UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

A hybrid approach for item collection recommendations: an application to automatic playlist continuation

Anna Gatzioura

A Hybrid Approach for Item Collection Recommendations

An Application to Automatic Playlist Continuation

Anna Gatzioura

PhD Thesis

Doctoral Program of Artificial Intelligence
Universitat Politècnica de Catalunya

Supervisor: Dr. Miquel Sànchez-Marrè

August 2018

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor, Prof. Miquel Sànchez-Marrè, for the continuous support and guidance throughout all the project realisation. Without his persistent help and valuable advices this PhD thesis would not have been possible.

An important part of this work was realised during my research stay in LIAAD-INESC TEC, in Porto. I am deeply grateful to Prof. Alípio Jorge for giving me this opportunity, for his supervision and constant encouragement. His feedback and the enriching conversations we had, have been an enormous help to improve the quality of this research. I would also like to thank all the members of the LIAAD group for the unforgettable time spent there. Special thanks to Dr. João Vinagre for the always interesting conversations and ideas, on music and recommender systems.

In addition, I would like to thank my colleagues and members of the KEMLG group for all the good moments spent together, the shared experiences and mutual support during these years.

Last but not least. Life, and as part of it this thesis, would be nonsense without the existence and unconditional support of my family and friends. The people that, no matter whether in Greece, Spain, Portugal or England, were and are always there for me, even in my worst. Thank you for that. Please accept my sincere apologies for all those moments that, due to geographical distance and time limitations, I was not able to be as present as I would like to.

Abstract

Current recommender systems aim mainly to generate accurate item recommendations, without properly evaluating the multiple dimensions of the recommendation problem. However, in many domains, like in music, where items are rarely consumed in isolation, users would rather need a set of items, designed to work well together, while having some cognitive properties as a whole, related to their perception of quality and satisfaction.

In this thesis, a hybrid case-based recommendation approach for item collections is proposed. In particular, an application to automatic playlist continuation, addressing similar cognitive concepts, rather than similar users, is presented. Playlists, that are sets of music items designed to be consumed as a sequence, with a specific purpose and within a specific context, are treated as cases. The proposed recommender system is based on a meta-level hybridization. First, Latent Dirichlet Allocation is applied to the set of past playlists, described as distributions over music styles, to identify their underlying concepts. Then, for a started playlist, its semantic characteristics, like it slatent concept and the styles of the included items, are inferred, and Case-Based Reasoning is applied to the set of past playlists addressing the same concept, to construct, and recommend, a relevant playlist continuation. A graph-based item model is used to overcome the semantic gap between songs' signal-based descriptions and users' high-level preferences, efficiently capture the playlists' structures and the similarity of the music items in those. As the proposed method bases its reasoning on previous playlists, it does not require the construction of complex user profiles to generate accurate recommendations. Furthermore, apart from relevance, support to parameters beyond accuracy, like increased coherence or support to diverse items is provided to deliver a more complete user experience.

Experiments on real music datasets have revealed improved results, compared to other state of the art techniques, while achieving a "good trade-off" between recommendations' relevance, diversity and coherence. Finally, although actually focusing on playlist continuations, the designed approach could be easily adapted to serve other recommendation domains with similar characteristics.

Resumen

Los sistemas de recomendación actuales tienen como objetivo principal generar recomendaciones precisas de artículos, sin evaluar propiamente las múltiples dimensiones del problema de recomendación. Sin embargo, en dominios como la música, donde los artículos rara vez se consumen en forma aislada, los usuarios más bien necesitarían recibir recomendaciones de conjuntos de elementos, diseñados para que se complementaran bien juntos, mientras se cubran algunas propiedades cognitivas, relacionadas con su percepción de calidad y satisfacción.

En esta tesis, se propone un sistema hiíbrido de recomendación meta-nivel, que genera recommendaciones de colecciones de artículos. En particular, el sistema se centra en la generación automática de continuaciones de listas de música, tratando conceptos cognitivos similares, en lugar de usuarios similares. Las listas de reproducción son conjuntos de elementos musicales diseñados para ser consumidos en secuencia, con un propósito específico y dentro de un contexto específico. El sistema propuesto primero aplica el método de Latent Dirichlet Allocation a las listas de reproducción, que se describen como distribuciones sobre estilos musicales, para identificar sus conceptos. Cuando se ha iniciado una nueva lista, se deducen sus características semánticas, como su concepto y los estilos de los elementos incluidos en ella. A continuación, el sistema aplica razonamiento basado en casos, utilizando las listas del mismo concepto, para construir una continuación relevante. Se utiliza un grafo que modeliza las relaciones de los elementos, para superar el "salto semántico" existente entre las descripciones de las canciones, normalmente basadas en características sonoras, y las preferencias de los usuarios, expresadas en características de alto nivel. También se utiliza para calcular la similitud de los elementos musicales y para capturar la estructura de las listas de dichos elementos.

Como el método propuesto basa su razonamiento en las listas de reproducción y no en usuarios que las construyeron, no se requiere la construcción de perfiles de usuarios complejos para poder generar recomendaciones precisas. Aparte de la relevancia de las recomendaciones, el sistema tiene en cuenta parámetros más allá de la precisión, como mayor coherencia o soporte a la diversidad de los elementos para enriquecer la experiencia del usuario.

Los experimentos realizados en bases de datos reales, han revelado mejores resultados, en comparación con las técnicas utilizadas normalmente. Al mismo tiempo, el algoritmo propuesto logra un "buen equilibrio" entre la relevancia, la diversidad y la coherencia de las recomendaciones generadas. Finalmente, aunque la metodología presentada se centra en la recommendación de continuaciones de listas de reproducción musical, el sistema se puede adaptar fácilmente a otros dominios con características similares.

Contents

List of Figures

List of Tables

List of Acronyms

APC – Automatic Playlist Continuation
APG - Automatic Playlist Generation
AR(s) - Association Rule(s)
CF - Collaborative Filtering
CB - Content-Based
CBR - Case-Based Reasoning
ILD - Intra List Diversity
IR - Information Retrieval
LDA - Latent Dirichlet Allocation
LSA – Latent Semantic Analysis
MBA - Market Basket Analysis
MF - Matrix Factorization
MIR - Music Information Retrieval
MRS(s) - Music Recommender System(s)
NLP - Natural Language Processing
RS(s) - Recommender System(s)
SP(s) - Sequential Pattern(s)

Part I

Introduction

Chapter 1

Introduction

1.1 General

The exponential growth of the World Wide Web, along with its extended use in recent years, have enabled the generation of a huge amount of multimedia and information about them, coming from different sources, and being available through the internet and the social networking platforms. In addition, the extended use of mobile devices, has enabled users in consuming and sharing those, any time and from any place.

When it comes to music, the improved capabilities of digital music production, have enabled its creation and public distribution, with lower costs than in the past, and without geographic limits. This has resulted in an increased amount of *music items*, and information about them, easily accessible. Music items in general can be songs, artists, genres, albums and radio stations (Schedl et al., 2015). In addition, a change in the way that music is "consumed" has been observed. Users, nowadays, more than "owning" songs in their personal collections tend to listen to them online (Schedl, Knees, & Gouyon, 2017). Therefore, in order to help users in finding and properly experiencing the music they want, numerous issues have arisen, related to the discovery, organisation, sharing and information services, that need to be facilitated (Celma, 2010).

To this direction, music recommender systems (MRSs), recommender systems (RSs) related to music, seem as among the promising solutions to handle the information overload related to music, and help users to find relevant songs and artists, that otherwise would have difficulties in discovering (Schedl, Yang, & Herrera-Boyer, 2017). In addition, due to the semantic gap between the songs' signal-based descriptions and the human perception of music, these systems seem as the most appropriate tools to better capture users' needs and ensure their satisfaction (Celma et al., 2006).

Music items are rarely consumed in isolation but rather as well designed sequences. Therefore, increased focus has been lately placed on automatic playlist generation (APG) and recommendation, to support users in organising their music libraries and having a more complete experience. *Playlists* can be defined as *sets of music items* designed to be consumed as a sequence, similar to traditional radio broadcasts (Bonnin & Jannach, 2014). In playlist recommendation, and similar domains, more than recommending isolated items, or presenting an ordered list of the most promising alternatives for a user, like the majority of RSs do, the underlying structure of *joint selections* should be evaluated, as item interactions within a set may heavily influence the result (Schedl, Zamani, et al., 2017)

Therefore, the presence of an item within a concrete concept should be captured, in order to recommend sets designed to be consumed together, under given circumstances, satisfying at the same time cognitive properties like relevance, coherence and diversity (Smyth & McClave, 2001). This becomes even more complex if we keep in mind the effect that the additional dimensions of the recommendation problem, like the circumstances under which those selections are made, have (Kaminskas & Ricci, 2017). Furthermore, the influence even of the same, or very similar contextual situations, on a user's mood, music perception and preferences, has been found to heavily depend on the user's character and emotional state (Kim et al., 2010).

1.2 Motivation

The general scope of a RS is to find the items that a specific user would like to use, buy or enjoy, in a given moment. Therefore, its functionality is based on the approximation of a utility function describing the degree of (expected) satisfaction that a user would experience from using an item, and the recommendation of the items that maximise this expected utility (Ricci et al., 2010).

Although having become a fundamental part of various applications, recommender systems still base their reasoning mainly on pairwise interactions or information on individual entities, like item attributes or ratings assigned to them by users. The widely used methodologies, like *Collaborative Filtering (CF)* and *Content-Based (CB)* tend to recommend users items that have been *liked by similar users* or *items similar to those they have already used and liked* in the past, based on the assumption that users' preferences remain stable over time. In general, these approaches focus on isolated item characteristics, or simple ratings and tend to ignore the underlying structure of users' preferences and selections. Thus they provide a limited insight into users' behaviour.

However, in various everyday situations, similar to playlists, users consume items as *sets*, or *sequences*, referred to also as collections or packages (Golbeck & Hansen, 2011; Interdonato et al., 2013). Therefore, their satisfaction depends on the characteristics of the entire set of items to be used together, more than on the characteristics of a specific item. In such cases, the item interactions within a set may be crucial for the acceptance of an item (complementary or incompatible items) and the quality of the set, as users evaluate the whole set more than selecting items from the top-N list (Maillet et al., 2009). In these application domains, the *"independence assumption"* that users select items based only on item characteristics, and their preferences over those, without taking into account the rest of the selected items, is not valid. The item interactions, the scope and the context under which those sets were performed may be more crucial that the general user preferences. For instance, although, a user may like from classical to rock music, usually would not place songs of these categories into the same playlist. More likely, he/she would organise songs of similar styles into coherent playlists that could listen to in different occasions (Jannach et al., 2015). For example, a user might select an instrumental playlist when studying and to an alternative rock one when preparing to get out.

