlist-to-playlist similarities. A playlist p is represented by a binary vector $\mathbf{s}_p \in \{0,1\}^{|S|}$ indicating the songs

that it includes. The similarity of a pair of playlists p, q is computed as the cosine between $\mathbf{s}_p$ and $\mathbf{s}_q$, i,.e.,

$$\text{sim}(p, q) = \cos(\mathbf{s}_p, \mathbf{s}_q) = \frac{\mathbf{s}_p \cdot \mathbf{s}_q}{\|\mathbf{s}_p\| \|\mathbf{s}_q\|}. \tag{5.7}$$

Given a reference playlist collection P, the score assigned to a song s as a candidate to extend a playlist p (which need not belong to P) is computed as

$$\text{score}(s, p) = \sum_{q \in P(s)} \text{sim}(p, q), \tag{5.8}$$

where $P(s)$ are the playlists from the collection P that contain the song s. The system considers the song s to be a suitable continuation for the playlist p if s has occurred in playlists of the collection P that are similar to p.

This system is closely related to the playlist-based k-nearest neighbors system [17, 53, 63]. The difference is that we consider the whole collection P as the neighborhood of p instead of considering only the k playlists most-similar to p. We found in preliminary experiments that, given the sparsity of the playlist collections, considering as many neighbors as possible is beneficial for the computation of the playlist–song scores.

5.5.4 Popularity

This system computes the popularity of a song s according to its relative frequency in a reference playlist collection P, i.e.,

$$\text{pop}(s) = \frac{|P(s)|}{|P|}, \tag{5.9}$$

where $P(s)$ are the playlists from the collection P that contain the song s. Candidate songs to extend a playlist are ranked by their popularity.

5.5.5 Random

This is a dummy system included as a reference. The fitness of any playlist–song pair (p, s) is randomly drawn from a uniform distribution $\mathcal{U}[0, 1]$.

5.6 EVALUATION

We conduct off-line experiments to assess the performance of the playlist continuation systems. Following the evaluation approaches

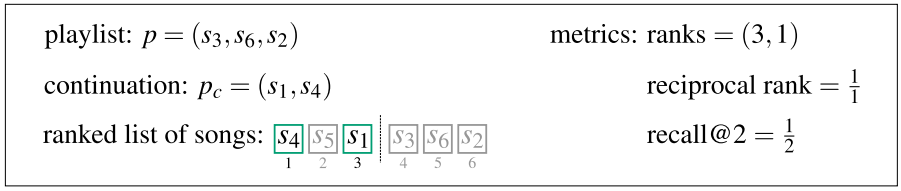


Figure 5.4: Illustration of the off-line experiment for one playlist. A system extends the playlist $p = (s_3, s_6, s_2)$. It ranks all the songs available according to its predictions, leaving out the songs in the playlist p. The songs in the continuation $p_c = (s_1, s_4)$ attain, respectively, ranks 3 and 1 in the ordered list of candidate songs. The first hit is at rank 1, thus the continuation's reciprocal rank is $\frac{1}{1}$. If we recommend the top-2 results, one of the two is a hit, therefore the continuation's recall@2 is $\frac{1}{2}$.

used in the literature [2, 17, 53, 63], we devise a retrieval-based task to measure the ability of the systems to recover withheld playlist continuations. Even though off-line experiments can not directly assess the user satisfaction as user experiments do, they provide a controlled and reproducible approach to compare different systems.

5.6.1 Off-line experiment

Given a playlist p, we assume that a continuation p_c, proportionally shorter than p, is known and withheld for test. For example, if continuations were set to have a length of 25% their original playlist length, two playlists of 8 and 12 songs would have continuations of 2 and 3 songs, respectively. This follows the evaluation methodology used by Aizenberg, Koren, and Somekh [2] but differs from the one used by Hariri, Mobasher, and Burke [53], Bonnin and Jannach [17], and Jannach, Lerche, and Kamehkhosh [63], where the withheld continuations have always one song regardless of the length of the playlist p.

We let the system under evaluation predict the fitness of the playlist–song pair (p, s) for each song $s \in S \setminus p$. The set of recommendable songs is restricted to $S \setminus p$ not to recommend songs from the very playlist p. We rank the candidate songs in the order of preference to extend p given by the system predictions. On the basis of this ordered list of song candidates, we compute rank-based metrics reflecting the ability of the system to recover the songs from the playlist continuation p_c. We find the rank that each song in the withheld continuation p_c occupies within the ordered list of song candidates. We compute two additional metrics for the continuation p_c as a whole: the reciprocal rank, i.e., the inverse of the top-most rank achieved by a song from p_c within the ordered list of song candidates, and the recall@100, i.e., the amount of songs from p_c within the top 100 positions of the ordered list of song candidates (Figure 5.4) [96, Chapter 8].

This process is repeated for all the playlists we set to extend. We finally report the median rank over all the songs in all the continua-

tions, the mean reciprocal rank (MRR) over all the continuations, and the mean recall@100 (R@100) over all the continuations. We construct 95% basic bootstrap confidence intervals for each of the reported metrics [33]. Since these are not necessarily symmetric, to avoid clutter in the tables, we will show the nominal metric value plus/minus the largest margin. For example, a median rank of 1091 with a confidence interval of (1001, 1162) will not be reported as $1001 \pm ^{71}_{90}$, but as 1001 ± 90.

5.6.2 Weak and strong generalization

We consider two evaluation settings as proposed by Aizenberg, Koren, and Somekh [2]. The first setting, or "weak generalization" setting, assumes that only one playlist collection P is available. The playlists are used to train a playlist continuation system. Then, the system recommends continuations to the very training playlists. The second setting, or "strong generalization" setting, assumes that two disjoint playlist collections P and P' are available. The playlists in the collection P are used to train a playlist continuation system. Then, the system recommends continuations to the playlists in the collection P', which it has not seen before.

5.7 DATASETS

We compile two datasets, each consisting of a collection of hand-curated music playlists and feature vectors for each of the songs in the playlists.

The playlists are derived from Art of the Mix[5] and 8tracks,[6] two on-line platforms where music aficionados can publish their playlists. Previous research in automated music playlist continuation has focused on these databases precisely because of the presumable careful curation of their playlists [17, 53, 63, 99, 100].

The song feature vectors are extracted from song audio clips gathered from the content provider 7digital,[7] and from social tags and listening logs obtained from the Million Song Dataset (MSD)[8] [12], a public database providing an heterogeneous collection of data for a million contemporary songs.

5 http://www.artofthemix.org
6 https://8tracks.com
7 https://www.7digital.com
8 https://labrosa.ee.columbia.edu/millionsong

5.7.1 Playlist collections

For Art of the Mix, we use the playlists published in the AotM-2011 dataset, a publicly available corpus of playlists crawled by McFee and Lanckriet [100]. The songs in the playlists that also belong to the MSD come properly identified. For 8tracks, we are given access to a private corpus of playlists. These playlists are represented by plain-text song titles and artist names. We match them against the MSD to get access to song-level descriptions and for comparability with the AotM-2011 dataset. The songs that are not present in the MSD are dropped from both playlist collections because we can not extract feature vectors without their song-level descriptions.

Playlist filtering

We presume that playlists with several songs by the same artist or from the same album may correspond to a not so careful compilation process (e.g., saving a full album as a playlist). We also observe that social tags, which we use for feature extraction, can contain artist or album information. Therefore, we decide not to consider artist- and album-themed playlists to ensure the quality of the playlists and to prevent leaking artist or album information into the evaluation. We keep only playlists with at least 7 unique artists and with a maximum of 2 songs per artist (the thresholds were manually chosen to yield sufficient playlists after the whole filtering process).

This type of filtering, which we already proposed in our previous works [153–156, 158], has also been adopted in the RecSys Challenge 2018.[9] On the other hand, other previous works have typically not filtered the playlists by such criteria [53, 99, 100] and have even investigated the exploitation of artist co-occurrences [17]. We believe that either approach conditions the type of patterns that playlist continuation systems will identify. Thus, filtering the playlists or not can be regarded as a design choice depending on the use case and the target users.

To ensure that the playlist continuation systems learn from playlists of sufficient length, we further keep only the playlists with at least 14 songs. The final length of the playlists may still be shortened because we drop songs missing some type of song-level description, for which we can not extract all the feature vector types. Finally, in order to set up training and evaluation playlist splits, we discard playlists that have become shorter than 5 songs after the song filtering.

The filtered AotM-2011 dataset has 2,711 playlists with 12,286 songs by 4,080 artists. The filtered 8tracks dataset has 3,269 playlists with 14,552 songs by 5,104 artists. Detailed statistics for the final playlist collections are provided in Table 5.1.

9 https://recsys-challenge.spotify.com/details

Table 5.1: Descriptive statistics of the filtered AotM-2011 and 8tracks playlist collections. We report the distribution of playlist lengths, number of artists per playlist, and song frequency in the dataset (i.e., the number of playlists in which each song occurs).

dataset	statistic	min	1q	med	3q	max
AotM-2011	Playlist length	5	7	9	11	26
	Artists per playlist	4	7	9	11	26
	Song frequency	1	1	1	2	42
8tracks	Playlist length	5	8	10	12	38
	Artists per playlist	3	8	10	11	34
	Song frequency	1	1	1	2	140

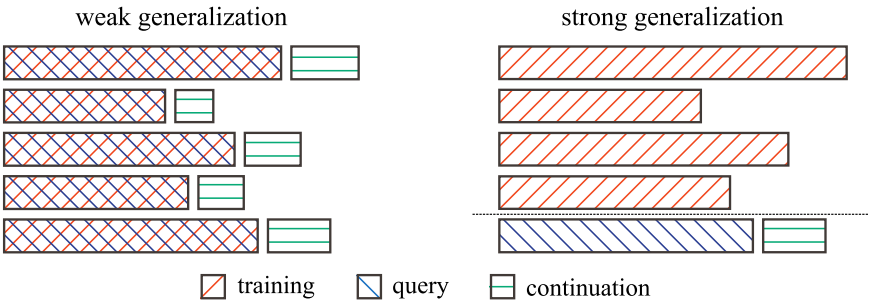


Figure 5.5: Illustration of the playlist splits. In the weak generalization setting every playlist is split withholding the last songs as a continuation. In the strong generalization setting the playlist collection is split into disjoint sub-collections. One sub-collection is used to train the system. Every playlist in the other sub-collection is split withholding the last songs as a continuation. The red stripes indicate the playlists used to train the systems. The blue stripes indicate the playlists that the systems have to extend. The green stripes indicate the withheld continuations used for evaluation.

Playlist splits

We create training and test playlist splits for the weak and strong generalization settings. For the weak generalization setting, we split each playlist leaving approximately the final 20% of the songs as a withheld continuation. For the strong generalization setting, we split each playlist collection into 5 disjoint sub-collections for cross validation. At each iteration, 4 disjoint sub-collections are put together for training and the playlists therein are not split. The playlists in the remaining sub-collection are used for evaluation and are split as in the weak generalization setting (Figure 5.5). The playlists in the training splits of both generalization settings are further split leaving approximately the final 20% of the songs as withheld continuations for validation.

5.7.2 Song features

For all the feature types we extract 200-dimensional vectors. According to our experiments, feature vectors of this dimensionality carry enough information.

Latent factors from independent listening logs ("Logs")

The Echo Nest Taste Profile Subset[10] is a dataset of user listening histories from undisclosed partners. It contains *(user, song, play-count)* triplets for songs included in the MSD. We factorize the triplets using the already discussed weighted matrix factorization model for implicit feedback datasets [60] with a factorization depth of 200 dimensions. However, the weighting scheme now depends on the play-counts. We use the obtained song latent factors as song feature vectors.

Latent factors from audio signal ("Audio2CF")

Following the work by Oord, Dieleman, and Schrauwen [109], we build a feature extractor to predict collaborative filtering song factors from song spectrograms. We use a convolutional neural network inspired by the VGG-style architecture [134] consisting of sequences of 3×3 convolution stacks followed by 2×2 max pooling. To reduce the dimensionality of the network output to the predefined song factor dimensionality, we insert, as a final building block, a 1×1 convolution having 200 feature maps followed by global average pooling [88].

We assemble a training set for the feature extractor using the latent factors of the songs from the Echo Nest Taste Profile Subset (Section 5.7.2) and the corresponding audio previews downloaded from 7digital. To prevent leaking information, we discard the songs present in the playlist collections. We use the trained feature extractor to predict song latent factors for the songs in the playlist collections, given audio snippets that we also download from 7digital.

Semantic features from social tags ("Tags")

The Last.fm Dataset[11] gathers social tags that users of the on-line music service Last.fm[12] assigned to songs included in the MSD. Along with the tag strings, the dataset provides relevance weights describing how well a particular tag applies to a song, as returned by the `tracks.getTopTags` function of the Last.fm API.[13]

We extract semantic features from the tags assigned to a song using *word2vec* [103]. Even though we have experimented with *word2vec*

10 https://labrosa.ee.columbia.edu/millionsong/tasteprofile
11 https://labrosa.ee.columbia.edu/millionsong/lastfm
12 https://www.last.fm
13 https://www.last.fm/api/show/track.getTopTags

models trained on very large text corpora (e.g., on GoogleNews[14]), we obtain best results using models trained on custom, smaller but music-informed text corpora (more details can be found in our previous works [153, 155]).

For each unique song in the playlists, we look up its social tags in the music-informed *word2vec* model. If a tag is a compound of several words (e.g., "pop rock"), we compute the average feature. Since a song may have several tags, the final semantic feature is the weighted average of all its tags' features, where the weights are the relevance weights provided by the Last.fm Dataset.

5.8 RESULTS

Tables 5.2, 5.3 and 5.4, 5.5 report the results achieved in the weak and strong generalization settings, respectively. The Profiles system can only operate in weak generalization and therefore it only appears in Tables 5.2, 5.3. The purely collaborative systems, i.e., MF and Neighbors, can not predict scores for out-of-set songs. During the evaluation of these systems, if a withheld continuation contains an out-of-set song, it is simply ignored. Thus, the overall performance of MF and Neighbors is not directly comparable to the performance of the other systems. To make this information clear, Tables 5.2–5.5 report the number N of songs in the withheld continuations that each system could consider, and the results corresponding to MF and Neighbors are displayed in italics. A fair comparison of the hybrid systems and MF is provided in Section 5.8.4, where the performance of each system is shown as a function of how often the songs in the withheld continuations occurred in training playlists.

5.8.1 Interpreting the results

Figure 5.6 displays the complete recall curve achieved by the playlist continuation systems on the AotM-2011 dataset in the weak generalization setting (all the hybrid systems use the Logs features). To highlight that MF and Neighbors can only deal with in-set songs, their recall curves are represented with dashed lines. Profiles, Membership, MF and Hybrid MF bend considerably to the upper left corner. This shows that they keep on predicting relevant songs as their recommendation lists grow. Neighbors starts similarly, but it quickly flattens, not finding additional relevant recommendations as its recommendation list grows. However, if we look closer at the top 200 results (detail box in Figure 5.6), we find that Neighbors actually starts comparably well to MF, and even better than Profiles and Membership. Its quick decline is likely explained by the high sparsity of the datasets: half

14 https://code.google.com/archive/p/word2vec

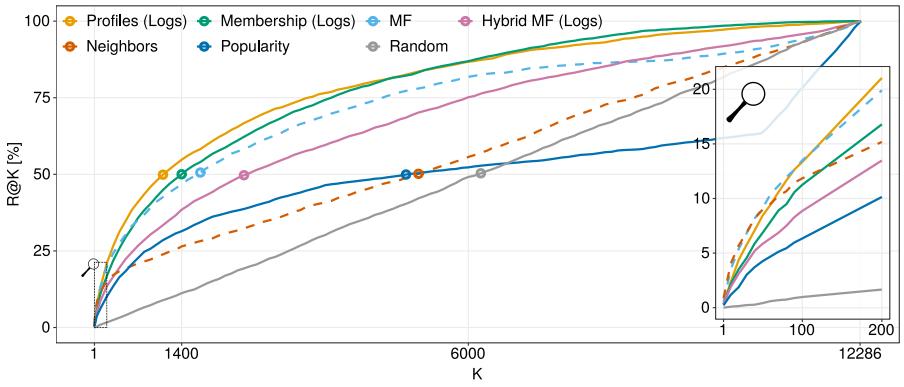


Figure 5.6: Recall curve on the AotM-2011 playlist collection in weak generalization. The top 200 positions are detailed in the box. MF and Neighbors, only evaluated for in-set songs, are displayed with a dashed line.

of the songs occur in one training playlist, and three quarters of the songs occur in no more than 2 training playlists (Table 5.1). Neighbors is only able to successfully predict recommendations for the most frequent songs, which co-occur often, but it is unable to do so for the vast majority of infrequent songs. Similarly, Popularity performs reasonably well for the top-most positions of the recommendation list, but its performance quickly degrades.

The median rank (corresponding approximately to the dots at 50% recall in Figure 5.6) summarizes the overall distribution of ranks attained by a system. The MRR and R@100 capture the performance at the top positions of the recommendation lists. Focusing on the overall performance of the systems, Profiles, Membership and MF are clearly preferable over Neighbors. Looking at the top positions only, Neighbors might appear preferable. It could be argued that only the top positions matter, because a user could not possibly look further than the top 10 recommendations. While this reasoning seems valid in on-line systems, where users react to the predicted recommendations, we believe that it is inaccurate in the context of off-line experiments. Assessing the usefulness of music recommendations is highly subjective. Given a playlist, there are multiple songs that a user could accept as relevant continuations. However, off-line experiments only accept exact matches to the withheld ground-truth continuation [99, 114]. For this reason, measuring the performance of a system focusing only on the top positions of recommendation lists can be misleading. Off-line experiments should be regarded as approximations of the final system performance, and the performance should be measured by the system's global merits. Throughout this section, we will mostly rely on the median rank as the metric to assess the global behavior of the playlist continuation systems.

5.8.2 Overall performance of the playlist continuation systems

For now we only let the proposed systems use the Logs and Audio2CF features, which the Hybrid MF system can also utilize. This makes the comparison fair.

Weak generalization

Profiles and Membership obtain lower (better) median rank than Hybrid MF using Logs and Audio2CF features, respectively (Tables 5.2, 5.3). Using Logs features, Profiles and Membership obtain higher (better) R@100 than Hybrid MF, but their MRR is comparable. Using Audio2CF features, the MRR and R@100 of Profiles, Membership and Hybrid MF are comparable in the AotM-2011 dataset, but Membership is clearly better than Profiles and Hybrid MF in the 8tracks dataset.

For all the hybrid systems (Profiles, Membership and Hybrid MF) the Logs features yield better results than the Audio2CF features, regardless of the metric considered. The improvements are not equally pronounced in all the combinations of systems and datasets, but the gains are clear. This is expected because Audio2CF features are an approximation of Logs features derived from the audio signal of songs. Despite its remarkable results, the Audio2CF features can not bridge the music semantic gap [22, 109].

The performance of Profiles and Membership is comparable. Overall, Profiles can achieve a higher performance when using Logs features, but Membership is superior using Audio2CF features, especially in the 8tracks dataset.

Finally, we comment on the performance of the pure CF baselines. MF obtains a clearly lower (better) median rank than Neighbors. On the other hand, Neighbors obtains MRR and R@100 comparable to MF. The reason for this apparent mismatch between median rank, MRR and R@100 was just exposed in Section 5.8.1.

Strong generalization

Membership obtains a clearly lower (better) median rank than Hybrid MF, regardless of the feature used (Tables 5.4, 5.5). Using Logs features, Membership and Hybrid MF obtain comparable MRR and R@100. Using Audio2CF features, the MRR and R@100 of Membership and Hybrid MF are comparable in the AotM-2011 dataset, but Membership is clearly better in the 8tracks dataset.

For the hybrid systems, again the Logs features yield better results than the Audio2CF features. Indeed, the different information that these two features carry is independent of the generalization setting used to evaluate the systems.

Table 5.2: Weak generalization results for the AotM-2011 dataset. The first two columns indicate the system and feature names (if any). The third column indicates the number N of songs in the withheld continuations involved in each experiment. MF and Neighbors, only evaluated for in-set songs, are displayed in italics. The median rank is relative to 12,286 unique songs. Lower is better. For MRR and R@100 higher is better. Systems clearly outperforming the best result of Hybrid MF (non-overlapping 95% CIs) are indicated in bold. MF or Neighbors clearly outperforming one another are indicated in italic bold.

AotM-2011 - weak generalization

system	feature	N	med rank	MRR [%]	R@100 [%]
Profiles	Audio2CF + Tags + Logs	4473	**891 ± 63**	**2.49 ± 0.41**	**15.91 ± 1.23**
	Logs	4473	**1091 ± 90**	1.91 ± 0.33	**13.34 ± 1.12**
	Audio2CF	4473	2298 ± 127	0.75 ± 0.20	6.68 ± 0.83
Membership	Audio2CF + Tags + Logs	4473	**1052 ± 73**	**2.18 ± 0.37**	**14.88 ± 1.16**
	Logs	4473	**1391 ± 101**	1.71 ± 0.34	**11.25 ± 1.01**
	Audio2CF	4473	**2042 ± 113**	1.08 ± 0.25	7.56 ± 0.88
MF	—	*2835*	***1572 ± 150***	*2.21 ± 0.45*	*13.50 ± 1.36*
Hybrid MF	Logs	4473	2404 ± 164	1.47 ± 0.32	8.86 ± 0.92
	Audio2CF	4473	2921 ± 164	0.74 ± 0.21	5.60 ± 0.77
Neighbors	—	*2835*	*5112 ± 321*	*2.66 ± 0.55*	*11.84 ± 1.28*
Popularity	—	4473	5217 ± 570	1.03 ± 0.27	6.34 ± 0.80
Random	—	4473	6163 ± 218	0.10 ± 0.03	0.98 ± 0.34

Table 5.3: Weak generalization results for the 8tracks dataset. The first two columns indicate the system and feature names (if any). The third column indicates the number N of songs in the withheld continuations involved in each experiment. MF and Neighbors, only evaluated for in-set songs, are displayed in italics. The median rank is relative to 14,552 unique songs. Lower is better. For MRR and R@100 higher is better. Systems clearly outperforming the best result of Hybrid MF (non-overlapping 95% CIs) are indicated in bold. MF or Neighbors clearly outperforming one another are indicated in italic bold.

8tracks - weak generalization

system	feature	N	med rank	MRR [%]	R@100 [%]
Profiles	Audio2CF + Tags + Logs	6289	**556 ± 38**	**3.90 ± 0.43**	**23.51 ± 1.21**
	Logs	6289	**718 ± 58**	**3.62 ± 0.44**⋆	**20.03 ± 1.12**
	Audio2CF	6289	1988 ± 90	1.13 ± 0.24	8.21 ± 0.80
Membership	Audio2CF + Tags + Logs	6289	**652 ± 46**	**3.73 ± 0.44**	**20.43 ± 1.12**
	Logs	6289	**986 ± 68**	2.89 ± 0.37	16.92 ± 1.04
	Audio2CF	6289	**1149 ± 69**	2.49 ± 0.35	15.50 ± 1.02
MF	—	*4299*	***1198 ± 145***	*4.04 ± 0.56*	*18.62 ± 1.29*
Hybrid MF	Logs	6289	1677 ± 138	2.79 ± 0.41	15.88 ± 1.05
	Audio2CF	6289	2636 ± 142	1.09 ± 0.24	7.03 ± 0.75
Neighbors	—	*4299*	*3566 ± 318*	*4.93 ± 0.64*	*19.50 ± 1.32*
Popularity	—	6289	3352 ± 272	1.80 ± 0.33	9.71 ± 0.81
Random	—	6289	7094 ± 234	0.15 ± 0.06	0.78 ± 0.24

⋆ The 95% bootstrap CIs for the MRR of Profiles (Logs) and Hybrid MF (Logs) do not overlap. However, this is not apparent from the table because, for readability, we only display the largest margin of the CIs.

Compared to one another, the pure CF baselines behave similarly as in weak generalization. MF obtains lower (better) median rank than Neighbors. However, Neighbors obtains R@100 comparable to MF and MRR superior than MF, especially in the AotM-2011 dataset.

Robustness to strong generalization

We analyze the robustness of each system (but Profiles) to the strong generalization setting by comparing whether its performance degrades from Tables 5.2, 5.3 to Tables 5.4, 5.5.

Membership performs comparably well in both generalization settings regardless of the feature utilized, the metric considered, and the dataset. This result indicates that regarding playlist–song pairs exclusively in terms of feature vectors (the key characteristic of Membership) does favor generalization and discourages the specialization towards particular training playlists.

The performance of MF is also comparable in weak and strong generalization. This supports the approach detailed in Section 5.5.1, by which latent song factors derived from the factorization of a collection of training playlists can be successfully utilized to extend out-of-set playlists.

We pointed out in Section 5.5.2 that Hybrid MF does not distinguish between in-set and out-of-set playlists. Now we observe that the performance of Hybrid MF is identical in both generalization settings. Only the confidence intervals are slightly different due to the randomness involved in bootstrap resampling.

The performance of Neighbors is not harmed in strong generalization. In fact, it is slightly superior for the 8tracks dataset and superior for the AotM-2011 dataset. Popularity is also not affected when recommending out-of-set playlists.

5.8.3 Combined features

Until now we only considered the proposed systems with Logs and Audio2CF features to make the comparison to Hybrid MF fair. However, Profiles and Membership can flexibly exploit any type of song feature vector. In particular, they can utilize feature vectors resulting from the concatenation of other feature vectors. This simple approach already yields performance gains.

To illustrate this effect, we now consider the Tags features as well. For each song, we create a combined feature by concatenating its Audio2CF, Tags and Logs feature vectors. Since each individual feature vector has 200 dimensions, the resulting feature vector is 600-dimensional. Profiles and Membership achieve clearly better results with the combined feature than with the Logs feature in terms of

Table 5.4: Strong generalization results for the AotM-2011 dataset. The first two columns indicate the system and feature names (if any). The third column indicates the number N of songs in the withheld continuations involved in each experiment. MF and Neighbors, only evaluated for in-set songs, are displayed in italics. The median rank is relative to 12,286 unique songs. Lower is better. For MRR and R@100 higher is better. Systems clearly outperforming the best result of Hybrid MF (non-overlapping 95% CIs) are indicated in bold. MF or Neighbors clearly outperforming one another are indicated in italic bold.

AotM-2011 - strong generalization

system	feature	N	med rank	MRR [%]	R@100 [%]
Membership	Audio2CF + Tags + Logs	4473	**1150 ± 72**	1.88 ± 0.33	**13.69 ± 1.13**
	Logs	4473	**1552 ± 96**	1.42 ± 0.29	10.71 ± 1.04
	Audio2CF	4473	**2065 ± 120**	0.93 ± 0.21	7.05 ± 0.85
MF	—	*2849*	***1428 ± 135***	*2.72 ± 0.53*	*15.43 ± 1.44*
Hybrid MF	Logs	4473	2404 ± 164	1.47 ± 0.31	8.86 ± 0.94
	Audio2CF	4473	2921 ± 164	0.74 ± 0.21	5.60 ± 0.78
Neighbors	—	*2849*	*4502 ± 304*	***4.89 ± 0.82***	*15.16 ± 1.43*
Popularity	—	4473	5143 ± 583	1.03 ± 0.27	7.28 ± 0.85
Random	—	4473	6161 ± 188	0.13 ± 0.05	1.23 ± 0.37

Table 5.5: Strong generalization results for the 8tracks dataset. The first two columns indicate the system and feature names (if any). The third column indicates the number N of songs in the withheld continuations involved in each experiment. MF and Neighbors, only evaluated for in-set songs, are displayed in italics. The median rank is relative to 14,552 unique songs. Lower is better. For MRR and R@100 higher is better. Systems clearly outperforming the best result of Hybrid MF (non-overlapping 95% CIs) are indicated in bold. MF or Neighbors clearly outperforming one another are indicated in italic bold.

8tracks - strong generalization

system	feature	N	med rank	MRR [%]	R@100 [%]
Membership	Audio2CF + Tags + Logs	6289	**710 ± 51**	3.46 ± 0.43	**20.40 ± 1.12**
	Logs	6289	**1038 ± 78**	2.85 ± 0.39	15.90 ± 1.03
	Audio2CF	6289	**1317 ± 86**	2.58 ± 0.37	14.54 ± 0.99
MF	—	*4298*	***997.5 ± 109.5***	*4.48 ± 0.61*	*20.32 ± 1.35*
Hybrid MF	Logs	6289	1677 ± 138	2.79 ± 0.41	15.88 ± 1.04
	Audio2CF	6289	2636 ± 142	1.09 ± 0.24	7.03 ± 0.75
Neighbors	—	*4298*	*3045.5 ± 390.5*	*5.27 ± 0.68*	*20.43 ± 1.41*
Popularity	—	6289	3293 ± 278	1.76 ± 0.32	9.32 ± 0.78
Random	—	6289	6876 ± 158	0.14 ± 0.06	0.73 ± 0.25

median rank and R@100, and modest but visible improvements in terms of MRR (Tables 5.2–5.5).

We have just exposed an example of a combined feature vector to illustrate the capability of the proposed systems. Appendix 5.B introduces additional feature types, and Appendix 5.C provides an exhaustive evaluation of the results achieved using all the feature types and their combinations.