The above example aims to highlight a slightly different perspective of the recommendation problem that has been less investigated, *the recommendation of sets, or sequences, of items*, that in the music domain can be captured as the *recommendation of playlists*, or *playlist continuations* (Logan, 2004; Bonnin & Jannach, 2014)

Therefore, the specifications of a RS used for this problem, should be slightly different from the usual. Further than the user-item relationships, the similarities between items and users and the generation of a set of top-N candidates, the items that are selected within the same, or similar, sessions have to be analysed and evaluated. The aim is to extract meaningful information about the structures of the sets, and the parameters that may influence their quality and acceptance by the users. In general, there should be some level of *coherence*, excluding items that do not fit within a given concept, while at the same time being of some *diversity*, and permitting *novel* items to be recommended. Enabling the discovery of varied items that a user would not be find otherwise, instead of presenting only closely similar items to the existing ones, might increase the user interest.

Finally, as users' joint selections may be varied, depending on various parameters, the starting part of the set/list each time is of high importance, as it defines the actual style of a user at a specific moment. Therefore, the ability of a system to analyse this initial set, and identify the important parameters based on which the resulting set should be built, is of high importance. We could suppose that the general style of a set would be heavily influenced by the context within which it is built, but such information may not always be available. Thus, first a *general style*, or *concept*, of the set should be identified, and then evaluate whether the available explicit contextual information improves the similarity approximations or not.

1.3 Scope

The scope of this thesis is the investigation of the possible alternatives to treat the recommendation problem in domains where items are consumed as "sets" or "sequences", and item co-occurrences affect further item selections, while the treated cases may be characterised of rich content and semantic data, like in the music domain.

More precisely, the proposed system aims to generate recommendations of sets of music items to complete a started playlist, not related to a specific user, but to a specific concept. User taste and needs are highly related to the context within which they are generated and consumed. Thus, the same user may look for very different items in different moments. Therefore, the recommended items should be liked by the user, but more crucially they should fit well within a started list.

A hybrid recommendation system for automatic playlist continuation (APC), has been designed and implemented and will be presented. In general, APC is considered as a variation of automatic playlist generation (APG). More than generating entire playlists based on a seed song or some constraints set by a user, APC consists of forming a pleasant continuation of a started playlist, by adding a number of songs to it, while having some target characteristics in common (Schedl, Zamani, et al., 2017). More precisely, given a started list, the focus is set on recommending sets of songs able to complete it, being relevant, coherent and of some diversity degree, in order to provide a more exciting user experience. These recommendations are generated independently of the user who has started the list, and without the necessity of having explicit contextual information.

Nevertheless, although its feasibility is illustrated by its application to the music domain, the designed recommendation approach is generic, and could be easily adapted to other item collection recommendation domains with similar needs, such as multimedia sequential domains, travel recommendations, etc. The initial idea of this research work came from the previously implemented Master Thesis (Gatzioura, 2013) where, based on the fact that joint item selections follow some underlying patterns, the possible alternatives to treat the Market Basket Analysis recommendation problem were evaluated.

1.4 Why music playlist recommendations?

As Frank Zappa[1] had said, *"without music to decorate it, time is just a bunch of boring production deadlines or dates by which bills must be paid"*. Following this notion, although the proposed approach is generic, the focus is set on APC recommendations. Increased emphasis has been placed lately on this topic also in the RSs' community, defining it among the current RSs research challenges (Schedl, Knees, & Gouyon, 2017).

Playlists contain the notion of *item sequences and set characteristics*, while being profoundly affected by the context in which they were generated and consumed (Kaminskas et al., 2012), as this is perceived and reflected by each user, depending on his/her personality characteristics (Ferwerda et al., 2017). To generate a pleasant playlist being able to complete an existing one, first of all it must be aligned with the style and the characteristics of the started list in order to form an appropriate continuation. Apart from the general music style, additional features, like the existing level of coherence and diversity, should be maintained. For instance, in cases that users seem not to care about preserving some specific style and enjoy more diverse songs the recommended set may also be more diverse. On the other hand, users with more specific preferences would place more importance on the coherence of the initial and the recommended set. This is the reason why the recommendation approach should first evaluate the attributes of the started list and then try to identify the most suitable recommendations.

This application domain, except from highlighting the importance of the joint item selections is also heavily affected by the so-called *semantic gap*, characterising the majority of multimedia, information retrieval and recommendation applications. In these domains, users, in general, describe their needs through high-level requirements while the treated items are associated with low-level, application specific characteristics (Kim et al., 2010). Therefore, it is important to find recommendation techniques capable of overcoming this gap. An effective mapping of the user requests into item specifications is crucial to identify the items that best fit within a given concept, as this has been defined in a started playlist.

[1]https://www.rollingstone.com/music/artists/frank-zappa/biography

However, there is still a lack of reliable methods combining users' cognitive perception of music with sound characteristics. Therefore, it becomes even more difficult to capture users' perception of a music playlist and specify the characteristics that the songs composing a "good" playlist should have. This notion can be highly subjective, depending on various parameters like a user's music knowledge, general preferences and current emotional state, context and intent (Schedl, Zamani, et al., 2017).

1.5 Why using a hybrid approach?

The functionalities that a music playlist recommender, as *collection recommenders* in general, must serve are slightly different from those of the usual RSs. Therefore, this thesis hypothesises that a recommendation approach, in order to capture the additional dimensions, and serve the needs of the specific problem, should follow a different design and a corresponding evaluation methodology.

Current popular recommendation approaches, like CF and CB, have different scopes and mainly use the user-item matrix to explode the past interactions and estimate users' future preferences in terms of isolated items. These techniques come along with specific hypothesis, and do not explore the joint item selections or the circumstances under which those were performed. Thus, they have a limited performance when it comes to recommendations of sets of items. This is due to the fact that these techniques try to predict the suitability of an item for a user, and not the suitability of an item for a specific *concept*. Furthermore, users may construct various sets and select different items under given circumstances, being highly different between them, and not always made of their favourite items or the most popular ones. Thus, simply predicting if an item would be liked by a user and recommending it, without evaluating the whole concept, is not enough and usually leads to lower performance, especially in domains where each user performs a lot of transactions and where the popularity of an item does not increase its utility value. In addition, in domains when we have many single item occurrences and user preferences cannot be ranked, these methods also have a low performance.

Furthermore, although widely used in commercial applications, CF recommendation systems, due to their reasoning that is based on the opinions of users on items, still face *cold-start*[2] and *popularity-bias*[3] problems that limit their performance (Su & Khoshgoftaar, 2009). They are domain independent and generally ignore the multi-aspect concept of user experience that has to be supported from various perspectives in order to be complete. In contrast, CB approaches, that in music applications have mainly used sound related attributes, may treat new items but show *limited diversity* of the recommended items and may lead to *recommendations' overspecialization*[4] (Lops et al., 2010; Bogdanov et al., 2010).

[2]Lack of ability to recommend new, yet un-rated items, and to perform well when new users come, until they have enough information about them

[3]Tendency towards the recommendation of popular items, those that have been rated more times

[4]Focus on the characteristics of the items that users have liked, and recommendation of closely similar ones

Therefore, this thesis hypothesises that a hybrid solution (R. Burke, 2002) designed for the specific domain would be more appropriate to address the problem's needs without facing the limitations of the current RSs. Further than the user-item relationships and similarities, the items selected within the same and similar sessions and concepts have to be analysed, to extract information about these sessions, like their structures and the item styles in them. In addition, more than focusing only on their predictive ability, these systems should go *beyond accuracy* and incorporate parameters related to the recommendations' quality, as perceived by the user, and present interesting alternatives that the users would not discover otherwise (Konstan & Riedl, 2012).

1.6 Contributions

In this thesis, we present *HybA*, (whose name was inspired by the title of the Radiohead album, *Kid A*) *a hybrid recommendation approach for APC, that aims to generate recommendations of playlist continuations of improved quality, not restricted to a specific user, but related to a specific "cognitive concept".*

The user should like the recommended items, but most important, they should fit well within the started list, as user tastes and item needs, are highly related to the concept within, or for, which they are generated and consumed (Pichl et al., 2015). Therefore, this recommender, given a list of already selected songs, aims to find and recommend the set of songs that seem more adequate to complete this list. The proposed approach evaluates the similarity among playlists based on the general concepts expressed in them, along with the joint distributions of music styles and songs' characteristics in them. When a new playlist is introduced, the system follows the general Case-Based Reasoning (CBR) cycle (López de Mántaras, 2001) to find the past most similar playlists, and based on their inherit characteristics and the specifications of the items co-occurring in those, to identify the most adequate way to complete the new list. The degree of similarity of music items is calculated based on the density of their connections in a graph model that connects them through their common attributes and appearances in playlists.

The basic functionalities that differentiate HybA from the commonly used techniques, which also form the contributions of this work, can be summarised in the following points:

- *The generated recommendations are not restricted to specific, or referring to similar, users but are addressing similar concepts.* The proposed approach focuses on the characteristics of entire playlists constructed in different time moments, and under different circumstances. The term "playlist concept" is introduced to capture the semantic characteristics of playlists, being beyond explicitly defined context or music attributes.

- *Item ratings are not taken into account.* Although the selection of an item is treated as a sign of preference, we do not use the opposite statement. More specific, due to the *data sparsity*[5] and the popularity bias usually observed, a non-selection cannot be always classified as a low, or negative,

[5]Despite the huge amount of item alternatives, each user accesses only a small percentage of them and rates even less

rating. It could simply be a consequence of the *long tail*[6] item popularity distribution, especially when items with similar characteristics have been used in similar concepts. We rather treat the selection of an item as a sign of preference towards the cluster of items with the item's characteristics.

- *Two abstraction levels are used, to evaluate the general characteristics of playlists and then the items in those.* Playlist latent concepts and music style co-occurrences are first evaluated, using songs' content descriptions or the metadata associated with them. Therefore, more than the presence or absence of a specific item within a set is evaluated.