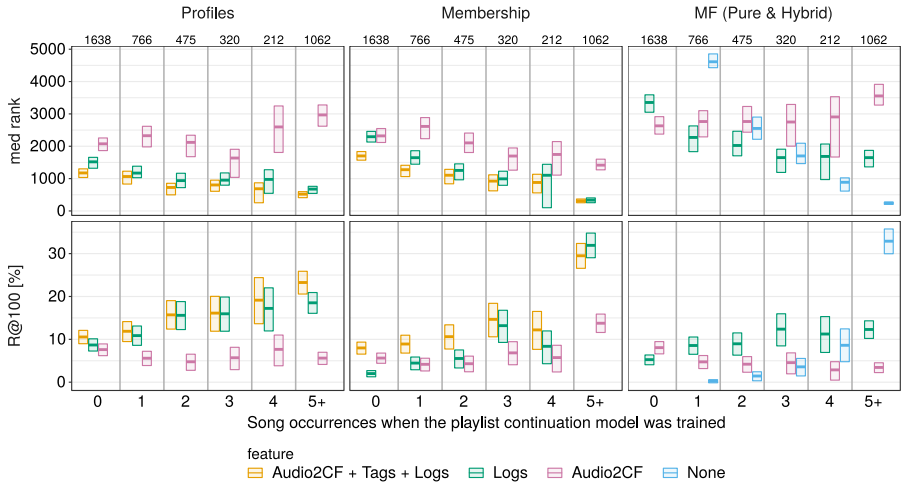
5.8.4 Infrequent and out-of-set songs

We analyze the performance of the hybrid systems Profiles, Membership and Hybrid MF, as well as the performance of the purely collaborative system MF, as a function of how often the songs in the withheld continuations occurred in training playlists. We restrict the analysis to the weak generalization setting, but the results are comparable in the strong generalization setting. Figure 5.7 reports the results. Profiles and Membership can use Audio2CF, Logs, or the combined feature described in Section 5.8.3. Hybrid MF can only use Audio2CF and Logs features. For the sake of space, MF is represented together with Hybrid MF in the figure. The legend, which indicates the color associated to each feature, points to MF with a dummy feature called "None."

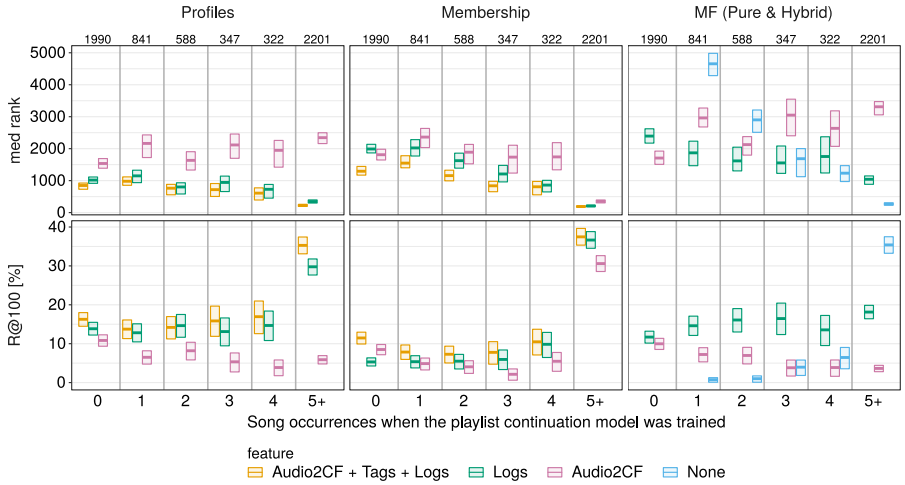
MF can not recommend out-of-set songs, and it achieves very low performance for songs that occurred in only one training playlist. This is expected because purely collaborative systems can not derive patterns in absence of sufficient playlist–song co-occurrences. MF steadily improves its performance as songs become more frequent in training playlists, until it achieves a very competitive performance for songs occurring in 5+ training playlists.

Hybrid MF with Logs features outperforms MF for very infrequent songs. Its performance improves as songs become more frequent, but it does not achieve the high performance of MF for frequent songs. Hybrid MF with Audio2CF features only outperforms MF for very infrequent songs, but MF quickly becomes better.

Profiles and Membership compete with Hybrid MF in terms of R@100, both using Logs and Audio2CF features. In terms of the median rank, Profiles and Membership compete with Hybrid MF using Audio2CF features, and they are generally superior using Logs features. Profiles and Membership further improve their performance using the combined feature, with which they perform reasonably well even for out-of-set and very infrequent songs. Using the combined features or Logs features, and despite some fluctuations, the proposed systems improve their performance as songs become more frequent, with results competitive with those of MF for songs occurring in 5+ training playlists, especially in the 8tracks dataset.



(a) AotM-2011: weak generalization



(b) 8tracks: weak generalization

Figure 5.7: Weak generalization results as a function of how often the songs in the withheld continuations occurred in training playlists. Left, center and right panels correspond to different systems. Upper and lower panels report the median rank (lower is better) and the R@100 (higher is better). The central values in the boxes correspond to the nominal metric value, and the ends correspond to the 95% CI. Each color corresponds to a feature type (MF is labeled as "None"). The text annotations on top indicate the number of songs in the withheld continuations falling in each box.

5.8.5 Additional remarks on the sparsity of playlist collections

The hybrid systems discussed throughout the chapter mitigate the sparsity of the playlist collections by introducing inherent song relationships derived from content information. In this way, they provide performance improvements over systems based exclusively on the playlist collections. An alternative, compatible approach to reduce the sparsity of playlist collections consists in representing songs by their artists. Bonnin and Jannach [17] proposed the Collocated Artists Greatest Hits (CAGH) system, a song-neighbors CF system where the pairwise song similarities are replaced by the pairwise similarities of their artists, and the obtained playlist–song scores are further scaled by the frequency of the songs in the training playlists. We have experimented with CAGH, as well as with a variation of CAGH where the playlist–song scores are not scaled by the song frequency, which we name "Artists." For comparison, we also evaluate Profiles and Membership for in-set songs only (Table 5.6). By design, CAGH is likely suffering from a bias towards popular songs that should be investigated in more detail. That is, the apparent outstanding performance of CAGH, even if only for in-set songs, may be the result of averaging accurate predictions for few but frequent songs, with poor predictions for many but infrequent songs (Chapter 4). In any case, Artists should not be affected by such bias and also provides competitive results that seem to validate the assumption that songs can be successfully approximated by their artists. This points to an interesting line for future work, namely the combination of hybridization and artist-level representations to reduce the sparsity in playlist collections.

5.9 CONCLUSION

We have introduced Profiles and Membership, two feature-combination hybrid recommender systems for automated music playlist continuation. The proposed systems extend collaborative filtering by not only considering hand-curated playlists but also incorporating any type of song feature vector. We have designed feature-combination hybrids, that is, systems that consolidate collaborative and content information into enhanced, standalone systems. Even though we have focused on music playlist continuation, the proposed systems are domain-agnostic and can be applied to other item domains.

We have conducted an exhaustive off-line evaluation to assess the ability of the proposed systems to retrieve withheld playlist continuations, and to compare them to the state-of-the-art pure and hybrid collaborative systems MF and Hybrid MF. The results of the off-line experiments indicate that Profiles and Membership compete with MF when sufficient training data is available and outperform MF for infre-

Table 5.6: Selection of weak generalization results in the AotM-2011 dataset for in-set songs only. The first two columns indicate the system and feature names (if any). The third column indicates the number N of songs in the withheld continuations involved in each experiment. The median rank is relative to 12,286 unique songs. Lower is better. For MRR and R@100 higher is better.

AotM-2011 - weak generalization on in-set songs

system	feature	N	med rank	MRR [%]	R@100 [%]
Profiles	Audio2CF + Tags + Logs	2835	735 ± 76	2.64 ± 0.50	17.76 ± 1.53
Membership	Audio2CF + Tags + Logs	2835	756 ± 73	2.39 ± 0.45	18.08 ± 1.54
CAGH	—	2835	821 ± 89	1.78 ± 0.36	15.18 ± 1.41
Artists	—	2835	1463 ± 140	0.76 ± 0.20	8.33 ± 1.13

quent and out-of-set songs. Profiles and Membership compete to, or outperform Hybrid MF when using Logs or Audio2CF features. The flexibility of the proposed systems to exploit any type of song feature vector provides a straightforward means of further improving their performance by simply combining different song feature vectors.

We have also evaluated MF, Hybrid MF and Neighbors thoroughly. Hybrid MF outperforms MF only for very infrequent songs. Thus, if we were restricted to use MF and Hybrid MF, it would seem advisable to use Hybrid MF for infrequent songs (occurring in up to 3 training playlists in our experiments) and switch to MF for more frequent songs. It should be noted that the proposed systems, Profiles and Membership, take care of this switch automatically. We have also investigated why Neighbors obtains competitive MRR and R@100 metrics but very poor median rank, and we have discussed the interpretation of these metrics.

We have observed the low predictive performance of Audio2CF features, derived exclusively from audio. While this can be explained by the music semantic gap, using Audio2CF features (or other purely audio-based features) as standalone features should be restricted to the recommendation of new releases, where the audio signal is the only song information available.

Preliminary experiments point to building artist-level representations into hybrid recommender systems as a promising line of future work.

5.A ADDITIONAL SYSTEM DETAILS

5.A.1 Profiles

System configuration

We conducted an initial, non-exhaustive exploration of architectures on a validation set by evaluating networks with {2, 3, 4} hidden layers, {50, 100, 200, 500} hidden units, learning rate values in {0.1, 0.5, 1.0}, and batch sizes of {10, 50, 100, 200} songs. We also experimented with the hyperbolic tangent, the logistic function, and the rectifier as activation functions for the hidden layers.

Given the results of the initial exploration, we systematically explored all the combinations of networks with {2, 3} hidden layers and with {50, 100, 200} hidden units. We decided to fix the number of layers to 3 and the number of units to 100. We further fixed the learning rate to 0.5, the batch size to 50 songs, and the hyperbolic tangent as the activation function for hidden layers. The output layer of the network is passed through logistic functions by design (Section 5.4.1).

We used batch normalization [61]. We also experimented with different dropout probabilities [138] and with L1 and L2 regularization to

prevent overfitting. We finally decided to use dropout with probabilities 0.1 and 0.5 at the input layer and the hidden layers, respectively. The feature vectors were standardized and L2-normalized.

The networks were optimized to minimize the cost function (5.1) using AdaGrad [38] with Nesterov momentum [107]. We trained for a maximum of 1,000 epochs but stopped before if the value of the cost function on the validation set did not decrease for 50 epochs. The cost function drove the optimizer, but the best model was chosen on the basis of the highest recall achieved on the validation set. We also used the recall on the validation set to decide an appropriate number of epochs for the final training on the entire training set. We implemented the networks using Lasagne [34], which is built on top of Theano [143].

Computational requirements

The instance of this system trained on the AotM-2011 dataset using Logs features has 323,144 learnable weights (almost 4 and 10 times fewer than Membership and MF, respectively). The system is trained with any variant of stochastic gradient descent [18] using mini-batches. That is, even if a much larger playlist collection were considered, the system could still be trained efficiently in the same manner. Training the aforementioned system until it achieved the reported results required 666.07 seconds (roughly 11 minutes) on a desktop computer with an Intel Core i5-4570 CPU. The system required 4.46 seconds to make the predictions necessary for the weak generalization evaluation. The implementation of this system was not optimized to obtain fast training and prediction times.

5.A.2 Membership

System configuration

We conducted an heuristic exploration of architectures and hyperparameters. We chose the architecture detailed in Table 5.7, which we found to provide good performance. We selected the hyperparameters that yielded lowest cost on the validation set. We used an initial learning rate of 0.001 and a batch size of 100 playlist–song pairs (50 matches and 50 mismatches). The features were not standardized nor normalized. The networks were optimized to minimize the cost function (5.2) using Adam [71]. We trained for a maximum of 1,000 epochs but stopped before if the value of the cost function on the validation set did not decrease for 25 epochs. We implemented the networks using Lasagne [34] and Theano [143].

Table 5.7: Membership architecture. The input to the system are the features $(\mathbf{X}_p, \mathbf{x}_s)$ of a playlist–song pair (p, s). $\mathbf{X}_p^t$ denotes the t-th row of the feature matrix $\mathbf{X}_p$, i.e., the t-th song of the playlist p. The upper part of the table corresponds to the feature transformation component **t**, and the lower part of the table corresponds to the match-discrimination component d. The boldface layers DE_k, BN_k in the transformation component **t** share their weights for all the songs in the playlist p and for song s (Section 5.4.2). The dimensionality of each layer is annotated in parentheses. DE: Dense layer, RE: Rectifier activation function, BN: Batch Normalization [61], DR: Dropout [138].

	$\mathbf{X}_p^1$ (D)	...	$\mathbf{X}_p^{\lvert p\rvert}$ (D)	$\mathbf{x}_s$ (D)
t:	**DE_1** + RE (512)	...	**DE_1** + RE (512)	**DE_1** + RE (512)
	BN_1 + DR (512)	...	**BN_1** + DR (512)	**BN_1** + DR (512)
	DE_2 + RE (512)	...	**DE_2** + RE (512)	**DE_2** + RE (512)
	BN_2 + DR (512)	...	**BN_2** + DR (512)	**BN_2** + DR (512)
	Average (512)			
	Concatenate (1024)			
d:	DE_3 + RE (512) BN_3 + DR (512) DE_4 + RE (512) BN_4 + DR (512) DE_5 + RE (1) Cost			

Computational requirements

Membership is computationally more demanding than Profiles, meaning that we need to use a larger neural network to obtain competitive results. This is reasonable after all, because by not specializing towards the training playlists, Membership accomplishes a much more general task. The instance of this system trained on the AotM-2011 dataset using Logs features has 1,159,681 learnable weights (almost 3 times fewer than MF). The system is trained with any variant of stochastic gradient descent [18] using mini-batches. That is, even if a much larger playlist collection were considered, the system could still be trained efficiently in the same manner. Training the aforementioned system until it achieved the reported results required 3649.90 seconds (roughly 1 hour) on a GeForce GTX 1080 Ti GPU. On the same GPU, the system required 1193.88 seconds (roughly 20 minutes) to make the predictions necessary for the weak generalization evaluation. The implementation of this system was not optimized to obtain fast training and prediction times.

5.A.3 MF and Hybrid MF

System configuration

We used the validation set to experiment with different values for the weight w_1 (Section 5.5.1). We found that using $w_1 = 2$ yielded

best results. MF and Hybrid MF use L2 regularization to prevent overfitting [60]. We did not describe the regularization in Section 5.5.1 for simplicity, but the discussion and properties of ALS remain valid. We decided to use a weight of 10 for the L2 regularization term based on experiments on the validation set.

Computational requirements

The instance of MF trained on the AotM-2011 dataset has 2,999,400 learnable weights. The system is trained efficiently with ALS. Training MF until it achieved the reported results required 190.08 seconds (roughly 3 minutes) on a desktop computer with an Intel Core i5-4570 CPU. MF required 0.22 seconds to make the predictions necessary for the weak generalization evaluation. Hybrid MF required 2.48 seconds to complete the weak generalization evaluation, including the ALS update and the predictions. We used the publicly available implementation of MF provided by Frederickson [49], which is highly optimized to obtain fast training times. The predictions, which simply consist in the multiplication of low rank matrices, are very fast.

5.A.4 Neighbors

This system does not require adjusting any hyperparameters or learning any weights. The weak evaluation of Neighbors on the AotM-2011 dataset using a desktop computer with an Intel Core i5-4570 CPU required 0.81 seconds to compute the pairwise playlist similarities given by Equation (5.7) and 1.38 seconds to compute the playlist–song scores given by Equation (5.8). Given the moderate size of the considered AotM-2011 and 8tracks playlist collections, both operations could be implemented as fast matrix multiplications.

5.B ADDITIONAL SONG FEATURES

5.B.1 Semantic features from audio signal ("Audio2Tag")

The on-line music service Jamendo[15] hosts songs under Creative Commons[16] licenses and allows free downloads. We obtain a dump of the Jamendo database, that includes, among other types of information, the social tags assigned to songs hosted in Jamendo. Then, we use the code released by Sonnleitner and Widmer [136] to collect the audio files corresponding to the database dump. This results in a collection of roughly 200k songs independent of the playlists collections and the

15 https://www.jamendo.com
16 https://creativecommons.org

MSD, for which both a full-length audio file and a list of social tags is known.

We build an auto-tagger system based on a convolutional neural network. The network takes a song spectrogram as input, processes it, and predicts the probability of this song being labeled with each of a series of known tags. We train this network using the audio files and the social tags from the Jamendo database. We use only the tags that have been assigned to at least 1,000 songs, resulting in 156 unique tags and 187,663 songs.

The tagging network is similar to the Audio2CF feature extractor (Section 5.7.2). It is inspired by the VGG-style architecture [134] and consists of sequences of 3×3 convolution stacks followed by 2×2 max pooling. To reduce the dimensionality of the network output to the predefined number of tags, we insert, as a final building block, a 1×1 convolution having 156 feature maps followed by global average pooling [88]. Each output neuron, which corresponds to one of the tags, is followed by a logistic activation function to obtain a valid tag probability. However, we use the 156-dimensional output layer before it is passed through the activation function as the Audio2Tag feature.

We use the trained auto-tagger to predict Audio2Tag features for the songs in the playlist collections, given audio snippets downloaded from 7digital. This approach is related to the systems presented by Liang, Zhan, and Ellis [85] and Choi, Fazekas, and Sandler [27].

5.B.2 I-vectors from timbral features

I-vectors were first introduced in the field of speaker verification [32], but recently they have also been successfully utilized for music similarity and music artist recognition tasks [39, 40].

The MSD splits songs into segments of variable length (typically under a second) and provides 12-dimensional timbral coefficients for each segment similar to MFCCs [117]. We build a Gaussian mixture model with 1,024 components using the segment-level timbral features of a collection of representative songs (more details can be found in our previous works [153, 155]). Using the unique songs in the playlists, we derive the total variability space yielding 200-dimensional i-vectors. Following the standard i-vector extraction pipeline, we further transform the obtained i-vectors using a linear discriminant analysis model [54] fit on the training playlists.

5.C ADDITIONAL RESULTS

We provide additional results to demonstrate the performance of Profiles and Membership with different types of song features, and with stepwise combinations of two or three types of song features.

Tables 5.8 and 5.9 report weak generalization results for the AotM-2011 and the 8tracks datasets, respectively. Tables 5.10 and 5.11 report strong generalization results for the AotM-2011 and the 8tracks datasets, respectively.

We also analyze the performance of Profiles, Membership and Hybrid MF simulating the recommendation of new song releases. To this end, we pretend that the audio signal is the only type of song description available, and we only consider songs that occurred in 4 or less training playlists. Tables 5.12 and 5.13 report the results for the AotM-2011 and the 8tracks datasets, respectively.

Besides the median rank, the MRR and the R@100, Tables 5.8–5.13 also report the Mean Average Precision (MAP), the R@10 and the R@30.

Table 5.8: Weak generalization results for the AotM-2011 dataset. The first two columns indicate the system and feature names (if any). The third column indicates the number N of songs in the withheld continuations involved in each experiment. MF and Neighbors, only evaluated for in-set songs, are displayed in italics. The median rank is relative to 12,286 unique songs. Lower is better. For MRR, MAP and R@{10, 30, 100} higher is better. Systems clearly outperforming Hybrid MF (non-overlapping 95% CIs) are indicated in bold. MF or Neighbors clearly outperforming one another are indicated in italic bold.

AotM-2011 - weak generalization

system	feature	N	med rank	MRR [%]	MAP [%]	R@10 [%]	R@30 [%]	R@100 [%]
Profiles	Audio2CF + Tags + Logs	4473	**891 ± 63**	**2.49 ± 0.41**	**1.75 ± 0.28**	**3.47 ± 0.61**	**7.21 ± 0.85**	**15.91 ± 1.23**
	Tags + Logs	4473	**947 ± 74**	2.01 ± 0.33	1.42 ± 0.22	2.82 ± 0.55	**6.59 ± 0.84**	**14.81 ± 1.17**
	Audio2CF + Logs	4473	**1010 ± 86**	1.98 ± 0.33	1.44 ± 0.27	2.57 ± 0.54	**6.17 ± 0.80**	**14.29 ± 1.16**
	Logs	4473	**1091 ± 90**	1.91 ± 0.33	1.31 ± 0.24	2.33 ± 0.49	**5.75 ± 0.74**	**13.34 ± 1.12**
	Audio2CF + Tags	4473	**1340 ± 95**	1.65 ± 0.31	1.22 ± 0.24	2.32 ± 0.52	5.23 ± 0.77	**12.15 ± 1.08**
	Tags	4473	**1453 ± 114**	1.45 ± 0.26	1.05 ± 0.20	1.84 ± 0.44	**5.80 ± 0.80**	**11.78 ± 1.05**
	Audio2CF	4473	2298 ± 127	0.75 ± 0.20	0.54 ± 0.14	0.80 ± 0.31	2.38 ± 0.52	6.68 ± 0.83
	i-vectors	4473	2820 ± 167	0.59 ± 0.17	0.41 ± 0.13	0.60 ± 0.25	1.45 ± 0.40	4.45 ± 0.71
	Audio2Tag	4473	3315 ± 172	0.31 ± 0.08	0.23 ± 0.06	0.25 ± 0.18	0.98 ± 0.33	3.14 ± 0.57
Membership	Audio2CF + Tags + Logs	4473	**1052 ± 73**	**2.18 ± 0.37**	**1.52 ± 0.26**	**2.95 ± 0.58**	**6.45 ± 0.81**	**14.88 ± 1.16**
	Audio2CF + Logs	4473	**1144 ± 78**	1.97 ± 0.34	1.35 ± 0.24	2.34 ± 0.50	5.43 ± 0.73	**12.88 ± 1.09**
	Tags + Logs	4473	**1145 ± 94**	**2.20 ± 0.37**	**1.54 ± 0.28**	2.68 ± 0.52	**6.09 ± 0.78**	**14.11 ± 1.15**
	Tags	4473	**1201 ± 104**	1.62 ± 0.29	1.16 ± 0.20	2.08 ± 0.48	4.92 ± 0.71	**12.55 ± 1.08**
	Audio2CF + Tags	4473	**1243 ± 94**	1.53 ± 0.26	1.12 ± 0.20	2.08 ± 0.48	**5.70 ± 0.78**	**12.76 ± 1.09**
	Logs	4473	**1391 ± 101**	1.71 ± 0.34	1.15 ± 0.22	2.16 ± 0.47	4.56 ± 0.69	**11.25 ± 1.01**
	Audio2CF	4473	**2042 ± 113**	1.08 ± 0.25	0.77 ± 0.19	1.28 ± 0.37	2.96 ± 0.58	7.56 ± 0.88
	i-vectors	4473	2632 ± 179	1.01 ± 0.22	0.72 ± 0.16	1.32 ± 0.37	3.26 ± 0.57	8.13 ± 0.88
	Audio2Tag	4473	2887 ± 163	0.57 ± 0.18	0.40 ± 0.12	0.69 ± 0.29	1.66 ± 0.44	4.48 ± 0.70
MF	—	*2835*	***1572 ± 150***	*2.21 ± 0.45*	*1.73 ± 0.37*	3.49 ± 0.74	*6.83 ± 1.01*	*13.50 ± 1.36*
Hybrid MF	Logs	4473	2404 ± 164	1.47 ± 0.32	1.03 ± 0.22	1.90 ± 0.47	4.12 ± 0.66	8.86 ± 0.92
	Audio2CF	4473	2921 ± 164	0.74 ± 0.21	0.54 ± 0.15	0.91 ± 0.33	2.33 ± 0.52	5.60 ± 0.77
Neighbors	—	*2835*	*5112 ± 321*	*2.66 ± 0.55*	*2.04 ± 0.43*	*4.02 ± 0.79*	*6.88 ± 1.04*	*11.84 ± 1.28*
Popularity	—	4473	5217 ± 570	1.03 ± 0.27	0.67 ± 0.17	1.17 ± 0.36	3.01 ± 0.57	6.34 ± 0.80
Random	—	4473	6163 ± 218	0.10 ± 0.03	0.07 ± 0.02	0.09 ± 0.13	0.24 ± 0.18	0.98 ± 0.34

Table 5.9: Weak generalization results for the 8tracks dataset. The first two columns indicate the system and feature names (if any). The third column indicates the number N of songs in the withheld continuations involved in each experiment. MF and Neighbors, only evaluated for in-set songs, are displayed in italics. The median rank is relative to 14,552 unique songs. Lower is better. For MRR, MAP and R@{10, 30, 100} higher is better. Systems clearly outperforming Hybrid MF (non-overlapping 95% CIs) are indicated in bold. MF or Neighbors clearly outperforming one another are indicated in italic bold.

8tracks - weak generalization

system	feature	N	med rank	MRR [%]	MAP [%]	R@10 [%]	R@30 [%]	R@100 [%]
Profiles	Audio2CF + Tags + Logs	6289	**556 ± 38**	**3.90 ± 0.43**	**2.46 ± 0.27**	**4.79 ± 0.62**	**11.46 ± 0.89**	**23.51 ± 1.21**
	Tags + Logs	6289	**587 ± 42**	**3.85 ± 0.45**	**2.48 ± 0.29**	**4.84 ± 0.61**	**10.65 ± 0.88**	**22.37 ± 1.20**
	Audio2CF + Logs	6289	**659 ± 45**	3.40 ± 0.41	2.17 ± 0.26	4.09 ± 0.56	**9.42 ± 0.83**	**20.62 ± 1.15**
	Logs	6289	**718 ± 58**	**3.62 ± 0.44***	**2.31 ± 0.28**	**4.30 ± 0.58**	**9.92 ± 0.83**	**20.03 ± 1.12**
	Audio2CF + Tags	6289	**965 ± 61**	2.84 ± 0.40	1.76 ± 0.25	3.22 ± 0.49	7.28 ± 0.75	15.74 ± 1.03
	Tags	6289	**1084 ± 65**	2.54 ± 0.36	1.63 ± 0.25	2.86 ± 0.47	6.55 ± 0.71	14.64 ± 0.98
	Audio2CF	6289	1988 ± 90	1.13 ± 0.24	0.69 ± 0.13	0.99 ± 0.29	2.75 ± 0.47	8.21 ± 0.80
	i-vectors	6289	2164 ± 114	1.34 ± 0.25	0.73 ± 0.12	1.32 ± 0.31	3.02 ± 0.46	7.10 ± 0.70
	Audio2Tag	6289	2946 ± 138	0.57 ± 0.15	0.35 ± 0.09	0.45 ± 0.18	1.26 ± 0.32	4.13 ± 0.56
Membership	Audio2CF + Tags + Logs	6289	**652 ± 46**	**3.73 ± 0.44**	**2.35 ± 0.28**	**4.52 ± 0.60**	**9.58 ± 0.83**	**20.43 ± 1.12**
	Audio2CF + Logs	6289	**691 ± 49**	3.33 ± 0.38	2.16 ± 0.25	4.20 ± 0.56	**9.55 ± 0.83**	**20.05 ± 1.09**
	Tags + Logs	6289	**719 ± 53**	3.47 ± 0.40	2.18 ± 0.26	**4.68 ± 0.58**	**9.45 ± 0.80**	**19.59 ± 1.12**
	Audio2CF + Tags	6289	**762 ± 59**	3.56 ± 0.43	2.19 ± 0.27	4.18 ± 0.56	**9.42 ± 0.84**	**19.58 ± 1.08**
	Tags	6289	**804 ± 54**	3.31 ± 0.40	2.05 ± 0.25	4.09 ± 0.56	8.88 ± 0.79	**18.53 ± 1.10**
	Logs	6289	**986 ± 68**	2.89 ± 0.37	1.78 ± 0.22	3.79 ± 0.52	8.00 ± 0.75	16.92 ± 1.04
	Audio2CF	6289	**1149 ± 69**	2.49 ± 0.35	1.47 ± 0.20	2.71 ± 0.46	6.44 ± 0.67	15.50 ± 1.02
	i-vectors	6289	**1441 ± 86**	2.23 ± 0.34	1.31 ± 0.19	2.31 ± 0.41	5.72 ± 0.61	12.76 ± 0.92
	Audio2Tag	6289	1841 ± 87	2.23 ± 0.36	1.33 ± 0.22	2.43 ± 0.42	5.63 ± 0.63	11.52 ± 0.88
MF	—	*4299*	***1198 ± 145***	*4.04 ± 0.56*	*2.77 ± 0.42*	*5.27 ± 0.72*	*10.38 ± 1.01*	*18.62 ± 1.29*
Hybrid MF	Logs	6289	1677 ± 138	2.79 ± 0.41	1.74 ± 0.25	3.19 ± 0.50	7.39 ± 0.75	15.88 ± 1.05
	Audio2CF	6289	2636 ± 142	1.09 ± 0.24	0.73 ± 0.18	1.06 ± 0.30	2.58 ± 0.46	7.03 ± 0.75
Neighbors	—	*4299*	*3566 ± 318*	*4.93 ± 0.64*	*3.49 ± 0.48*	***6.94 ± 0.86***	*12.11 ± 1.08*	*19.50 ± 1.32*
Popularity	—	6289	3352 ± 272	1.80 ± 0.33	1.05 ± 0.19	1.79 ± 0.36	4.14 ± 0.53	9.71 ± 0.81
Random	—	6289	7094 ± 234	0.15 ± 0.06	0.11 ± 0.06	0.15 ± 0.13	0.31 ± 0.17	0.78 ± 0.24

* The 95% bootstrap CIs for the MRR of Profiles (Logs) and Hybrid MF (Logs) do not overlap. However, this is not apparent from the table because, for readability, we only display the largest margin of the CIs.