- *The two-level reasoning of the designed algorithm permits the evaluation and incorporation into the final recommendations of parameters beyond accuracy, related to playlists' quality.* In contrast to most recommendation techniques, that aim simply at maximising some similarity, or minimising a distance function in to find the top-N most relevant items, this system retrieves the most appropriate past cases and then constructs the recommendation list based also on additional parameters, like coherence and diversity.

- *The combined use of accuracy and quality related evaluation metrics is proposed in order to set more emphasis on the parameters related to user perception of playlist quality.* In general, when evaluating a RS accurate recommendations are considered as better. However, recommendations being only relevant may fail in addressing users' expectations and delivering an added value from the system's use. Users would probably discover those items without being recommended them, or maybe these items do not form a pleasant joint result.

1.7 Research Questions

Based on the defined scope and the characteristics of the proposed RS, the following research questions will be addressed in this document, and finally evaluated:

1. *Are the currently used recommendation algorithms able to efficiently handle automatic playlist continuation, and item collection, recommendations? Does the proposed algorithm manage to address the needs of the specific problem in a better way?*

2. *How does the use of additional parameters, related to the quality characteristics and user perception of the sets of music items, affect the recommendation results?*

3. *Are the commonly used evaluation metrics suitable when recommending item collections? Which other factors should be evaluated?*

[6]In the majority of RSs applications, there is a small group of items frequently used and recommended while the majority of items remain un-discovered by the users

1.8 Overview

The document at hand, after this introductory chapter, is followed by two basic parts. One related to the necessary background and state of the art review and one associated with the primary contribution of this thesis: the description of the designed and developed system, its evaluation and comparison with other state of the art recommendation techniques.
More specific, the rest of the document is structured as follows:

As the proposed system tries to address a slightly different problem, by applying a hybrid approach, different from the commonly applied techniques, various topics and research domains are first presented to enable the further comprehension of this document. In the second chapter, the general background of recommender systems and the commonly used techniques can be found. From those, more information on Case-Based Reasoning, case-based recommenders, probabilistic topic models and latent semantic analysis are presented, as these are the methodologies used in the development the proposed hybrid solution. Further, in chapter 3, music information retrieval and music recommender systems, as well as the basic approaches to automatic playlist generation and continuation are described. The characteristics that differentiate this application domain and the reasons that make it challenging are also introduced. Finally, the additional dimensions of the recommendation problem, that shift its focus from accuracy to quality and user experience, especially related to music recommenders, are presented.

Going on, in chapter 4, the detailed description of the proposed system, its architectural design, the used data models and similarity functions, as well as the recommendation methodology, along with its gradual evolution, are presented in more detail. In chapter 5, the evaluation of the system is provided. First, some details on the used evaluation datasets, the evaluation metrics and the compared techniques, used as baseline are presented. Then, from one part the evaluation of the various versions of the designed system, the effect of the several parameters used, and the definition of the "optimal" version of the HybA can be found. Following, the comparison of its results with those of the previously presented state of the art techniques is presented.

Finally, in chapter 6, some conclusions and the relevant findings of this thesis, forming its contributions, are described. In continuation, a brief presentation of the main ongoing and future planned, research issues, can be found. The document is ended with a list of relevant publications by the author, followed by a list of bibliographic references related to the presented work that have been studied.

Part II

Related Work

Chapter 2

Recommendation Techniques

In this part, the state of the art of the basic concepts and areas that this thesis examines are presented. Specifically, the definition of Recommender Systems (RSs), their main techniques and some historical data are first described. The designed system expands the usual recommendation approaches, and more than recommending single items, songs to users, focuses on the recommendation of *playlist continuations*, being *sets of songs* that seem most appropriate to complete a user's experience concerning relevance and quality. Moreover, the distinctive characteristics of sets of items and the basic dimensions that differentiate these recommendations are presented. Furthermore, the basics of Case-Based Reasoning (CBR), case-based recommenders, Latent Analysis and Probabilistic Topic Models with focus on Latent Dirichlet Allocation (LDA), along with some of their applications are described, as the proposed approach is based on the combined use of these techniques.

2.1 Recommender Systems

The amount of items being able to support the same or very similar needs has increased in recent years. As a result, the number of alternative choices and information about them, that users have to find and review, has increased. This fact also increases the complexity of the related decision-making processes. The knowledge, and even more the evaluation, of all the possible alternatives by one person, have been transformed into a non-feasible problem (Bollen et al., 2010). Thus, average (non-expert) users seek support from friends and/or experts in fields that they lack of expertise. Recommender Systems have been identified as a promising solution as they "mime" these behaviours, they can efficiently filter the current information overload and support users in their search and decision making processes (Ricci et al., 2010).

Recommender (or Recommendation) Systems (RSs) are software tools and techniques for information retrieval (IR) and filtering used to generate and provide meaningful and accurate recommendations of items, that is expected that the active user would like to use (Melville & Sindhwani, 2010). These systems apply knowledge discovery techniques and possibly adapt the results through personalised models to provide accurate recommendations to users seeking for support to their decisions (Jannach et al., 2010).

The initial ideas and techniques on which the development of RSs was based come from the extensive work previously done in the fields of information retrieval, forecasting techniques, cognitive science as well as management, marketing and customers' preference modelling. RSs, as part of an independent research field, emerged around the 1990s with the first commercially used recommender systems being mainly based on *collaborate filtering* algorithms, (Adomavicius & Tuzhilin, 2005). These systems were initially inspired by the necessity to handle the increased amount of information that people receive through electronic mail and discussion lists on the web. In 1992, in order to enable people in subscribing to newsgroups and receiving only documents of interest to them, the first commercial RS that used a "social" collaborative filtering approach, called Tapestry, was developed (Goldberg et al., 1992). In 1994, GroupLens[1] developed another collaborative filtering system for Usenet newsgroups with scope to enable users in finding articles of interest to them, using past users' subjective evaluations of articles to predict other users' interest in articles, (Konstan et al., 1997; Resnick et al., 1994).

In recent years, RSs have received an increased amount of interest, from both academic and industrial research centres. Their potential ability to handle the information overload and supporting users in finding new items able to fulfil their needs, while on the other hand supporting providers in increasing the amount and diversity of the items that they sell, has led to their establishment as an independent research area. In addition, the range of RSs' applications has faced a significant increase transforming them into an essential part of many frequently visited internet sites, especially in *e-commerce applications* and *online marketing* as efficient personalised RSs increase the possibility of a user purchasing an item (Prasad, 2003). Currently, recommendation techniques are also used in *leisure time* and *travel activities*, *music* and *multimedia*, *reading* and *information sharing*.

In 2006, Netflix[2], recognising the importance of an effective and accurate RSs, to improve the prediction accuracy of the recommendation algorithm that was using, announced a competition for the implementation of the best collaborative filtering algorithm with a high prize for the winner, (Takács et al., 2008; Schafer et al., 2001). Some of the actual well-known RSs' applications are Amazon[3], Youtube[4], Ebay[5], Moviefinder[6], Last.fm[7], IMDd[8], etc. The usually applied recommendation methodologies in those RSs are based both on *item-to-item* and *user-to-user* correlations.

In the Amazon book section, for example, the "Customers who bought" feature can be found in the information page of each item (book) and it provides two recommendation lists, one containing books that are usually purchased by customers who bought the concrete book, while the second suggests authors whose books are frequently purchased together with books of the author of the

[1]http://grouplens.org/
[2]https://www.netflix.com
[3]http://www.amazon.com/
[4]https://www.youtube.com/
[5]http://www.ebay.com/
[6]http://www.moviefinderonline.com/
[7]http://www.last.fm
[8]http://www.imdb.com/

selected book (Linden et al., 2003). In CDNow, the album advisor feature also works in two different modes, the single album mode that generates a list of ten albums that may be of interest to the user based on an album he/she has already selected, while on the multiple artist mode, a list of ten recommended albums is generated based on the user's selection of up to three artists. The Moviefinder's Match Maker enables users in finding movies with similar mood, theme or cast to a particular movie. The recommendations are also generated in two lists, one with the suggested films and one containing links to other films by the director and/or key actors of the film (Schafer et al., 2001).

RSs either generate a set of recommendations/suggestions of the (top-N) items that are expected to be useful for a user, or intend to predict whether a specific item will or not be of interest to him/her, based on his/her previous preferences as well as on the preferences of similar users.

In its simplest form, the set of recommendations provided is a list of ranked items. The term *item* refers to the type of entity being recommended by each recommender (ex: products, songs, web pages, information, services, etc.) to users, and of course, depends on the application domain and the scope of the specific system. The term *transaction* refers to a recorded interaction between a user and a system (Deshpande & Karypis, 2004). Transactions are log-like data that contain important information collected and/or generated through the interaction of the user with the system, that can be used to infer additional information and improve future recommendations. A *transactional model* contains references to previously selected items, that depending on the recommendation methodology used, may be expressed in terms of ratings (implicitly or explicitly collected), likes, descriptions of their context, comments or another adequate way for the application representation (Ricci et al., 2010).

RSs primary refer to users with limited or no personal experience and knowledge in a specific area, and therefore, without the ability to evaluate and/or select among the offered items in this category. These users would traditionally seek help from friends or experts in order to complete their selection. To retrieve their preferences and based on them to gain the ability to generate meaningful and personalised suggestions, RSs observe and try to gather explicit or infer implicit user preferences, while taking into account possible constraints arising or imposed by those users. The core recommendation algorithm generally consists of a particular type of a data mining algorithm that includes adequate data preprocessing, analysis and interpretation (Amatriain et al., 2011).

The main components/data important for the use of a RS, as shown in figure 2.1, can be divided into:

- *Background knowledge* that consists of the information/data necessary for the system before the instantiation of the recommendation process. These data may be related to users, items, context, sentimental and emotional factors as well as characteristics of the application domain and/or the specific methodology being used.

- *Input data* that is the information that the active user provides to the system to receive recommendations that may be submitted in terms of a specific request or as seed item(s). Usually, this information is entered through a user interface that may map the user requirements to specific parameters that are then used by the rest of the system.

- The core component of the recommender is the *recommendation algorithm* that combines and processes the background and input data in order to generate meaningful recommendations in line with the active user's request (Deshpande & Karypis, 2004). The exact functionality of this component depends heavily on the type and scope of the system, together with the data that this system is required to process.