Table 5.10: Strong generalization results for the AotM-2011 dataset. The first two columns indicate the system and feature names (if any). The third column indicates the number N of songs in the withheld continuations involved in each experiment. MF and Neighbors, only evaluated for in-set songs, are displayed in italics. The median rank is relative to 12,286 unique songs. Lower is better. For MRR, MAP and R@{10, 30, 100} higher is better. Systems clearly outperforming Hybrid MF (non-overlapping 95% CIs) are indicated in bold. MF or Neighbors clearly outperforming one another are indicated in italic bold.

AotM-2011 - strong generalization

system	feature	N	med rank	MRR [%]	MAP [%]	R@10 [%]	R@30 [%]	R@100 [%]
Profiles	—	—	—	—	—	—	—	—
Membership	Audio2CF + Tags + Logs	4473	**1150 ± 72**	1.88 ± 0.33	1.37 ± 0.25	2.53 ± 0.51	**5.91 ± 0.78**	**13.69 ± 1.13**
	Tags + Logs	4473	**1223 ± 77**	1.82 ± 0.33	1.23 ± 0.21	2.20 ± 0.48	**5.66 ± 0.74**	**13.50 ± 1.13**
	Audio2CF + Logs	4473	**1248 ± 95**	1.86 ± 0.33	1.25 ± 0.23	2.31 ± 0.48	5.41 ± 0.75	**12.64 ± 1.08**
	Audio2CF + Tags	4473	**1285 ± 89**	1.73 ± 0.31	1.26 ± 0.23	2.57 ± 0.55	5.42 ± 0.77	**12.54 ± 1.10**
	Tags	4473	**1423 ± 106**	1.79 ± 0.35	1.27 ± 0.26	1.97 ± 0.46	4.95 ± 0.73	**12.33 ± 1.07**
	Logs	4473	**1552 ± 96**	1.42 ± 0.29	0.98 ± 0.20	1.88 ± 0.47	4.18 ± 0.64	10.71 ± 1.04
	Audio2CF	4473	**2065 ± 120**	0.93 ± 0.21	0.62 ± 0.12	1.40 ± 0.39	3.18 ± 0.58	7.05 ± 0.85
	i-vectors	4473	2768 ± 185	0.93 ± 0.20	0.63 ± 0.15	1.11 ± 0.35	2.99 ± 0.58	7.45 ± 0.85
	Audio2Tag	4473	2900 ± 151	0.51 ± 0.14	0.35 ± 0.09	0.57 ± 0.26	1.29 ± 0.37	4.00 ± 0.64
MF	—	*2849*	***1428 ± 135***	*2.72 ± 0.53*	*2.12 ± 0.42*	*4.03 ± 0.81*	*8.41 ± 1.10*	*15.43 ± 1.44*
Hybrid MF	Logs	4473	2404 ± 164	1.47 ± 0.31	1.03 ± 0.23	1.90 ± 0.47	4.12 ± 0.67	8.86 ± 0.94
	Audio2CF	4473	2921 ± 164	0.74 ± 0.21	0.54 ± 0.14	0.91 ± 0.33	2.33 ± 0.52	5.60 ± 0.78
Neighbors	—	*2849*	*4502 ± 304*	***4.89 ± 0.82***	***4.26 ± 0.73***	***6.54 ± 1.03***	*10.31 ± 1.24*	*15.16 ± 1.43*
Popularity	—	4473	5143 ± 583	1.03 ± 0.27	0.67 ± 0.17	1.09 ± 0.34	3.07 ± 0.57	7.28 ± 0.85
Random	—	4473	6161 ± 188	0.13 ± 0.05	0.10 ± 0.05	0.13 ± 0.15	0.35 ± 0.22	1.23 ± 0.37

Table 5.11: Strong generalization results for the 8tracks dataset. The first two columns indicate the system and feature names (if any). The third column indicates the number N of songs in the withheld continuations involved in each experiment. MF and Neighbors, only evaluated for in-set songs, are displayed in italics. The median rank is relative to 14,552 unique songs. Lower is better. For MRR, MAP and R@{10, 30, 100} higher is better. Systems clearly outperforming Hybrid MF (non-overlapping 95% CIs) are indicated in bold. MF or Neighbors clearly outperforming one another are indicated in italic bold.

8tracks - strong generalization

system	feature	N	med rank	MRR [%]	MAP [%]	R@10 [%]	R@30 [%]	R@100 [%]
Profiles	—	—	—	—	—	—	—	—
Membership	Audio2CF + Tags + Logs	6289	**710 ± 51**	3.46 ± 0.43	2.19 ± 0.27	**4.31 ± 0.58**	**9.77 ± 0.84**	**20.40 ± 1.12**
	Audio2CF + Logs	6289	**773 ± 52**	3.28 ± 0.42	2.04 ± 0.27	3.57 ± 0.52	8.68 ± 0.78	**18.38 ± 1.07**
	Tags + Logs	6289	**792 ± 51**	3.41 ± 0.44	2.10 ± 0.26	3.69 ± 0.53	8.83 ± 0.80	**18.90 ± 1.11**
	Audio2CF + Tags	6289	**849 ± 52**	3.23 ± 0.39	2.04 ± 0.25	4.17 ± 0.57	8.58 ± 0.78	**18.94 ± 1.12**
	Tags	6289	**869 ± 73**	3.31 ± 0.42	1.99 ± 0.25	3.87 ± 0.53	7.93 ± 0.76	17.76 ± 1.05
	Logs	6289	**1038 ± 78**	2.85 ± 0.39	1.70 ± 0.22	3.10 ± 0.47	7.36 ± 0.71	15.90 ± 1.03
	Audio2CF	6289	**1317 ± 86**	2.58 ± 0.37	1.52 ± 0.22	2.72 ± 0.44	5.96 ± 0.64	14.54 ± 0.99
	i-vectors	6289	**1455 ± 91** ⋆	2.01 ± 0.29	1.22 ± 0.18	2.32 ± 0.40	5.71 ± 0.64	12.83 ± 0.94
	Audio2Tag	6289	2093 ± 125	1.91 ± 0.31	1.11 ± 0.19	2.15 ± 0.38	4.61 ± 0.57	9.68 ± 0.81
MF	—	*4298*	***997.5 ± 109.5***	*4.48 ± 0.61*	*3.06 ± 0.42*	*5.41 ± 0.75*	11.02 ± 1.05	20.32 ± 1.35
Hybrid MF	Logs	6289	1677 ± 138	2.79 ± 0.41	1.74 ± 0.26	3.19 ± 0.50	7.39 ± 0.73	*15.88 ± 1.04*
	Audio2CF	6289	2636 ± 142	1.09 ± 0.24	0.73 ± 0.18	1.06 ± 0.30	2.58 ± 0.47	7.03 ± 0.75
Neighbors	—	*4298*	*3045.5 ± 390.5*	*5.27 ± 0.68*	*3.87 ± 0.54*	***7.11 ± 0.85***	*12.76 ± 1.14*	20.43 ± 1.41
Popularity	—	6289	3293 ± 278	1.76 ± 0.32	1.04 ± 0.20	1.75 ± 0.36	4.49 ± 0.55	9.32 ± 0.78
Random	—	6289	6876 ± 158	0.14 ± 0.06	0.10 ± 0.05	0.15 ± 0.13	0.21 ± 0.14	0.73 ± 0.25

⋆ The 95% bootstrap CIs for the median rank of Membership (i-vectors) and Hybrid MF (Logs) do not overlap. However, this is not apparent from the table because, for readability, we only display the largest margin of the CIs.

Table 5.12: Results on the AotM-2011 dataset recommending songs that occurred in 4 or less training playlists using only features derived from the audio signal. The first three columns indicate the generalization setting, the system name, and the feature name (if any). The fourth column indicates the number N of songs in the withheld continuations involved in each experiment. The median rank is relative to 12,286 unique songs. Lower is better. For MRR, MAP and R@{10, 30, 100} higher is better. Systems clearly outperforming Hybrid MF (non-overlapping 95% CIs) are indicated in bold.

AotM-2011 - recommending songs that occurred in 4 or less training playlists with audio-based features only

generalization	system	feature	N	med rank	MRR [%]	MAP [%]	R@10 [%]	R@30 [%]	R@100 [%]
Weak	Profiles	Audio2CF	3411	**2130 ± 140**	0.58 ± 0.16	0.46 ± 0.13	0.61 ± 0.30	2.23 ± 0.55	6.65 ± 0.94
		i-vectors	3411	2738 ± 177	0.49 ± 0.15	0.35 ± 0.10	0.57 ± 0.29	1.33 ± 0.43	4.06 ± 0.73
		Audio2Tag	3411	3141 ± 189	0.32 ± 0.08	0.25 ± 0.07	0.29 ± 0.20	1.07 ± 0.40	3.53 ± 0.68
	Membership	Audio2CF	3411	**2264 ± 135**	0.55 ± 0.17	0.46 ± 0.16	0.68 ± 0.33	1.75 ± 0.50	5.33 ± 0.84
		Audio2Tag	3411	3106 ± 174	0.26 ± 0.05	0.21 ± 0.04	0.28 ± 0.22	0.96 ± 0.39	3.36 ± 0.69
		i-vectors	3411	3514 ± 154	0.10 ± 0.01	0.09 ± 0.01	0.00 ± 0.00	0.06 ± 0.10	0.75 ± 0.34
	Hybrid MF	Audio2CF	3411	2718 ± 205	0.75 ± 0.22	0.62 ± 0.19	1.10 ± 0.40	2.74 ± 0.63	6.12 ± 0.92
	Random	—	3411	6108 ± 224	0.10 ± 0.03	0.08 ± 0.03	0.13 ± 0.17	0.30 ± 0.24	1.10 ± 0.41
Strong	Profiles	—	—	—	—	—	—	—	—
	Membership	Audio2CF	3456	**2257.0 ± 134.5**	0.43 ± 0.08	0.35 ± 0.07	0.68 ± 0.33	1.90 ± 0.52	5.20 ± 0.83
		Audio2Tag	3456	3029 ± 218	0.25 ± 0.07	0.19 ± 0.04	0.21 ± 0.17	0.53 ± 0.25	2.82 ± 0.62
		i-vectors	3456	3578.5 ± 194.5	0.11 ± 0.01	0.09 ± 0.01	0.00 ± 0.00	0.08 ± 0.11	0.80 ± 0.34
	Hybrid MF	Audio2CF	3456	2705 ± 202	0.75 ± 0.21	0.63 ± 0.20	1.11 ± 0.41	2.74 ± 0.63	6.12 ± 0.92
	Random	—	3456	6118 ± 214	0.12 ± 0.05	0.10 ± 0.05	0.15 ± 0.17	0.43 ± 0.26	1.23 ± 0.42

Table 5.13: Results on the 8tracks dataset recommending songs that occurred in 4 or less training playlists using only features derived from the audio signal. The first three columns indicate the generalization setting, the system name, and the feature name (if any). The fourth column indicates the number N of songs in the withheld continuations involved in each experiment. The median rank is relative to 14,552 unique songs. Lower is better. For MRR, MAP and R@{10, 30, 100} higher is better. Systems clearly outperforming Hybrid MF (non-overlapping 95% CIs) are indicated in bold.

8tracks - recommending songs that occurred in 4 or less training playlists with audio-based features only

generalization	system	feature	N	med rank	MRR [%]	MAP [%]	R@10 [%]	R@30 [%]	R@100 [%]
Weak	Profiles	Audio2CF	4088	**1778 ± 152**	0.99 ± 0.22	0.71 ± 0.17	1.13 ± 0.36	2.94 ± 0.58	8.79 ± 1.01
		i-vectors	4088	2279.5 ± 145.5	0.71 ± 0.17	0.48 ± 0.10	0.86 ± 0.32	2.14 ± 0.50	5.48 ± 0.77
		Audio2Tag	4088	2727.0 ± 156.5	0.56 ± 0.17	0.38 ± 0.11	0.51 ± 0.23	1.48 ± 0.41	4.62 ± 0.72
	Membership	Audio2CF	4088	1922 ± 118	0.57 ± 0.15	0.43 ± 0.09	0.59 ± 0.27	1.96 ± 0.49	6.60 ± 0.89
		i-vectors	4088	2596.5 ± 119.0	0.32 ± 0.12	0.23 ± 0.07	0.20 ± 0.16	0.66 ± 0.27	2.35 ± 0.51
		Audio2Tag	4088	2779.5 ± 155.5	0.26 ± 0.04	0.21 ± 0.03	0.17 ± 0.13	0.93 ± 0.33	3.31 ± 0.64
	Hybrid MF	Audio2CF	4088	2180 ± 187	1.15 ± 0.28	0.84 ± 0.21	1.33 ± 0.40	3.03 ± 0.61	8.09 ± 0.94
	Random	—	4088	7168 ± 256	0.10 ± 0.04	0.07 ± 0.03	0.10 ± 0.12	0.20 ± 0.15	0.70 ± 0.27
Strong	Profiles	—	—	—	—	—	—	—	—
	Membership	Audio2CF	4134	2002 ± 136	0.58 ± 0.15	0.43 ± 0.11	0.51 ± 0.23	1.55 ± 0.43	6.03 ± 0.85
		i-vectors	4134	2522.5 ± 123.0	0.23 ± 0.05	0.17 ± 0.02	0.05 ± 0.06	0.55 ± 0.24	2.11 ± 0.50
		Audio2Tag	4134	2747.5 ± 152.5	0.35 ± 0.11	0.25 ± 0.06	0.30 ± 0.18	0.83 ± 0.32	3.59 ± 0.64
	Hybrid MF	Audio2CF	4134	2202.5 ± 190.0	1.16 ± 0.29	0.84 ± 0.21	1.35 ± 0.41	3.03 ± 0.60	8.11 ± 0.96
	Random	—	4134	6921.5 ± 220.5	0.08 ± 0.03	0.06 ± 0.02	0.02 ± 0.04	0.05 ± 0.06	0.61 ± 0.27

6 CONCLUSION

Music recommender systems have become crucial technologies to support the navigation of large music catalogs in music streaming services, on-line shops and private collections. This thesis has concentrated on the machine learning aspects of music recommendation, proposing hybrid recommender systems that integrate the strengths of collaborative and content information while taking into consideration the particularities of the music domain.

The first part of the thesis focused on the task of music artist recommendation as an approximation of the users' general music preferences over an extended period of time. I proposed a hybrid extension to a well-established matrix factorization model consisting in a co-factorization of listening and tagging histories. Incorporating the tagging histories with an appropriate weighting scheme improved the accuracy of artist recommendations in the conducted experiments.

The second part of the thesis studied the task of automated music playlist continuation and, in contrast to the first part, had a focus on local relationships in short listening sessions. I investigated fundamental playlist characteristics — namely the song context length, the song order and the song popularity — and their impact on the prediction of next-song recommendations. The conducted experiments revealed interwoven effects, but it could generally be observed that: (1) a longer song context had a positive effect on the accuracy of next-song recommendations; (2) dealing with infrequent music items was challenging for some recommender systems; (3) knowing the song order did not affect the accuracy of next-song recommendations with the considered systems and datasets. Building on these observations, I introduced two hybrid recommender systems for music playlist continuation based on neural network models. The proposed hybrid systems consider the full song context and efficiently deal with infrequent music through the incorporation of song feature vectors derived from any type of music descriptions.

The hybrid music recommender systems proposed throughout the thesis performed comparably or better than the considered baselines in the conducted off-line experiments, and further analysis revealed that the improvements were largely explained by the superior ability of the proposed systems to deal with infrequent music. Thus, the proposed systems provide means to mitigate the cold-start problem for infrequent music and can be utilized to support the discovery of music beyond the *charts* of popular music.

6.1 OPEN CHALLENGES

In the following I discuss two main limitations of the research presented in this thesis: the evaluation of music recommender systems using off-line experiments, and the obvious but ineludible impact of the chosen datasets on the final system recommendations.

6.1.1 Off-line evaluation of subjective opinions

Retrieval tasks like music identification [130] or music fingerprinting [136] have well-defined sets of relevant results for a given query. In contrast, music recommendation (or the recommendation of any items depending on the users' taste) is a much more undefined task. Even given accurate user preference profiles and rich contextual information, users can find multiple items relevant, and their final choice is likely subjective and affected by uncertainty [3, 66]. Evaluating recommender systems with off-line experiments is challenging precisely because the final judgment of the user is missing, and the assessment is only based on the comparison of system predictions to withheld historical user–item interactions. I provide examples to illustrate this challenge.

Figure 6.1 shows music playlists curated and published by users of the Art of the Mix playlist-sharing platform. As described in Chapters 4 and 5, and according to the study conducted by Cunningham, Bainbridge, and Falconer [31], it is reasonable to presume a careful compilation process in playlists from this database.

Playlists 1 and 2 are *similar*: they share four songs and contain five more songs by the same artists. The remaining songs are different. A recommender system predicting any of these remaining songs when seeded with the intersection of playlists 1 and 2 should be considered to work reasonably well, even if the system does not reproduce any of the original playlists exactly. However, missing to reproduce the playlists exactly will be penalized in an off-line experiment, and the assessed system performance will be too pessimistic [99, 114].

Playlist 3 consists of pairs of different songs which happen to have the same title. Cunningham, Bainbridge, and Falconer [31] already identified the existence of playlists satisfying artificial criteria. Even if this compilation rule is not musically motivated, Art of the Mix users provided positive comments to the playlist.[1] However, unless specifically designed to do so, a usual music recommender system would never produce such an artificial playlist.

These playlists illustrate the subjectivity involved in the music selection process and the consequent difficulty of evaluating music recommendations with off-line experiments. Trying to account for it, in Chapters 4 and 5 I proposed approaches to interpreting off-line retrieval-based experiments. I summarize them here again.

1 http://www.artofthemix.org/FindAMix/getcontents2.aspx?strMixID=94784

1 **Life is hard... and so am I**
Art of the Mix id 28016, user 'Pushkin_Sanchez'

Eels - Novocaine for the Soul
Pixies - Wave of Mutilation
Weezer - Say It Ain't So
Alice In Chains - Them Bones
○ Nirvana - Smells Like Teen Spirit
Bush - Glycerine
△ Smashing Pumpkins - Disarm
Live - Lightning Crashes
○ Green Day - Basket Case
Radiohead - Creep
○ Faith No More - Epic
△ Pearl Jam - Jeremy
△ Metallica - Nothing Else Matters
○ Stone Temple Pilots - Plush
△ Soundgarden - Black Hole Sun
○ Spacehog - In the Meantime
△ Red Hot Chili Peppers - Under the Bridge

2 **90's nostalgia (the K-tel mix)**
Art of the Mix id 8712, user 'AMPHEAD1'

○ Faith No More - Epic
○ Nirvana - Smells Like Teen Spirit
△ Pearl Jam - Evenflow
△ Soundgarden - Rusty Cage
△ Red Hot Chili Peppers - Give It Away
Collective Soul - Gel
△ The Beastie Boys - Sabotage
○ Stone Temple Pilots - Plush
△ Metallica - Enter Sandman
○ Green Day - Basket Case
Lenny Kravitz - Are You Gonna Go My Way
Beck - Loser
○ Spacehog - In the Meantime
White Zombie - More Human Than Human
Blur - Song 2
Harvey Danger - Flagpole Sitta
Monster Magnet - Spacelord
Foo Fighters - Everlong
Kid Rock - Bawitabada
Offspring - Pretty Fly (for a White Guy)
Limp Bizkit - Nookie

3 **Same name, different song**
Art of the Mix id 94784, user 'RachelFaith'

Phantom Planet - California
Rufus Wainwright - California
Joe Jackson - Another World
The Roches - Another World
The Martinis - Free
Train - Free
Phish - Bliss
Tori Amos - Bliss
△ The Beastie Boys - Girls
Prodigy - Girls
Frankie Goes to Hollywood - Relax
G. Love & Special Sauce - Relax
Martin Sexton - Hallelujah
Rufus Wainwright - Hallelujah
○ Green Day - Basket Case
Warren Zevon - Basket Case
Bluse Traveler - Freedom
Paul McCartney - Freedom
Poe - Today
△ Smashing Pumpkins - Today
Ben Harper - Faded
Howie Day - Faded
G. Love & Special Sauce - Love
△ Smashing Pumpkins - Love
Jack Johnson - Gone
Ben Folds - Gone

Figure 6.1: Hand-curated playlists from the Art of the Mix platform. Songs occurring in more than one playlist are indicated with a circle. Artists occurring in more than one playlist (with different songs) are indicated with a triangle.

Inspecting the full list of recommendations

Off-line retrieval-based experiments usually focus on the amount of relevant music items that a recommender system can rank in the top positions of a list of recommendations. While top positions are the most relevant for end users, off-line experiments do not have access to actual user assessments. In fact, focusing only on top positions can provide inaccurate performance estimates. For example, in Chapter 5, the collaborative systems MF and Neighbors achieved comparable R@100 in a playlist continuation experiment (Table 5.8). However, if we inspect the complete recall curves of both systems, the global performance of MF is clearly superior (Figure 5.6). Therefore, I propose to utilize metrics that consider the full list of recommendations, such as the expected percentile rank (Chapter 3), the rank distribution (Chapter 4), or the median rank (Chapters 4 and 5). This can provide fairer estimates of the overall system performance.

The effects of song popularity

The music domain is strongly affected by popularity effects [21]. In most music collections a few items are frequently listened to, while the majority of them are rarely listened to (Figure 4.3, Tables 4.2, 5.1). Music recommender systems respond differently to such biases. For example, in a playlist continuation experiment in Chapter 4, the performance of the RNN-based system was mostly unaffected by the popularity of the withheld next songs, but the playlist-neighbors system was only competitive when the withheld next songs were popular (Figure 4.4). These behaviors must be taken into consideration and related to the final use case. Users interested in music discovery may prefer the recommendations of the RNN-based system. On the other hand, the higher computational complexity of the RNN-based system may be unnecessary for users interested in popular music.

6.1.2 Impact of data selection and preparation

The music recommender systems proposed in the thesis are based on machine learning models trained using datasets of user–item interactions and music descriptions. In this way, they are expected to automatically identify music taste patterns from data. This contrasts to recommender systems following hand-designed, arbitrary rules, such as recommending items with similar acoustic features, or from the same genre. I generally advocate the purely data-driven approach, but it must be noted that arbitrary design choices may actually be transferred (voluntarily or not) to the selection and preparation of the training dataset. I provide two examples to illustrate this limitation.

Chapter 3, as other works [17, 24, 63, 64, 97, 167], exploits listening histories from the Last.fm music platform. Unfortunately, these

histories can be influenced by recommendations of the very platform. Consequently, recommender systems trained on such histories can not distinguish the actual user preferences from patterns caused by previous automatic recommendations [23].

Chapters 4 and 5 improve on this situation by exploiting collections of hand-curated music playlists, which are not influenced by automatic recommendations. However, I decided to apply filters to ensure the quality of the considered playlists and of the songs therein. The songs occurring in both playlist collections were linked to the MSD to resolve spelling duplicates and thus identify them accurately. The songs not present in the MSD were discarded. The songs for which some type of description was not available were further discarded in Chapter 5 for the sake of comparability. Furthermore, I considered only playlists with a minimum length and with enough songs from different artists in order to discard artist- and album-themed playlists. The goal of filtering was to ensure the quality of the data, but in any case this was a design choice with an impact on the patterns that music recommender systems would be able to identify.

Data filtering or preprocessing need to be considered when comparing results from different works which apparently use the same datasets. In my own work, Chapters 4 and 5 filter the playlist collections differently (compare Tables 4.2 and 5.1). This is because the former study does not require the availability of song descriptions, but the latter does. Similarly, other works exploiting the same or related collections of hand-curated music playlists did not report to have applied such filters [17, 53, 63, 99, 100].

6.2 OUTLOOK

Finally, and despite the important advances in music understanding, recommender systems, and their interplay, I outline interesting future research directions where there is still much room for improvement.

EVALUATION As already discussed, the assessment of recommender systems is one of the most critical research directions, and it is necessary to develop off-line evaluation protocols that relate more closely to the final user satisfaction. This topic attracts the attention of researchers, as demonstrated by the recent organization of the specialized REVEAL Workshop [67]. Research efforts include the better understanding of off-line metrics and their biases [20, 62, 146], the development of beyond-accuracy metrics [70, 159], and the proposal of alternative off-line evaluation paradigms such as reinforcement learning [82, 122, 166]. The evaluation of recommender systems would also benefit from a better understanding and a wider adoption of small- and medium-scale user experiments [121, Chapter 9]. Finally, a

tighter collaboration between academy and industrial platforms able to conduct large-scale on-line experiments would be greatly advantageous.

USER NEEDS AND INTERACTION Recommending music is complex, especially due to the changeable needs of the users and their subjective assessment of the received recommendations. Research in music recommender systems has mostly concentrated on automatically inferring the user music preferences and needs, but the truth is that current systems only partially manage to do so. This gap could be reduced by letting the users explicitly indicate their needs. Conversational recommender systems provide contributions in this direction, for example, letting the users critique recommendations [160], suggesting query revisions [13], or providing interfaces that let the users refine their recommendations [141], and they should be further investigated. In fact, conversational systems are becoming central also beyond recommender systems, with well-known examples such as Amazon's Alexa,[2] the Google Assistant,[3] or Yandex's Alice.[4]

METHODS The algorithmic and machine learning aspects of recommender systems have received and continue to receive much attention. The technologies powering recommender systems have evolved in parallel to the advances in machine learning, such as matrix factorization [60], support vector machines [118], learning to rank [133], or deep learning [7]. Thus, the recent explosion of research in machine learning may also foster original, brave ideas beyond improvements in predictive accuracy, such as the combination of recommender systems and generative models to suggest items that may not yet exist [161], or the explanation of recommendations using model interpretability techniques [90]. Scalability must remain a priority, especially given the ever-increasing volume of music available and the fact that deep learning models can easily become computationally demanding.

2 https://developer.amazon.com/alexa
3 https://assistant.google.com
4 https://alice.yandex.ru

BIBLIOGRAPHY

[1] Gediminas Adomavicius and Alexander Tuzhilin. "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions." In: *IEEE Transactions on Knowledge and Data Engineering* 17.6 (2005), pp. 734–749.

[2] Natalie Aizenberg, Yehuda Koren, and Oren Somekh. "Build your own music recommender by modeling internet radio streams." In: *Proc. WWW*. Lyon, France, 2012, pp. 1–10.

[3] Xavier Amatriain, Josep M. Pujol, and Nuria Oliver. "I like it... i like it not: Evaluating user ratings noise in recommender systems." In: *Proc. UMAP*. Trento, Italy, 2009, pp. 247–258.

[4] Kristina Andersen and Peter Knees. "Conversations with expert users in music retrieval and research challenges for creative MIR." In: *Proc. ISMIR*. New York, NY, USA, 2016, pp. 122–128.

[5] Chris Anderson. *The long tail: How endless choice is creating unlimited demand*. Random House, 2007.

[6] Jean-Julien Aucouturier and François Pachet. "Improving timbre similarity: How high is the sky." In: *Journal of Negative Results in Speech and Audio Sciences* 1.1 (2004), pp. 1–13.

[7] Zeynep Batmaz, Ali Yurekli, Alper Bilge, and Cihan Kaleli. "A review on deep learning for recommender systems: Challenges and remedies." In: *Artificial Intelligence Review* (2018).

[8] Nicholas J. Belkin and W. Bruce Croft. "Information filtering and information retrieval: Two sides of the same coin?" In: *Communications of the ACM* 35.12 (1992), pp. 29–38.

[9] Robert M. Bell and Yehuda Koren. "Scalable collaborative filtering with jointly derived neighborhood interpolation weights." In: *Proc. ICDM*. Omaha, NE USA, 2007, pp. 43–52.

[10] James Bennett and Stan Lanning. "The Netflix Prize." In: *Proc. KDDCup*. San Jose, CA, USA, 2007, p. 35.

[11] James Bergstra and Yoshua Bengio. "Random search for hyper-parameter optimization." In: *Journal of Machine Learning Research* 13.1 (2012), pp. 281–305.

[12] Thierry Bertin-Mahieux, Daniel PW Ellis, Brian Whitman, and Paul Lamere. "The million song dataset." In: *Proc. ISMIR*. Miami, FL, USA, 2011, pp. 591–596.

[13] Henry Blanco and Francesco Ricci. "Acquiring user profiles from implicit feedback in a conversational recommender system." In: *Proc. RecSys*. Hong Kong, China, 2013, pp. 307–310.

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

[15] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez. "Recommender systems survey." In: *Knowledge-Based Systems* 46 (2013), pp. 109–132.

[16] Dmitry Bogdanov and Perfecto Herrera. "Taking advantage of editorial metadata to recommend music." In: *Proc. CMMR*. London, UK, 2012, pp. 618–632.

[17] Geoffray Bonnin and Dietmar Jannach. "Automated generation of music playlists: Survey and experiments." In: *ACM Computing Surveys* 47.2 (2014), pp. 1–35.

[18] Léon Bottou, Frank E. Curtis, and Jorge Nocedal. "Optimization methods for large-scale machine learning." In: *arXiv preprint arXiv:1606.04838* (2018).

[19] Robin Burke. "Hybrid recommender systems: Survey and experiments." In: *User Modeling and User-Adapted Interaction* 12.4 (2002), pp. 331–370.

[20] Pablo Castells and Rocío Cañamares. "Characterization of fair experiments for recommender system evaluation - A formal analysis." In: *REVEAL Workshop at RecSys*. Vancouver, British Columbia, Canada, 2018.

[21] Òscar Celma. *Music recommendation and discovery*. Springer, 2010.