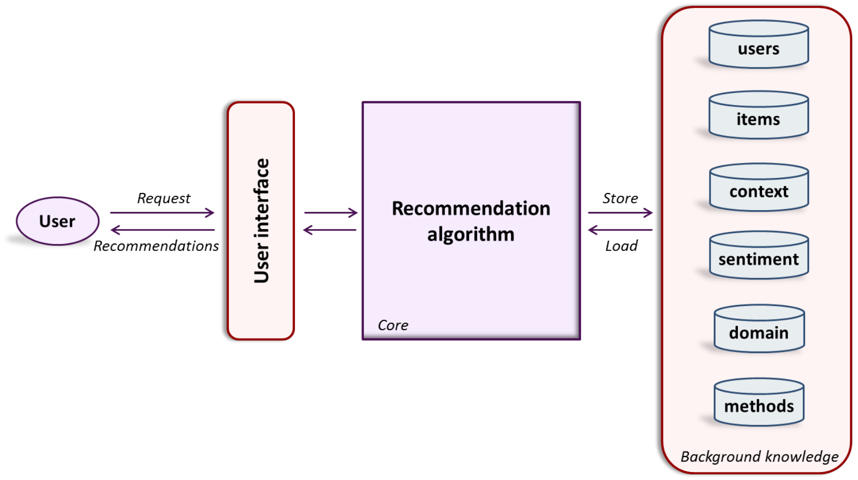


Figure 2.1: Basic components of a Recommender System

In general, the basic idea behind a RS is that a rational user, who is aware of the existing alternative items and their characteristics would always select the item(s) that can best serve his/her needs, to maximise his/her utility under certain circumstances. Therefore, in order to be successful, a recommendation algorithm has to identify those items, concerning the needs and preferences of the current active user, within a reasonable response time. Thus, the recommendation problem can be transformed into the estimation of a utility function able to best capture the level of utility a user would obtain from the use of not yet used items, and the recommendation of those that maximise the user's utility.

The *utility* of an item can be represented by a rating showing the level of its likeliness for a user. Recommending users items able to maximise their utility may increase their satisfaction of using both the items and the recommendation system, thus leading to increased trust in it (Adomavicius & Tuzhilin, 2005).

On the other hand, a system failing to recommend adequate items to the users that interact with it, or generating recommendations with a substantial delay, most probably will lead to the dissatisfaction of users and their unwillingness to keep using the concrete system (Huang, 2011). Hence, the adequate modelling of the traded items, the target users with their needs and expectations, as well as the situation within which the item selection takes place is crucial for the outcome of the recommendation process and the system's popularity. The use of inappropriate representations of the user profiles that do not reveal the needs and preferences of users, as well as the low quality of the used data collections have been identified among the common issues that restrict the effectiveness of RSs (Park & Tuzhilin, 2008).

2.1.1 Problem Formalization

Therefore, the aim of a RS can be defined, as follows. Given:

- A set of *users*, U, that interact with the system
- A set of items, I, that may be recommended (songs, products, etc.)
- A utility function v which measures the utility that a user from U gets by using an item from I, $v : U \times I \rightarrow R$, where R is a totally ordered set.
- Recommend a user $u \in U$ the item(s) $i' \in I$ that maximises the expected utility, thus:
$$u \in U, \, \forall i \in I : i' = argmax\{v(u, i)\} \tag{2.1}$$

However, among the main problems that come from, is the fact that usually both the set of users and items are very large, and as a result, each user only uses a very small percent of items making the accurate estimation of the utility function more difficult.

2.2 Main Recommendation Approaches

The widely used recommendation methodologies in commercial applications can be mainly divided into *Collaborative Filtering* and *Content-Based*. However, due to the extended use of RSs in various areas, along with the shortcomings that these two methodologies come with, other recommendation techniques are also being investigated and used. The primary categories of recommendation techniques identified in the literature (Ricci et al., 2010) are listed below,

- Collaborative Filtering (memory-based, model-based)
- Content-based Filtering (probabilistic models, Nave Bayes classifier, Artificial Neural Networks)
- Knowledge-based (Case-based, constraint-based)
- Context-based
- Utility-based
- Rule-based
- Demographic
- Hybrid
- Other

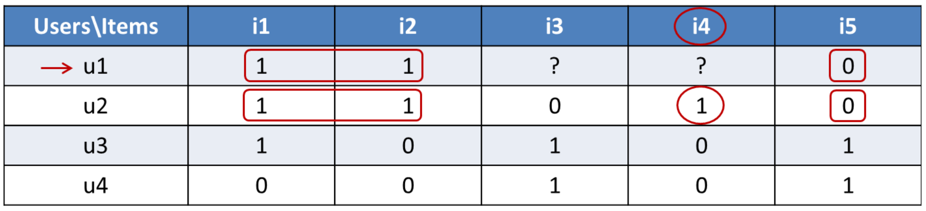
2.2.1 Collaborative Filtering

In *Collaborative Filtering (CF)* recommendation techniques, items among those liked by similar users *(neighbours)* are recommended to the active user. CF techniques have been identified among the most successful approaches for building RSs and have been successfully used in many commercial applications (Su & Khoshgoftaar, 2009; Adomavicius & Tuzhilin, 2005). However, due to their reasoning, collaborative RSs still face *cold-start* and *popularity-bias* problems, that limit their performance. The primary approaches of CF can be divided into neighbourhood and latent factor, models.

Neighbourhood Models

Neighbourhood models use the hypothesis that users have a stable buying behaviour in time, i.e., *if two users have rated some items similarly in the past, they are most probably going to evaluate other items in the future in a similar way.* Therefore, they build user profiles of the items that have been highly rated by them and the similarity of users' tastes is deducted from their previous ratings. Users and items may be represented through the user-item (or preference) matrix where each cell, r_{ij} represents the rating the i-th user (line index) has assigned to the j-th item (column index) or zero in case that this user has not used this item before, like in figure 2.2 (Melville & Sindhwani, 2010).

For a given request these techniques first identify the neighbours of the active user, based on their previous interactions usually expressed through ratings. Then, from the set of their highly rated candidate items, they recommend those that the active user has not yet tried.



Users\Items	i1	i2	i3	i4	i5
$\rightarrow$ u1	1	1	?	?	0
u2	1	1	0	1	0
u3	1	0	1	0	1
u4	0	0	1	0	1

Figure 2.2: Preference matrix example for a CF recommendation process

Based on the use of the user-item matrix that they perform, CF techniques can be divided into *memory-based* and *model-based* recommendation techniques. Memory-based techniques use the whole user-item matrix to generate item suggestions, while model-based approaches use this matrix to train the system and based on the learned relationships to generate recommendations.

Latent Factor Models

Latent factor models try to explain users' rating patterns based on the latent factors inferred from them. One of their most successful implementations are *matrix factorization (MF)* models, closely related to *singular value decomposition (SVD)*. MF models have been found able to outperform nearest-neighbours approaches while better handling with additional information sources (Bell et al., 2009).

MF models map both users and items to a joint latent space of dimensionality f, being R^f, and model their interactions as inner products on that space. More specific, each user u is associated with a vector $p_u \in R^f$ and each item i with a vector $q_i \in R^f$. These vectors represent, respectively, the factors which a user is highly interested in, and the factors that an item possesses. Therefore the interaction of a user and an item is the resulting dot product between their latent vectors being $q_i^T p_u$.

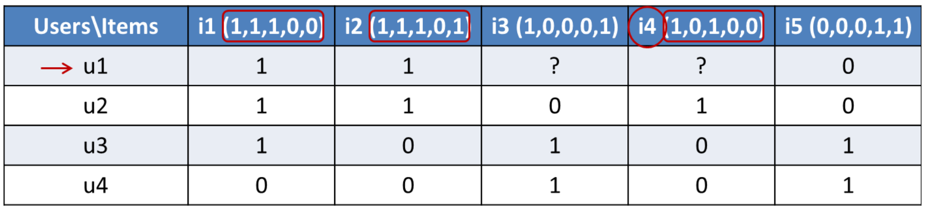
When used for recommendations, the approximation of the rating r_{ui}, of user u on item i, with b_{ui} being the bias associated with the rating, can be computed as, (Cremonesi et al., 2010):

$$\hat{r_{ui}} = b_{ui} + q_i^T p_u \tag{2.2}$$

The main approaches used for learning the user and item vectors are the *alternating least squares (ALS)* and the *stochastic gradient descent (SGD)* (Bell et al., 2009). One of the main challenges that these technique face is that many times explicit feedback, users directly rating items, is not available. Therefore they have to implicitly extract user preferences, and the majority of these techniques treats all the missing data as negative feedback. However, in many cases, this assumption degrades the learning efficiency and, as a consequence, the recommendation performance. Various improvements have been proposed aiming to better capture the dynamic nature of online data, by learning from positive only data streams (Vinagre et al., 2014) or changing the uniform item weighting schema (He et al., 2016).

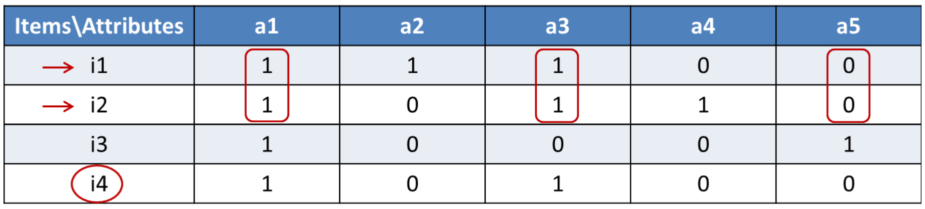
2.2.2 Content-Based

Content-Based (CB) RSs recommend users items similar to those they have shown preference in the past, (Adomavicius & Tuzhilin, 2005). In CB systems, user profiles are built from the characteristics of the items that a user has rated highly, and the items that she/he has not tried yet are compared to them. The items with the higher estimated possibility of being liked by the active user, as deducted from his/her past preferences, are then recommended (Melville & Sindhwani, 2010). An example can be found in figure 2.3 where a user has used and liked items i_1 and i_2 while the analysis of the items with their attributes explain the CB recommendation example is shown in figure 2.4.



Users\Items	i1 (1,1,1,0,0)	i2 (1,1,1,0,1)	i3 (1,0,0,0,1)	i4 (1,0,1,0,0)	i5 (0,0,0,1,1)
→ u1	1	1	?	?	0
u2	1	1	0	1	0
u3	1	0	1	0	1
u4	0	0	1	0	1

Figure 2.3: Preference matrix example for a CB recommendation process



Items\Attributes	a1	a2	a3	a4	a5
→ i1	1	1	1	0	0
→ i2	1	0	1	1	0
i3	1	0	0	0	1
i4	1	0	1	0	0

Figure 2.4: Item-attribute matrix for the CB recommendation example

CB approaches mainly handle the item recommendation problem mostly as an information retrieval, or an item classification problem. Thus, machine learning techniques can be used. In addition, as CB techniques rely on more specific information about users and items, the creation of appropriate user and item profiles becomes more crucial. A profile that accurately reflects user preferences increases the system's recommendation effectiveness. This is something that has become of great importance in the recently evolved business strategies in e-commerce (Prasad, 2003). These techniques are able to recommend new items that could cover the user's needs. However, as they always recommend items similar to those that a user has already used and liked, the main problems that they have to overcome are the recommendations' *limited diversity* and the possible *overspecialization* (Lops et al., 2010).

2.2.3 Utility-Based

Utility-based RSs generate recommendations based on the computation of the *expected utility* of each item for the active user. A common approach to utility-based RSs can be designed based on multi-attribute utility theory and constraint optimization, aggregating the total utility value of an item based on the values of its attributes along with the importance of these characteristics to the target user (Huang, 2011). As a result, these RSs do not face cold start and sparsity problems. However, in order to be effective and able to provide accurate recommendations to users, these systems should develop a different utility function for each user, or for small groups of really similar users, based on their preferences to capture their real needs and the importance that he/she assigns to each of the items' attributes, depending on the application domain (Adomavicius et al., 2011).

2.2.4 Rule-Based

Rule-based recommendation techniques generate item recommendations based on a set of rules extracted from a data corpus. *Association rules (AR) mining* refers to the observation of transactions with the aim to discover interesting hidden patterns and frequent associations among the existing items, usually expressed in the form of "if-then" statements, (Agrawal et al., 1993). Similarly, when it comes to sequential data, itemsets that occur in specific order, *sequential pattern mining* that follows similar principles with AR but also evaluates the relative order of events, can be used (Cavique, 2007).

In general, given two non-over-lapping sets of items, X and Y, with X and Y being subsets of the set of available items and $X \cap Y = \emptyset$ an association rule is an implication of the form $X \rightarrow Y$ which indicates the existence of a strong relationship among the presence of X and Y. When implementing a recommender system based on such rules, when the items of the antecedent X are found in a given set, the items of Y seem as the most suitable to complete this based on the observed correlation that indicates their joint selection in past transactions. In order to evaluate the strength and usefulness of the rules found in a dataset, *support*,*confidence* and *interest* or *lift*, of a rule are used.

- *Support* of a rule determines how often a rule is applicable to a given transactional dataset, and it can be defined as the relative number of transactions, that contain all items from both X and Y, in other words it is an estimation of the joint probability of X and Y, $s = Pr(X \cup Y)$.
- *Confidence*, in general, gives a measure of the reliability of a rule. It is the parameter that determines how frequently items in the consequent appear in transactions that contain the antecedent. Therefore, it is an estimation of the conditional probability of X given that Y has occurred, $c = \frac{Pr(X \cap Y)}{Pr(X)}$.
- *Lift* is used in cases that X and Y are statistically independent to evaluate whether they occur together more often than expected, and it can be defined as the ratio of the rules' confidence to its expected confidence.

Among the main drawbacks of using association rule analysis is its computational difficulty when applied to large scale data, as well as the difficulty to evaluate all of the discovered patterns (Leake, 1996). More precisely, depending on the underlying data, low values of minimum support permit the extraction of rules that rarely occur, therefore may lead to misleading recommendations. On the other hand, using high support levels may result in very strict constraints that do permit the extraction of enough rules to generate recommendations.

2.2.5 Knowledge-Based

Knowledge-based recommendation methods insert knowledge models into the recommendation process, usually by techniques based on the inference of users' needs and preferences. These techniques may extract functional knowledge about the relationships between a user and an item and the way this item is able to meet a concrete user's need. Therefore, they may be especially useful in domains of complex products or in complex purchasing techniques.

Knowledge-based and especially *Case-based recommenders* (More details on CBR and Case-Based recommenders can be found in sections 2.4.1 and 2.4.2, respectively) have mainly emerged as an alternative to CF recommenders intending to overcome their shortcomings, while efficiently handling the current information overload (Lorenzi & Ricci, 2005). As knowledge-based RSs have functional knowledge about the relationship between a user and an item, they can be used in more complex cases. Case-based recommenders implement a type of content-based recommendation that relies on a structured representation of problems as cases, usually as sets of well-defined characteristics with their values. These systems generally search and recommend items similar to those that the active user has described in his/her request based on previous similar requests that have been satisfied (R. D. Burke, 2004).

2.2.6 Semantic Analysis and Probabilistic Models

Recently, *semantic analysis*, *latent factors* and *probabilistic topic models*, arising from the area of Natural Language Processing (NLP), and afterwards applied to information retrieval, have also been applied to RSs, starting from tag recommendations. The basic idea behind those models is that people when writing a document have specific ideas in mind that can be expressed by including topics related to these ideas in the document. Topics are seen as sets of words coming from an existing vocabulary. Therefore, depending on the writers' intentions, documents are formed as probability distributions over the existing topics.

These techniques, aim to reveal the underlying meaning of words and documents based on the structure of their co-occurring patterns in data corpora without the support of additional knowledge bases. These techniques have been found able to reach higher accuracy than some of the commonly used techniques while better handling sparsity problems, (Steyvers & Griffiths, 2007). As Latent Dirichlet Allocation is among the techniques used in the development of the RS of this thesis, more details on latent analysis and probabilistic models will be provided in section 2.5.3.

2.2.7 Hybrid approaches

As *hybrid* is referred any recommendation approach that combines at least two recommendation methodologies. The intention of these techniques is, depending on the application domain, to combine the characteristics of other popular recommendation techniques, in a way that permits them to use their important functionalities, in order to overcome the limitations of each one and generate improved recommendations (R. Burke, 2002). In general, hybridisation techniques may be:

- *Weighted*: the scores generated by the various recommendation techniques are combined to generate a single recommendation
- *Switching*: the systems switches between the different recommendation techniques based on some criterion
- *Mixed*: the recommendations generated by the different recommenders are presented together
- *Feature combination*: the features coming from different sources are used together in the same recommendation model
- *Cascade*: one technique refines the recommendations generated by another
- *Feature augmentation*: the output of one technique being used as an input feature to another
- *Meta-level*: the model generated by one technique are used as the input of another

From the above list, the first four combinations are *order-insensitive*, as the order in which these are applied does not change the result while the last three are *inherently ordered* as the recommendation techniques are applied sequentially, thus the relative order in which they are applied differentiates the final result.

In addition, in recent years the application of various novel approaches, arising or being influenced by other domain specific research paradigms, have been proposed and tested. For example, due to the evolution of mobile devices and the use of recommender systems in applications that highly depended on the location or the context in which those are performed, mobile and context-aware recommenders have lately received increased attention (Ricci, 2010; Adomavicius & Tuzhilin, 2011; Braunhofer et al., 2013).

2.3 From Items to Collections

As it can be seen from the presentation of the main recommendation techniques that took place, these techniques focus on past user-item interactions aiming mainly to improve the accuracy of the predicted matchings. Usually, this kind of interactions are modelled based on past selections, user similarities and item characteristics, to generate as output a ranked list of the top-N items most likely to satisfy a user.

However, in many cases more than a single item, or various alternatives among really similar items, what a user would need in order to fulfil a specific need, is a *collection of items* (also referred to as set, list, compilation or package), similar to playlists (Cunningham et al., 2006; Bonnin & Jannach, 2013).

For example, when a user has bought a computer that she liked, more than recommending more similar computers or the ones that similar users liked, the user would be interested in receiving recommendations for complementary products, like maybe an external hard disk, headphones, etc. When searching for travel recommendations, more than generating recommendations for similar destinations, when having selected a destination a user may be interested in organizing the trip, so that more than getting a list of the 10 most famous museums in a city, the set of a museum, a nice caf or restaurant and some outdoor activities close to it, may be more useful. Similarly, a user that in general likes beer would find as more useful the recommendation of a good white wine when having fish for dinner, more than other brands of beer. In some cases, like in the market basket analysis, where users tent to purchase together some item groups with higher frequency than others, sometimes preferences and association among item groups may seem obvious due to the type of the selected products (for example buying coffee and sugar), while in other cases, it may be difficult to identify the underlying relationships and the rationale behind the joint selection of some product groups (for example a high number of men buying beers and diapers on Thursday afternoons has been observed in US) (Cavique, 2007).

The importance of item collections for user satisfaction, as the above examples highlight, was first mentioned about ten years ago, but still little research has been done towards the recommendation of item collections. Even in cases that the recommended items are presented as a set, in the majority of cases, these items have not been selected or designed in order to work well together, but rather form alternative solutions for the same problem. Lists, forming more than simple aggregations of item recommendations coming from the top-N list, have been found to deliver an added value to the user. Therefore, although in some domains the human factor is the most important in order to construct a collection of high quality, adding automatic reasoning and support to this process could further improve the quality of the resulting sets and speed up the construction process (Hansen & Golbeck, 2009).

Hansen and Golbeck define a *collection recommender* as a system that *"recommends compilations of at least two items"* where each compilation is used and evaluated as a whole. They propose a general domain independent framework for item collections that defines the following four basic features, in order to properly characterize a collection (Golbeck & Hansen, 2011):

- *Unit or Selection*: Unit collections are designed to be treated as being one item as a whole, (like playlists) while selections exist as a set from which a user can further select a subset to use (like libraries).

- *Ordered or unordered*: depending on the importance of the items' order in them, for example order is generally considered as important in a playlist while not in the market basket.

- *Constrained or unconstrained*: refers to domains where constraints on specific values have to be met in order the collection to be considered as useful (like in a medical diet).