[22] Òscar Celma, Perfecto Herrera, and Xavier Serra. "Bridging the music semantic gap." In: *Proc. ESWC Workshop on Mastering the Gap: From Information Extraction to Semantic Representation*. Budva, Montenegro, 2006.

[23] Allison J. B. Chaney, Brandon M. Stewart, and Barbara E. Engelhardt. "How algorithmic confounding in recommendation systems increases homogeneity and decreases utility." In: *Proc. RecSys*. Vancouver, British Columbia, Canada, 2018, pp. 224–232.

[24] Chih-Ming Chen, Ming-Feng Tsai, Yu-Ching Lin, and Yi-Hsuan Yang. "Query-based music recommendations via preference embedding." In: *Proc. RecSys*. Boston, MA, USA, 2016, 79–82.

[25] Shuo Chen, Josh L. Moore, Douglas Turnbull, and Thorsten Joachims. "Playlist prediction via metric embedding." In: *Proc. SIGKDD*. Beijing, China, 2012, pp. 714–722.

[26] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." In: *arXiv preprint arXiv:1406.1078* (2014).

[27] Keunwoo Choi, George Fazekas, and Mark Sandler. "Automatic tagging using deep convolutional neural networks." In: *Proc. ISMIR*. New York, NY, USA, 2016.

[28] Tom Collins, Peter Knees, and Christian Coulon. "MIR web interface for shaping musical creativity." In: *Extended abstracts for the Late-Breaking Demo Session of ISMIR*. New York, NY, USA, 2016.

[29] Dan Cosley, Shyong K. Lam, Istvan Albert, Joseph A. Konstan, and John Riedl. "Is seeing believing?: How recommender system interfaces affect users' opinions." In: *Proc. CHI*. Ft. Lauderdale, FL, USA, 2003, pp. 585–592.

[30] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. "Performance of recommender algorithms on top-N recommendation tasks." In: *Proc. RecSys*. Barcelona, Spain, 2010, pp. 39–46.

[31] Sally Jo Cunningham, David Bainbridge, and Annette Falconer. ""More of an art than a science": Supporting the creation of playlists and mixes." In: *Proc. ISMIR*. Victoria, British Columbia, Canada, 2006.

[32] Najim Dehak, Patrick J Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet. "Front-end factor analysis for speaker verification." In: *IEEE Transactions on Audio, Speech, and Language Processing* 19.4 (2011), pp. 788–798.

[33] Thomas J. DiCiccio and Bradley Efron. "Bootstrap confidence intervals." In: *Statistical Science* (1996), pp. 189–212.

[34] Sander Dieleman et al. *Lasagne: First release.* 2015. URL: http://dx.doi.org/10.5281/zenodo.27878.

[35] Matthias Dorfer, Jan Schlüter, Andreu Vall, Filip Korzeniowski, and Gerhard Widmer. "End-to-end cross-modality retrieval with CCA projections and pairwise ranking loss." In: *International Journal of Multimedia Information Retrieval* 7.2 (2018), pp. 117–128.

[36] Gideon Dror, Noam Koenigstein, and Yehuda Koren. "Yahoo! music recommendations: modeling music ratings with temporal dynamics and item taxonomy." In: *Proc. RecSys*. Chicago, IL, USA, 2011, pp. 165–172.

[37] Gideon Dror, Noam Koenigstein, Yehuda Koren, and Markus Weimer. "The Yahoo! Music Dataset and KDD-Cup'11." In: *Proc. KDD-Cup 2011 competition*. San Diego, CA, USA, 2012, pp. 8–18.

[38] John Duchi, Elad Hazan, and Yoram Singer. "Adaptive subgradient methods for online learning and stochastic optimization." In: *Journal of Machine Learning Research* 12 (2011), pp. 2121–2159.

[39] Hamid Eghbal-zadeh, Markus Schedl, and Gerhard Widmer. "Timbral modeling for music artist recognition using i-vectors." In: *Proc. EUSIPCO*. Nice, France, 2015, pp. 1286–1290.

[40] Hamid Eghbal-zadeh, Bernhard Lehner, Markus Schedl, and Gerhard Widmer. "I-vectors for timbre-based music similarity and music artist classification." In: *Proc. ISMIR*. Málaga, Spain, 2015, pp. 554–560.

[41] Yi Fang and Luo Si. "Matrix co-factorization for recommendation with rich side information and implicit feedback." In: *Proc. HETREC*. Chicago, IL, USA, 2011, pp. 65–69.

[42] Katayoun Farrahi, Markus Schedl, Andreu Vall, David Hauger, and Marko Tkalčič. "Impact of listening behavior on music recommendation." In: *Proc. ISMIR*. Taipei, Taiwan, 2014, 483–488.

[43] Bruce Ferwerda, Andreu Vall, Marko Tkalčič, and Markus Schedl. "Exploring music diversity needs across countries." In: *Proc. UMAP*. Halifax, Nova Scotia, Canada, 2016, pp. 287–288.

[44] Bruce Ferwerda, Mark P. Graus, Andreu Vall, Marko Tkalčič, and Markus Schedl. "The influence of users' personality traits on satisfaction and attractiveness of diversified recommendation lists." In: *EMPIRE Workshop at RecSys*. Boston, MA, USA, 2016, pp. 43–47.

[45] Bruce Ferwerda, Mark P. Graus, Andreu Vall, Marko Tkalčič, and Markus Schedl. "How item discovery enabled by diversity leads to increased recommendation list attractiveness." In: *Proc. SAC*. Marrakech, Morocco, 2017, pp. 1693–1696.

[46] Ben Fields, Christophe Rhodes, and Mark d'Inverno. "Using song social tags and topic models to describe and compare playlists." In: *Workshop on Music Recommendation and Discovery at RecSys*. Barcelona, Spain, 2010.

[47] Arthur Flexer, Dominik Schnitzer, Martin Gasser, and Gerhard Widmer. "Playlist generation using start and end songs." In: *Proc. ISMIR*. Philadelphia, PA, USA, 2008, pp. 173–178.

[48] Frederic Font. "Tag recommendation using folksonomy information for online sound sharing platforms." PhD thesis. Barcelona, Spain, 2015.

[49] Ben Frederickson. *Fast Python collaborative filtering for implicit feedback datasets*. Original date: 2016-04-17. URL: https://github.com/benfred/implicit (visited on 12/12/2018).

[50] K. Ruben Gabriel and S. Zamir. "Lower rank approximation of matrices by least squares with any choice of weights." In: *Technometrics* 21.4 (1979), p. 489.

[51] Stephen J. Green, Paul Lamere, Jeffrey Alexander, François Maillet, Susanna Kirk, Jessica Holt, Jackie Bourque, and Xiao-Wen Mak. "Generating transparent, steerable recommendations from textual descriptions of items." In: *Proc. RecSys*. New York, NY, USA, 2009, pp. 281–284.

[52] Uri Hanani, Bracha Shapira, and Peretz Shoval. "Information filtering: Overview of issues, research and systems." In: *User Modeling and User-Adapted Interaction* 11.3 (2001), pp. 203–259.

[53] Negar Hariri, Bamshad Mobasher, and Robin Burke. "Context-aware music recommendation based on latent topic sequential patterns." In: *Proc. RecSys*. Dublin, Ireland, 2012, pp. 131–138.

[54] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning*. Springer series in statistics, 2008.

[55] Ruining He and Julian McAuley. "VBPR: Visual Bayesian personalized ranking from implicit feedback." In: *Proc. AAAI*. Phoenix, AR, USA, 2016, pp. 144–150.

[56] Balázs Hidasi, Massimo Quadrana, Alexandros Karatzoglou, and Domonkos Tikk. "Parallel Recurrent Neural Network Architectures for Feature-rich Session-based Recommendations." In: *Proc. RecSys*. Boston, MA, USA, 2016, pp. 241–248.

[57] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. "Session-based recommendations with recurrent neural networks." In: *Proc. ICLR*. 2016.

[58] Matthew D. Hoffman, David M. Blei, and Perry R. Cook. "Easy as CBA: A simple probabilistic model for tagging music." In: *Proc. ISMIR*. Kobe, Japan, 2009, pp. 369–374.

[59] Thomas Hofmann. "Latent semantic models for collaborative filtering." In: *ACM Transactions on Information Systems* 22.1 (2004), pp. 89–115.

[60] Yifan Hu, Yehuda Koren, and Chris Volinsky. "Collaborative filtering for implicit feedback datasets." In: *Proc. ICDM*. Pisa, Italy, 2008, pp. 263–272.

[61] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." In: *arXiv preprint arXiv:1502.03167* (2015).

[62] Dietmar Jannach, Iman Kamehkhosh, and Geoffray Bonnin. "Biases in automated music playlist generation: A comparison of next-track recommending techniques." In: *Proc. UMAP*. Halifax, Nova Scotia, Canada, 2016, pp. 281–285.

[63] Dietmar Jannach, Lukas Lerche, and Iman Kamehkhosh. "Beyond "hitting the hits": Generating coherent music playlist continuations with the right tracks." In: *Proc. RecSys*. Vienna, Austria, 2015, pp. 187–194.

[64] Dietmar Jannach and Malte Ludewig. "When recurrent neural networks meet the neighborhood for session-based recommendation." In: *Proc. RecSys*. Como, Italy, 2017, pp. 306–310.

[65] Andreas Jansson, Colin Raffel, and Tillman Weyde. "This is my jam data dump." In: *Proc. ISMIR*. Málaga, Spain, 2015.

[66] Kevin Jasberg and Sergej Sizov. "Human uncertainty and ranking error: Fallacies in metric-based evaluation of recommender systems." In: *Proc. SAC*. Pau, France, 2018.

[67] Thorsten Joachims, Adith Swaminathan, Yves Raimond, Olivier Koch, and Flavian Vasile. "REVEAL 2018: Offline evaluation for recommender systems." In: *Proc. RecSys*. Vancouver, British Columbia, Canada, 2018, pp. 514–515.

[68] Christopher C. Johnson. "Logistic Matrix Factorization for Implicit Feedback Data." In: *Distributed Machine Learning and Matrix Computations Workshop at NIPS*. 2014.

[69] Iman Kamehkhosh and Dietmar Jannach. "User perception of next-track music recommendations." In: *Proc. UMAP*. Bratislava, Slovakia, 2017, pp. 113–121.

[70] Marius Kaminskas and Derek Bridge. "Diversity, serendipity, novelty, and coverage: A survey and empirical analysis of beyond-accuracy objectives in recommender systems." In: *ACM Transactions on Interactive Intelligent Systems* 7.1 (2016), pp. 1–42.

[71] Diederik P. Kingma and Jimmy Ba. "Adam: A method for stochastic optimization." In: *Proc. ICLR*. San Diego, CA, USA, 2015.

[72] Peter Knees. "Text-based description of music for indexing, retrieval, and browsing." PhD thesis. Linz: Johannes Kepler University, 2010.

[73] Peter Knees and Kristina Andersen. "Searching for audio by sketching mental images of sound: A brave new idea for audio retrieval in creative music production." In: *Proc. ICMR*. New York, NY, USA, 2016, pp. 95–102.

[74] Peter Knees, Tim Pohle, Markus Schedl, and Gerhard Widmer. "Combining audio-based similarity with web-based data to accelerate automatic music playlist generation." In: *Proc. International Workshop on Multimedia IR*. Santa Barbara, CA USA, 2006, pp. 147–154.

[75] Yehuda Koren. "Factorization meets the neighborhood: a multifaceted collaborative filtering model." In: *Proc. SIGKDD*. Las Vegas, NV, USA, 2008, pp. 426–434.

[76] Yehuda Koren, Robert Bell, and Chris Volinsky. "Matrix factorization techniques for recommender systems." In: *Computer* 42.8 (2009), pp. 30–37.

[77] Amanda E. Krause and Adrian C. North. "Contextualized music listening: Playlists and the Mehrabian and Russell model." In: *Psychology of Well-Being* 4.1 (2014), p. 22.

[78] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." In: *Proc. NIPS*. 2012, pp. 1097–1105.

[79] Alejandro Lago, Victor Martínez-de-Albéniz, Philip Moscoso, and Andreu Vall. "The role of quick response in accelerating sales of fashion goods." In: *Analytical modeling research in fashion business*. Springer Series in Fashion Business. Springer, 2016, pp. 51–78.

[80] Jin Ha Lee, Bobby Bare, and Gary Meek. "How similar is too similar?: Exploring users' perceptions of similarity in playlist evaluation." In: *Proc. ISMIR*. Miami, FL, USA, 2011, 109–114.

[81] M. Levy and M. Sandler. "Music information retrieval using social tags and audio." In: *IEEE Transactions on Multimedia* 11.3 (2009), pp. 383–395.

[82] Lihong Li, Wei Chu, John Langford, and Xuanhui Wang. "Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms." In: *Proc. WSDM*. Hong Kong, China, 2011, pp. 297–306.

[83] Yanen Li, Jia Hu, ChengXiang Zhai, and Ye Chen. "Improving one-class collaborative filtering by incorporating rich user information." In: *Proc. CIKM*. Toronto, Ontario, Canada, 2010, pp. 959–968.

[84] Zhizhong Li and Derek Hoiem. "Learning without forgetting." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2017).

[85] Dawen Liang, Minshu Zhan, and Daniel P.W. Ellis. "Content-aware collaborative music recommendation using pre-trained neural networks." In: *Proc. ISMIR*. Málaga, Spain, 2015, 295–301.

[86] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. "Variational autoencoders for collaborative filtering." In: *Proc. WWW*. Lyon, France, 2018.

[87] Elad Liebman, Maytal Saar-Tsechansky, and Peter Stone. "DJ-MC: A reinforcement-learning agent for music playlist recommendation." In: *Proc. AAMAS*. Istanbul, Turkey, 2015, 591–599.

[88] Min Lin, Qiang Chen, and Shuicheng Yan. "Network in network." In: *arXiv preprint arXiv:1312.4400* (2013).

[89] Greg Linden, Brent Smith, and Jeremy York. "Amazon. com recommendations: Item-to-item collaborative filtering." In: *IEEE Internet Computing* 7.1 (2003), pp. 76–80.

[90] Zachary C. Lipton. "The mythos of model interpretability." In: *arXiv:1606.03490* (2016).

[91] Zachary C. Lipton and John Berkowitz. "A critical review of recurrent neural networks for sequence learning." In: *arXiv preprint arXiv:1506.00019* (2015).

[92] Beth Logan. "Mel frequency cepstral coefficients for music modeling." In: *Proc. International Symposium on Music Information Retrieval*. Plymouth, MA, USA, 2000.

[93] Beth Logan. "Content-based playlist generation: Exploratory experiments." In: *Proc. ISMIR*. Paris, France, 2002.

[94] Hao Ma, Tom Chao Zhou, Michael R. Lyu, and Irwin King. "Improving recommender systems by incorporating social contextual information." In: *ACM Transactions on Information Systems* 29.2 (2011), pp. 1–23.