- *Finite or infinite*: Although in practice there are no infinite collections, this attribute refers to whether they have a fixed, small enough size so that it could be considered at once or are designed to be of an increasing size. For example a playlist is finite, while a streaming service starting from this playlist and sampling music continuously could be considered as infinite.

In addition, they define the following list of attributes that could be used for properly evaluating the quality of a set:

- *Individual item values*: forms the basic focus of item recommender systems and are also important in collections. However, although users' preferences on individual items are important, the tolerance on a lower item values is higher in collections.

- *Order interaction*: may be absolute or relative. More specific, absolute refers to placing specific items at given positions, while relative placement refers to the ordering of the items within a collection.

- *Co-occurrence effects*: the interaction of items within collections. Independently of being ordered or not and of the rates of each item, some items may not work well together. This attribute is mostly important for unit collections and is among the factors that may define the success or the failure of a collection.

- *Size*: refers to the number of items in a finite collection and generally depends on the domain and the purpose. Even if all the items in a collection work well together and their order is good, a very small set may result being insufficient, while the opposite could be tiring (a playlist consisting only of just three songs would be considered as too short.)

- *Item distribution*: within a collection items should follow given patterns to make it pleasant and successful, in terms of the diversity of items in it, its coverage and balance. More specific, *diversity* refers to the variety of item styles in a collection in order to capture different user's interests, while *coverage* refers to the subset of item styles included in the collection. *Balance*, on the other hand, refers to the item distributions in categories.

For instance, when referring to playlists, among those features, the authors report as more important the songs individual values, their co-occurrence effects on a pair-wise and larger scale, as well as their order and items' distribution in them.

2.3.1 Initial Approaches

Although various cases of complementary items that, when consumed together, the value of each item is increased due to their combination have been reported, still the majority of sites referring to item collections mainly support their manual continuation rather than generating proper recommendations completing the started collections (Golbeck & Hansen, 2011).
The initial approaches to sets' composition and recommendations treated the problem mainly through:

- *Rule-based approaches* with aim to discover interesting hidden patterns, and frequent associations among items purchased together (Cavique, 2007; Mild & Reutterer, 2002; Chen et al., 2005).
- *Graph-models* to capture the structure of usual connections and joint selections among items (Interdonato et al., 2013; Guerraoui et al., 2017).
- *Cost constraint optimization*, aiming to compose the optimal package from the existing item alternatives (Xie et al., 2012).

For instance, searching for frequently purchased together items based on a set of past transactional data, using the Apriori algorithm (Agrawal et al., 1993) would lead to a set of association rules of the form $\{a, b\} \rightarrow \{c\}$ interpreted as *"if a customer has bought $\{a, b\}$ he probably will also purchase $\{c\}$"*. Therefore, when implementing a recommender system based on such rules, the item c would be recommended to a user that has already selected a and b based on their observed correlation (Cavique, 2007).

Guerraoui et al. use the term *consumed item pack (CIP)*, as a notion of a higher level abstraction of item sequences consumed by a user (Guerraoui et al., 2017). Furthermore, they model the problem data as an undirected graph, with the items being the vertices where an edge exists between two of them, if some minimal number of users have consumed both items in a short interval (being two or three continuous consumption logs). Further they apply a community detection algorithm and study the structure of the network by the evaluation of its modularity. This evaluation leads to the discovery of some communities densely connected by specific latent features, that however, cannot be reduced simply to the genre of the items (for example movies or music genres). In addition, a smaller distance among some of the observed communities is found.

Interdonato et al. also use a graph-based approach and refer to recommendations of groups, or sets, of items of various types as *package recommendations* and address this problem through a graph-based approach (Interdonato et al., 2013). However, the proposed model still focuses on user preferences, rather that package patterns, expressed through predicted ratings over the packages based on the items belonging to each and the ratings users have assigned to those. In order to generate package recommendations, first a package recommendation network is built, being the probability distribution over the existing sets of packages, for any user, based on which then the transition probabilities and the packages ranking, are defined.

Xie et al. inspired by real life examples like travel planning services and tweeter following selections, consider composite recommendations that consist of sets of items (Xie et al., 2012). They consider that every item has a rating and a cost, while each user has a bounded budget that can be offered for the finally selected item set. Their scope is to identify the top K packages as recommendations to address the above problem, that is NP-complete. However, they treat the problem mainly as a constraint-based cost optimization problem rather than pure recommendation.

2.4 Other Methodologies

2.4.1 Case-Based Reasoning

Case-Based Reasoning (CBR) is a problem solving paradigm closely related to the human way of reasoning and acting in everyday situations, when facing new problems. CBR in order to solve new problems, uses old experiences that adapts to current situations, based on the following sentence, known as the CBR assumption, *"Similar problems have similar solutions"*. Several results of cognitive and psychological research have confirmed the claim that CBR simulates this type of human problem solving behaviour, particularly seen in early learning (López de Mántaras, 2001; Richter & Weber, 2013).

A situation experienced in a way that it has been captured and learnt, is referred to as *past/previous case* and is stored in the case base. A new situation asking for solution forms the description of a *new/target* case. An important part of the CBR methodology is its learning ability that comes as a natural result of its problem solving process, as the case base is updated each time a new experience is obtained from a problem solution. This knowledge may be reused when needed without implementing the whole process from scratch, or highlight a methodology that should be avoided in a similar case. This way, case-based reasoners are able to use the knowledge gained through previous cases, as well as to improve their problem solving performance over time through new experiences. In addition, as these techniques derive their reasoning from complete cases rather than decomposing then into simple rules, they are more appropriate for domains with limited understanding or high complexity where rule-based techniques usually fail to provide an efficient solution (Aamodt & Plaza, 1994; Leake, 1996).

In CBR, a case is not a rule. A case denotes a problem situation in a wider term and does not necessarily need to refer to finding a concrete solution to an application. Depending on the domain that is analyzed, cases may be represented as simple collections of attribute-value pairs of specific characteristics that occurred in a situation or, in more complicate domains hierarchical or graph representations may be more adequate. Except from the adequate cases modeling, the parameters that highly affect the performance of a case-based reasoner are its previous experiences, its ability to understand and adapt to new situations, based on those and finally its ability to evaluate the reasoning results and incorporate the feedback of the process (Bergmann, 1991).

In general, a case in a CBR system can be denoted as an ordered pair $c = (p, q)$ with $p \in P$ and $q \in Q$ being the problem description and solution, coming from the sets of the problem descriptions P and solutions Q, respectively. The case base can be defined as the set of all the known cases, therefore denoted as $C = (P, Q)$.

When a new problem, a new case, has to be solved, first of all, based on the comparison of their problem descriptions, the most similar case(s) that have been previously experienced have to be retrieved from the case base. In this step, similarity measures are involved to provide an appropriate ranking of the existing cases based on the appropriate features. The solution is then obtained by reusing the retrieved cases' solutions with a proper adaptation, so that it becomes adequate for the current situation. The solution can be modified either manually by the users wishing to define some characteristics of interest, or automatically by the system based on domain knowledge and solution generators able to adapt the solution to the current requests. The new solution is then evaluated in order to ensure that it can address the initial problem and to verify that its quality and performance will be the expected. Finally, the new experience is incorporated in the case base, providing additional solution knowledge or highlighting a methodology that should be avoided in the future in similar problems (Kolodner, 1992).

The CBR solving and learning process can be described as a cyclical process comprising of four steps, also known as the four REs, or as the *CBR cycle*, that can be seen in figure 2.5 below. Each of those may be further divided into subtasks depending on the domain and the scope of the system (Richter & Weber, 2013; Lopez De Mantaras et al., 2005).

In more detail, the four REs refer to the following steps:

- *Retrieve*: the most relevant previous case(s) from the case base.
- *Reuse*: the information and knowledge provided in the retrieved case(s) in an adequate way for the new problem.
- *Revise*: the solution obtained to ensure that its performance will be the expected.
- *Retain*: the parts of the solution/experience that are likely to be reused (or should be avoided) in the future, and incorporate this new knowledge into the case base.

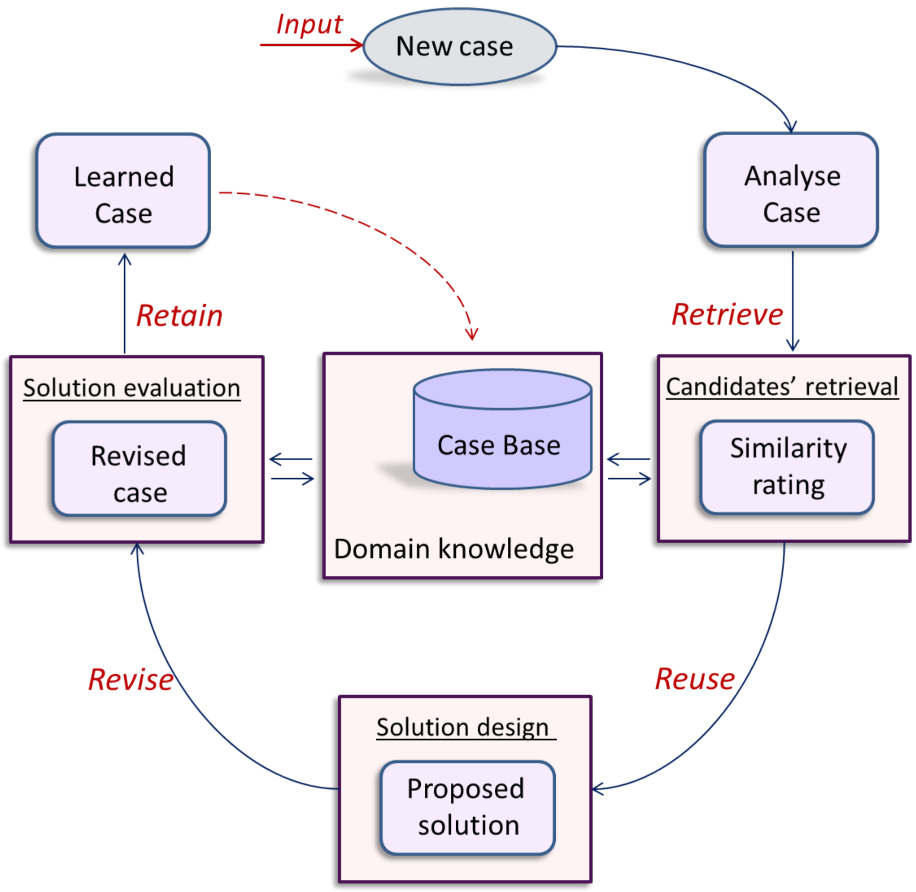


Figure 2.5: CBR Cycle

As it can be derived from the above, a key factor of a CBR system is the similarity function used to identify the most similar past cases from the system's case base. The similarity function is a metric used to estimate the level of resemblance between a new and the existing cases. This metric provides an approximation of the level of utility the solution could provide related to its re-usability, with the intention to provide an approximation as close to the real value of re-usability as possible, while at the same moment being easily computable and interpretable (Xiong & Funk, 2006). As cases may be represented through various ways, depending on the application domain, various are also the similarity metrics that can be used for each of the cases. The values of the similarity function range in the set of $[0 \ldots 1]$, where 0 is assigned to totally different cases and 1 to cases that are regarded as identical according to a given similarity measure.