[95] François Maillet, Douglas Eck, Guillaume Desjardins, Paul Lamere, and others. "Steerable playlist generation by learning song similarity from radio station playlists." In: *Proc. ISMIR*. Kobe, Japan, 2009, pp. 345–350.

[96] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *An introduction to information retrieval*. Cambridge University Press, 2009.

[97] Brian McFee, Luke Barrington, and Gert RG Lanckriet. "Learning similarity from collaborative filters." In: *Proc. ISMIR*. Utrecht, Netherlands, 2010, pp. 345–350.

[98] Brian McFee and Gert RG Lanckriet. "Heterogeneous embedding for subjective artist similarity." In: *Proc. ISMIR*. Kobe, Japan, 2009, pp. 513–518.

[99] Brian McFee and Gert RG Lanckriet. "The natural language of playlists." In: *Proc. ISMIR*. Miami, FL, USA, 2011, pp. 537–542.

[100] Brian McFee and Gert RG Lanckriet. "Hypergraph models of playlist dialects." In: *Proc. ISMIR*. Porto, Portugal, 2012, 343–348.

[101] Brian McFee, Thierry Bertin-Mahieux, Daniel P.W. Ellis, and Gert R.G. Lanckriet. "The million song dataset challenge." In: *Proc. WWW Companion*. Lyon, France, 2012, pp. 909–909.

[102] Sean M. McNee, John Riedl, and Joseph A. Konstan. "Being accurate is not enough: How accuracy metrics have hurt recommender systems." In: *Proc. CHI'06 Extended Abstracts*. Montréal, Québec, Canada, 2006, pp. 1097–1101.

[103] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. "Distributed representations of words and phrases and their compositionality." In: *Proc. NIPS*. 2013, 3111–3119.

[104] Andriy Mnih and Ruslan Salakhutdinov. "Probabilistic matrix factorization." In: *Proc. NIPS*. 2007, pp. 1257–1264.

[105] Raymond J. Mooney and Loriene Roy. "Content-based book recommending using learning for text categorization." In: *Proc. DL*. San Antonio, TX, U.S., 2000, pp. 195–204.

[106] Meinard Müller. *Fundamentals of music processing: Audio, analysis, algorithms, applications*. Springer, 2015.

[107] Yurii Nesterov. "A method of solving a convex programming problem with convergence rate O(1/k2)." In: *Soviet Mathematics Doklady* 27.2 (1983), pp. 372–376.

[108] Xia Ning and George Karypis. "Slim: Sparse linear methods for top-N recommender systems." In: *Proc. ICDM*. Vancouver, BC, Canada, 2011, pp. 497–506.

[109] Aaron van den Oord, Sander Dieleman, and Benjamin Schrauwen. "Deep content-based music recommendation." In: *Proc. NIPS*. 2013, pp. 2643–2651.

[110] Sergio Oramas, Oriol Nieto, Mohamed Sordo, and Xavier Serra. "A deep multimodal approach for cold-start music recommendation." In: *Proc. DLRS Workshop at RecSys*. Como, Italy, 2017, pp. 32–37.

[111] Elias Pampalk, Tim Pohle, and Gerhard Widmer. "Dynamic playlist generation based on skipping behavior." In: *Proc. ISMIR*. Vol. 5. London, UK, 2005, pp. 634–637.

[112] Rong Pan, Yunhong Zhou, Bin Cao, Nathan Nan Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. "One-class collaborative filtering." In: *Proc. ICDM*. Pisa, Italy, 2008, pp. 502–511.

[113] Michael J. Pazzani. "A framework for collaborative, content-based and demographic filtering." In: *Artificial Intelligence Review* 13.5-6 (1999), pp. 393–408.

[114] John C. Platt, Christopher JC Burges, Steven Swenson, Christopher Weare, and Alice Zheng. "Learning a Gaussian process prior for automatically generating music playlists." In: *Proc. NIPS*. 2002, pp. 1425–1432.

[115] Tim Pohle. "Automatic characterization of music for intuitive retrieval." PhD thesis. Linz, Austria: Johannes Kepler University, 2009.

[116] Tim Pohle, Elias Pampalk, and Gerhard Widmer. "Generating similarity-based playlists using traveling salesman algorithms." In: *Proc. DAFx*. Madrid, Spain, 2005, pp. 220–225.

[117] Lawrence R Rabiner and Biing-Hwang Juang. *Fundamentals of speech recognition*. PTR Prentice Hall, 1993.

[118] Steffen Rendle. "Factorization machines." In: *Proc. ICDM*. Sidney, Australia, 2010, pp. 995–1000.

[119] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. "Factorizing personalized markov chains for next-basket recommendation." In: *Proc. WWW*. Raleigh, NC, USA, 2010, pp. 811–820.

[120] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. "BPR: Bayesian personalized ranking from implicit feedback." In: *Proc. UAI*. 2009, pp. 452–461.

[121] Francesco Ricci, Lior Rokach, and Bracha Shapira. *Recommender systems handbook*. 2nd. Springer US, 2015.

[122] David Rohde, Stephen Bonner, Travis Dunlop, Flavian Vasile, and Alexandros Karatzoglou. "RecoGym: A reinforcement learning environment for the problem of product recommendation in online advertising." In: *REVEAL Workshop at RecSys*. Vancouver, British Columbia, Canada, 2018.

[123] Bernard Rous. "Major update to ACM's Computing Classification System." In: *Communications of the ACM* 55.11 (2012), pp. 12–12.

[124] Ruslan Salakhutdinov and Andriy Mnih. "Bayesian probabilistic matrix factorization using Markov chain Monte Carlo." In: *Proc. ICML*. Helsinki, Finland, 2008, pp. 880–887.

[125] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. "Restricted Boltzmann machines for collaborative filtering." In: *Proc. ICML*. Corvallis, OR, USA, 2007, pp. 791–798.

[126] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. "Item-based collaborative filtering recommendation algorithms." In: *Proc. WWW*. Hong Kong, China, 2001, 285–295.

[127] Markus Schedl, Andreu Vall, and Katayoun Farrahi. "User geospatial context for music recommendation in microblogs." In: *Proc. SIGIR*. Gold Coast, QLD, Australia, 2014, pp. 987–990.

[128] Markus Schedl, Gerhard Widmer, Peter Knees, and Tim Pohle. "A music information system automatically generated via Web content mining techniques." In: *Information Processing & Management* 47.3 (2010), pp. 426–439.

[129] Markus Schedl, Hamed Zamani, Ching-Wei Chen, Yashar Deldjoo, and Mehdi Elahi. "Current challenges and visions in music recommender systems research." In: *International Journal of Multimedia Information Retrieval* 7.2 (2018), pp. 95–116.

[130] J. Serra, E. Gomez, P. Herrera, and X. Serra. "Chroma binary similarity and local alignment applied to cover song identification." In: *IEEE Transactions on Audio, Speech, and Language Processing* 16.6 (2008), pp. 1138–1151.

[131] Guy Shani, David Heckerman, and Ronen I. Brafman. "An MDP-based recommender system." In: *Journal of Machine Learning* 6 (2005), pp. 1265–1295.

[132] Upendra Shardanand and Pattie Maes. "Social information filtering: Algorithms for automating "word of mouth"." In: *Proc. CHI*. Denver, CO, USA, 1995, pp. 210–217.

[133] Yue Shi, Martha Larson, and Alan Hanjalic. "List-wise learning to rank with matrix factorization for collaborative filtering." In: *Proc. RecSys*. Barcelona, Spain, 2010, p. 269.

[134] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." In: *Proc. ICLR* (2014).

[135] Vikas Sindhwani, Serhat S. Bucak, Jianying Hu, and Aleksandra Mojsilovic. "One-class matrix completion with low-density factorizations." In: *Proc. ICDM*. Sidney, Australia, 2010, 1055–1060.

[136] Reinhard Sonnleitner and Gerhard Widmer. "Robust quad-based audio fingerprinting." In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 24.3 (2016), pp. 409–421.

[137] Nathan Srebro, Jason DM Rennie, and Tommi S. Jaakkola. "Maximum-margin matrix factorization." In: *Proc. NIPS*. Vol. 17. 2004, pp. 1329–1336.

[138] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: A simple way to prevent neural networks from overfitting." In: *Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.

[139] Kirsten Swearingen and Rashmi Sinha. "Beyond algorithms: An HCI perspective on recommender systems." In: *SIGIR Workshop on Recommender Systems*. Vol. 13. 2001, pp. 1–11.

[140] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going deeper with convolutions." In: *Proc. CVPR*. Boston, MA, USA, 2015.

[141] Taavi T. Taijala, Martijn C. Willemsen, and Joseph A. Konstan. "Movieexplorer: Building an interactive exploration tool from ratings and latent taste spaces." In: *Proc. SAC*. Pau, France, 2018, pp. 1383–1392.

[142] Yong Kiam Tan, Xinxing Xu, and Yong Liu. "Improved recurrent neural networks for session-based recommendations." In: *Proc. DLRS Workshop at RecSys*. Boston, MA, USA, 2016, 17–22.

[143] Theano Development Team. "Theano: A Python framework for fast computation of mathematical expressions." In: *arXiv e-prints* abs/1605.02688 (2016).

[144] Douglas Turnbull, Luke Barrington, and Gert Lanckriet. "Five approaches to collecting tags for music." In: *Proc. ISMIR*. Vol. 8. Philadelphia, PA, USA, 2008, pp. 225–230.

[145] Douglas Turnbull, Luke Barrington, Gert Lanckriet, and Mehrdad Yazdani. "Combining audio content and social context for semantic music discovery." In: *Proc. SIGIR*. Boston, MA, USA, 2009, pp. 387–394.

[146] Daniel Valcarce, Alejandro Bellogín, Javier Parapar, and Pablo Castells. "On the robustness and discriminative power of information retrieval metrics for top-N recommendation." In: *Proc. RecSys*. Vancouver, British Columbia, Canada, 2018, 260–268.

[147] Andreu Vall. *Matrix factorization models with social tags for music recommendation*. Original date: 2015-11-19. URL: https://github.com/andreuvall/WeightedTags-MF (visited on 12/12/2018).

[148] Andreu Vall. "Listener-inspired automated music playlist generation." In: *Proc. RecSys*. Vienna, Austria, 2015, pp. 387–390.

[149] Andreu Vall and Matthias Dorfer. *Hybrid recommender systems for music playlist continuation*. Original date: 2017-04-20. URL: https://github.com/andreuvall/HybridPlaylistContinuation (visited on 12/12/2018).

[150] Andreu Vall and Gerhard Widmer. "Machine learning approaches to hybrid music recommender systems." In: *Proc. ECML PKDD*. Dublin, Ireland, 2018.

[151] Andreu Vall, Marcin Skowron, Peter Knees, and Markus Schedl. "Improving music recommendations with a weighted factorization of the tagging activity." In: *Proc. ISMIR*. Málaga, Spain, 2015, pp. 65–71.

[152] Andreu Vall, Hamid Eghbal-zadeh, Matthias Dorfer, and Markus Schedl. "Timbral and semantic features for music playlists." In: *Machine Learning for Music Discovery Workshop at ICML*. New York, NY, USA, 2016.

[153] 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: *Proc. DLRS Workshop at RecSys*. Como, Italy, 2017, pp. 46–54.

[154] Andreu Vall, Markus Schedl, Gerhard Widmer, Massimo Quadrana, and Paolo Cremonesi. "The importance of song context in music playlists." In: *RecSys Poster Proceedings*. Como, Italy, 2017.

[155] Andreu Vall, Matthias Dorfer, Markus Schedl, and Gerhard Widmer. "A hybrid approach to music playlist continuation based on playlist-song membership." In: *Proc. SAC*. Pau, France, 2018, pp. 1374–1382.

[156] Andreu Vall, Massimo Quadrana, Markus Schedl, and Gerhard Widmer. "The importance of song context and song order in automated music playlist generation." In: *Proc. ICMPC-ESCOM*. Graz, Austria, 2018.

[157] Andreu Vall, Matthias Dorfer, Hamid Eghbal-zadeh, Markus Schedl, Keki Burjorjee, and Gerhard Widmer. "Feature-combination hybrid recommender systems for automated music playlist continuation." In: *User Modeling and User-Adapted Interaction* (2019, in press).

[158] Andreu Vall, Massimo Quadrana, Markus Schedl, and Gerhard Widmer. "Order, context and popularity bias in next-song recommendations." In: *International Journal of Multimedia Information Retrieval* (2019, in revision).

[159] Saúl Vargas and Pablo Castells. "Rank and relevance in novelty and diversity metrics for recommender systems." In: *Proc. RecSys*. Chicago, IL, USA, 2011, pp. 109–116.

[160] Paolo Viappiani, Pearl Pu, and Boi Faltings. "Conversational recommenders with adaptive suggestions." In: *Proc. RecSys*. Minneapolis, MN, USA, 2007, p. 89.

[161] Thanh Vinh Vo and Harold Soh. "Generation meets recommendation: Proposing novel items for groups of users." In: *Proc. RecSys*. Vancouver, British Columbia, Canada, 2018, 145–153.

[162] Richard Vogl, Matthias Leimeister, Carthach Ó Nuanáin, Sergi Jordà, Michael Hlatky, and Peter Knees. "An intelligent interface for drum pattern variation and comparative evaluation of algorithms." In: *Journal of the Audio Engineering Society* 64.7/8 (2016), pp. 503–513.

[163] Chong Wang and David M. Blei. "Collaborative topic modeling for recommending scientific articles." In: *Proc. KDD*. San Diego, CA, USA, 2011, pp. 448–456.

[164] Xinxi Wang, Yi Wang, David Hsu, and Ye Wang. "Exploration in interactive personalized music recommendation: a reinforcement learning approach." In: *ACM Transactions on Multimedia Computing, Communications, and Applications* 11.1 (2014).

[165] Markus Weimer, Alexandros Karatzoglou, and Alex Smola. "Improving maximum margin matrix factorization." In: *Machine Learning* 72.3 (2008), pp. 263–276.

[166] Zhe Xing, Xinxi Wang, and Ye Wang. "Enhancing collaborative filtering music recommendation by balancing exploration and exploitation." In: *Proc. ISMIR*. Taipei, Taiwan, 2014, 445–450.

[167] Yuan Cao Zhang, Diarmuid Ó Séaghdha, Daniele Quercia, and Tamas Jambor. "Auralist: Introducing serendipity into music recommendation." In: *Proc. WSDM*. Seattle, WA, USA, 2012, pp. 13–22.

[168] Elena Zheleva, John Guiver, Eduarda Mendes Rodrigues, and Nataša Milić-Frayling. "Statistical models of music-listening sessions in social media." In: *Proc. WWW*. Raleigh, NC, USA, 2010, pp. 1019–1028.

[169] Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. "Large-scale parallel collaborative filtering for the Netflix Prize." In: *Algorithmic Aspects in Information and Management*. 2008, pp. 337–348.

[170] Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, and Georg Lausen. "Improving recommendation lists through topic diversification." In: *Proc. WWW*. Chiba, Japan, 2005, pp. 22–32.