In general, similarity measures can be defined at a local and global case level. *Local similarity* metrics are used to capture the similarity of the different features that the cases are composed of. On the other hand, *global similarity* metrics provide an overall approximation of the cases' similarity by aggregating local similarities along with appropriate importance factors assigned to each of them. An appropriate similarity (or distance) function able to capture the hidden relationships among the various objects associated, and reflect the level of their similarity (or dissimilarity) with the cases enables the system to provide an increased performance (Liao et al., 1998; Finnie & Sun, 2002).

Let $sim({f_i}^N, {f_i}^R)$ be the *local similarity* function used for the comparison of the *i-th* among the n different attributes that the cases consist of, that have values ${f_i}^N$, ${f_i}^R$ in the new f^N and the retrieved f^R case respectively, or for a distance function being:

$$d({f_i}^N, {f_i}^R) = 1 - sim({f_i}^N, {f_i}^R) \tag{2.3}$$

On the other hand, let $Sim(f^N, f^R)$ be the *global similarity* that can be found as an aggregation of the local similarities. Therefore, if w_i is the importance weighting factor of each of the case's $i = 1 \ldots, n$ features, when using a *k-nearest neighbours* approach, the global similarity can be calculated as in the following equation,

$$Sim(f^N, f^R) = \frac{\sum_{i=1}^{n} w_i \times sim({f_i}^N, {f_i}^R)}{\sum_{i=1}^{n} w_i} \tag{2.4}$$

2.4.2 Case-Based Recommenders and Applications

Lately, the CBR paradigm has been applied to domains further from its traditional knowledge related applications, to areas like scheduling, decision making support, electronic commerce, product retrieval and recommender systems intending to decrease the observed gap between user requirements and item characteristics (R. D. Burke, 2004).

Case-based recommenders follow the general CBR cycle and rely on the core CBR concepts of similarity and retrieval. The recommendation process is instantiated when a new user looking for recommendations accesses the system and introduces his/her requirements. The user request serves as a new problem description while the available items form the possible solutions that can be found in the case base of the system. These items can be retrieved according to their similarity to the user query, computed through an appropriate similarity function. In case-based item recommenders, the inserted parameters/ constraints of a user refer to the item's specifications. Therefore, the level of user request specification heavily influences the outcome of the CBR recommendation process, as both an overspecialized and an underspecified query will most probably result in solutions unable to cover user needs, (Lorenzi & Ricci, 2005; Prasad, 2003).

These recommenders evolved as an alternative mainly to CF in order to overcome the shortcomings that these systems come with. CBR recommenders perform a kind of content-based analysis, usually based on structured representations of cases as sets of attribute-value pairs, and generate recommendations based on the user request and not on the items' ratings. They analyze the request and the constraints imposed by the user and try to find items with characteristics that match these requirements in the best possible way. This existence of a common and structured way of representing both the query and the treated items enables the system in calculating and understanding similarities among items, generating meaningful recommendations, as well as, the evaluating and incorporating the feedback in the process (Bridge et al., 2005). An additional advantage of CBR recommenders is that they can include in the recommendation generation process semantics and characteristics that were present during past item selections.

In recent years, there has been observed an expansion of on-line stores and e-commerce applications selling different types of physical and information goods. Although recommender systems are being used in the majority of these sites in order to enable users in finding the most suitable items, most of these applications lack of proper customer support. CBR has been proposed as the main methodology being able to provide an intelligent product selection support, able to support the request submission and dialog through between the user and the system, selecting the most appropriate products according to the user needs more than recommending the most popular items (Bergmann, 1991). Finally, the item(s) that seem as more appropriate for the user are recommended, sometimes along with a presentation and/or explanation of the reasons that led to this selection. In such approaches, items and cases are considered as identical objects. The requested characteristics describe the target product and the system tries to find the item in the product database (that is the case base of the system) that best matches this description. The effectiveness of the system highly depends on the characteristics selected to describe the cases, as well as on its ability to capture the similarity between the descriptions of the target and the existing products (Lorenzi & Ricci, 2005; Bergmann, 1991).

As in CBR, cases' modelling may differ according to the application domain, complicated domains where more details than simple user ratings have to be taken into account can be analysed and solved by applying an appropriate CBR approach. Another area where CBR recommenders have been successfully applied is in travel planning recommendations. In this domain, both recommendations for destinations or concrete types of activities may be requested, as well as complete travel plans including accommodation and other activities in a selected destination (Ricci et al., 2006).

Ricci and Werthner in (Ricci & Werthner, 2001) refer to those as travel plans and travel services. More than presenting the travel choices as a list of products among which the users are asked to select, the authors propose a CBR recommendation system that starts from a simple destination model and enables the user to update and reshape by setting his/her requirements and constraints. The case base of the system consists of the travel plans previously made by other users satisfying their preferences. In (Ricci et al., 2002) the factors that influence travel selections are divided into two groups, user and item related, named personal (socio-economic and psychological/cognitive) and travel (purpose, length, distance etc.) respectively. The proposed approach is based on a case base of travel bags (coherent sets of travel items without temporal scheduling), follows a hierarchical case modelling of the travel factors while taking into account a human choice model, extracted from the analysis of travellers' behaviour. When building a new travel plan, the system focuses more on previous cases in similar contexts than on previous selections of the active user.

An interesting CBR approach to automatic playlist generations, treating playlists as cases being sequences of items, can be found in chapter 3 that refers to music recommender systems, in section 3.4.2.

2.5 Semantic Analysis and Probabilistic Topic Models

Semantic analysis and *probabilistic topic model* approaches are thought to be able to provide a better insight into human cognition and support explanatory approaches related with fundamental cognitive science questions like language acquisition, learning and processing (Steyvers & Griffiths, 2007; Blei, 2012).

These approaches were initially used to handle problems related to documents' semantic retrieval, as various linguistic studies have revealed that these techniques are able to match human text learning processes, where the meaning of words is defined through the concepts in which these words were present and absent, (Si & Sun, 2009). They aim to discover the signification of words and to find their similarity based only on the similarity of the context within which they appear without using any additional knowledge base. In addition, they follow the *bags of words* assumption, meaning that the semantics are determined only from their lexical level, neither from the words' order nor from the syntactic structure of the sentences in which those appeared.

Due to the shortcomings of the mainly used recommendation techniques, like *cold start* and *overspecialisation*, these techniques have recently gained more ground and have been effectively applied to the area of recommender systems, starting mainly from those related with some kind of text items. One of the major strengths of these techniques is their ability to reveal hidden relationships among the items of a corpus, through the analysis of co-occurring patterns which these recommender systems intent to reveal and use.

2.5.1 Latent Semantic Analysis

Latent Semantic Analysis (LSA) is a knowledge induction and representation theory for the representation of the contextual meaning of words, through the application of statistical computations to large text sets. It was initially developed with the purpose of improving information retrieval and better handling problems related to text semantic meaning retrieval.

LSA is thought to be closely related to neural network models but it is based on SVD that reduces the latent representation space and then ignores the less significant dimensions (regarded as noise or non-essential factors) of the new representation space in order to achieve the best possible dimensionality. This has as a result that, words occurring in similar contexts are matched to more similar vectors and therefore result in a higher similarity rating.

LSA is based on the idea of mapping high dimension vectors into lower dimensional representations in a latent semantic space (Landauer et al., 1998). More specific, LSA uses the assumption that semantic information can be derived by dimensional reduction of the document-word co-occurrence matrix and these documents and words can be represented as points in the Euclidean space. A word document co-occurrence matrix can be decomposed by *singular value decomposition (SVD)* into a word vectors', a diagonal and a document vectors' matrix, like shown in figure 2.6.

LSA induces the words' and documents' meaning only from the underlying text, based on the co-occurring words, with words being considered as similar if they appear in similar documents, while documents are considered as similar to the extent that they contain similar words (Landauer et al., 1998). The word representation is close to a kind of average meaning of the contexts that it appears in, while a document is regarded as an average of the meanings of words present in it. However, these characteristics may be regarded as both LSA's strengths and weaknesses depending on its application domain and purpose.

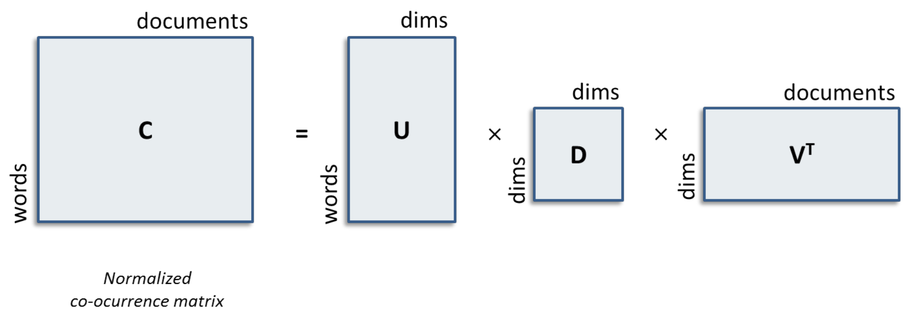


Figure 2.6: SVD performed by the LSA

2.5.2 Probabilistic Latent Semantic Analysis

Probabilistic Latent Semantic Analysis (pLSA) initially known as *Probabilistic Latent Semantic Indexing (pLSI)* is a statistical technique that defines a generative data model for the analysis of two-mode and co-occurrence data, with applications in information retrieval, natural language processing, computational linguistics, machine learning from text and other.

pLSA aims at identifying and distinguishing among different significations of words, according to the context of their usage without the use of a dictionary. It can be regarded as a statistical view of LSA that associates a latent context variable with each word co-occurrence and was the first probabilistic approach towards modelling text documents. The starting point of pLSA is a statistical model called *aspect model* (Hofmann, 2001) a latent variable model for data co-occurrence that associates an unobserved class variable with each observation of a word' s occurrences in a particular document. This model was first proposed for language modelling, where it is referred to as aggregate Markov model, based on the assumption that observations or sets of observations come from an underlying latent class.

Although highly influenced by LSA, pLSA is an *unsupervised learning method* that uses a generative latent class model in order to perform a probabilistic mixture decomposition. Furthermore, pLSA defines a proper generative data model, enabling the use of standard statistics techniques for the model fitting, selection and control, and was found to improve the results achieved by the standard LSA (Hofmann, 1999).

2.5.3 Latent Dirichlet Allocation

Topic models are models of probabilistic nature that aim to reveal the underlying semantic structure of large sets of documents. They are based on the hypothesis that a document consists of various topics, and each topic is a probability distribution over words. Therefore a topic model is a *generative model* for documents that specifies a probabilistic procedure by which documents may be generated.

The basic idea behind their application is that a person writing a document has specific ideas in mind that can be expressed by selecting various topics related to these ideas, and specifying a distribution over them. The words that will appear in the document are selected from the topics in the generated topics' distribution. Based on this approach, documents with different content can be created by selecting different topic distributions (Krestel et al., 2009). These algorithms are based on statistical methods and analyse the words within the documents, in order to discover the themes that run through them, the way they are connected to each other, and how do they change over time. The topics emerge from the analysis of the given documents, without the necessity of prior knowledge of the topics in text corpus or previous annotations of the text, while the topic distributions arise by computing the hidden structure generated by the collection of documents in the corpus.

Latent Dirichlet Allocation (LDA) is a generative probabilistic model for collections of discrete data, such as text corpora (Krestel et al., 2009). It is a three-level hierarchical Bayesian model where items are modelled as finite mixtures over an underlying set of topics. Therefore, in the case of applied to text modelling, the representation of a document is provided explicitly by the set of its topic probabilities (Blei, Ng, & Jordan, 2003). Apart from the *bag of words assumption*, LDA works based on the assumptions that the number of the existing topics in a corpus is fixed and known, and that the order of the documents in it is not important, that may not be suitable for corpus composed of document collections that vary through years (Hofmann, 1999; Blei et al., 2003).

In *generative probabilistic models*, the data are treated as arising from a generative process with hidden variables that define a joint probability distribution over both the observed and the hidden variables. This distribution is also called *posterior distribution.* The observed variables are the words of the documents while the hidden variables are the *topic structure.* The computational problem that arises refers to inferring this hidden topic structure, therefore to compute the posterior distribution, the conditional distributions of the hidden variables given the documents.

A *topic* is defined as a distribution over a fixed vocabulary, while for each document the words are generated in a two stage process, first a distribution over topics is randomly chosen and for each word in the document a topic of the above distribution is chosen and a word from the corresponding distribution over the vocabulary (Steyvers & Griffiths, 2007; Blei, 2012). Therefore, the word-document matrix is split into two parts, rather than three like in LSA, the word-topic and the topic-document matrix, each representing the corresponding probability distributions, as it can be seen in figure 2.7 (Steyvers & Griffiths, 2007).

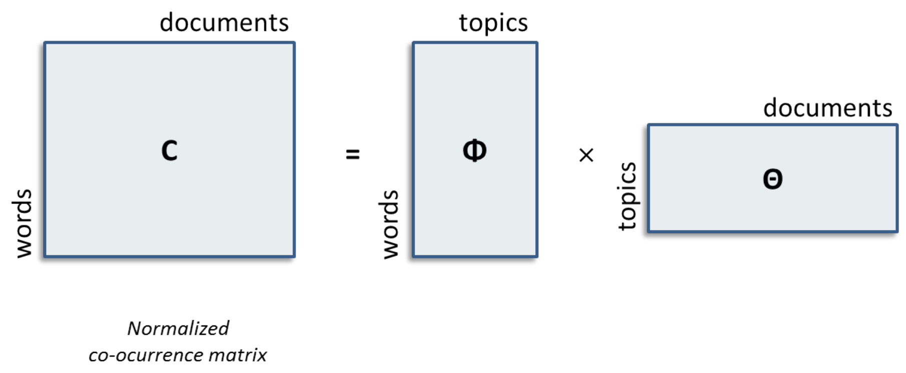


Figure 2.7: Matrix factorization performed by the the topic model

Problem Formalization

LDA, although not restricted to text data applications, as it was initially related with those, uses the terms "words", "documents" and "corpora" for the problem entities. The problem can be formalized as follows:

- A *word* is the basic item unit coming from a given vocabulary. It can be represented as a vector with a single component equal to one and the rest of the components equal to zero, as:
$$w = \{0, \dots, 1 \dots, 0\} \tag{2.5}$$
- A *document* is a sequence of N words, with w_n being the n-th word in it, denoted as:
$$\mathbf{w} = \{w_1, w_2, \dots, w_N\} \tag{2.6}$$
- A *corpus* is a collection of M documents denoted as:
$$D = \{d_1, d_2, \dots, d_M\} = \{\mathbf{w_1}, \mathbf{w_2}, \dots, \mathbf{w_M}\} \tag{2.7}$$

Based on those,

- $P(w_i/d)$ is the probability of the i-th word in document d.
- $\theta^{(d)} = P(z)$ is the multinomial distribution over topics z in document d. Therefore $P(z_i = j)$ is the probability that the j-th topic was sampled for the i-th word token.
- $\varphi^{(j)} = P(w/z = j)$ is the multinomial probability distribution over words w for topic j, thus $P(w_i/z_i = j)$ is the probability of word w_i occurring under topic j.
- For a number of topics equal to T, the model specifies the distribution over words within a document as:
$$P(w_i/d) = \sum_{j=1}^{T} P(w_i/z_i = j) P(z_i = j/d) \tag{2.8}$$
$$P(w_i) = \sum_{j=1}^{T} P(w_i/z_i = j) P(z_i = j) \tag{2.9}$$

Comparison

LDA extends pLSA by specifying the way θ is generated. For a given number of latent topics, LDA estimates the topic-word $P(w/z)$ and the document-topic $P(z/d)$ distributions by using Dirichlet priors. More precisely, given the observed words w, the *Gibbs sampling algorithm* is used to estimate the posterior distribution over z, as it is considered an efficient method for extracting sets of topics from large corpus (Steyvers & Griffiths, 2007; Krestel et al., 2009; Si & Sun, 2009). In addition, the performance of LDA when assigning probabilities to new documents has been found higher and with a lower perplexity than pLSA that works based on the *Expectation Maximization (EM) algorithm* (Blei et al., 2003).

On the other hand, LDA differs from LSA in the representation part, as it expresses the semantic properties of words and documents through probabilistic terms, which enables the interpretation of each topic. Furthermore, two words are thought to be similar to the extent that they appear in same topics and two documents are thought as similar to the extent that same topics appear in them. The increased utility provided by topic models comes from their ability to resemble the thematic structure of a data collection by the inferred hidden structure (Blei, 2012).

2.5.4 Topic Model Recommenders and Applications

Having extracted the set of topics from a corpus, in order to find the similarity of new concepts, words or documents, the similarity of their topic distributions is calculated using an appropriate similarity metric. Thus, for recommendation applications, given a new query, being a document, its topic distribution is found and then similarity computations based on the topic distributions of the documents in the corpus, take place in order to find the most similar ones. Recommendations are then generated from the top words of the main topics of the most similar documents found.

Tags coming from a "tag cloud" can be used to characterize documents and multimedia items, like photos, videos, songs. The items can be then represented through their tags and latent models may be used in order to infer the tags' topic distribution of each item. Based on the similarities of the extracted topic distributions, items coming from the set of items with the most similar topic distribution, are recommended (Krestel et al., 2009). Similarly, an article recommender system based on the Wikipedia corpus is presented in (Haruechaiyasak & Damrongrat, 2008). The advantage of this approach is that is able to discover articles with relevant topic profiles, coming from different fields/subjects, which would not be reached through the existing hyperlinks and keywords, and would not be recommended through recommendation techniques that focus on the user ratings.

Recently, these approaches have been applied also to market domains of both physical and electronic stores. In (Christidis et al., 2010), a topic model recommendation approach for the market basket analysis problem is presented. In this approach, items coming from different item topics are treated like words placed with some probability within different market baskets, that can be seen as documents created of these items. The baskets are compared based on their topic probability distributions, and when a new user having already placed some items in his/her basket comes, the basket's topic profile is compared to the past ones. The items coming from the baskets with most similar topic distributions, having not yet been selected are recommended. Other topic model recommenders for market analysis that additionally insert into the model the prices of the purchased items can be found in (Iwata et al., 2009; Iwata & Sawada, 2013).

2.6 Conclusions

In this chapter, the first part of the theoretic background of this thesis, has been presented. More specific, the basic information on RSs, their basic assumptions, categorization and common limitations have been described.

A summary of the basic characteristics of the commonly used recommendation techniques, along with their main limitations and advantages, can be found in the following table:

RS	**Core**	**Input/Output**	**Limitations**	**Advantages**
CF-NN	Nearest neighbours	user ratings/ topN items	cold start popularity bias	commercially successful
CF-MF	Latent vectors	user ratings/ topN items	missing data modelling	commercially successful
CB	Item attributes	user ratings/ topN items	overspecialisation limited diversity	specialized domains
AR	Rules	antecedent/ consequent	computation cost eval. difficulty	co-occurring patterns
Utility	Optimis. function	user request/ topN items	limited personalization	no cold-start or sparsity
CBR	Similarity function	case description/ case solution	data quality dependence	learning ability handle complex.
LSA	SVD	documents/ topic distributions	information loss	reveal hidden relationships
LDA	Dirichlet priors	documents/ topic distributions	less stable performance	reveal hidden relationships

Table 2.1: Recommendation techniques' basic characteristics

Special emphasis has been placed on CBR and LDA as these are the techniques that have been further used in the implementation of a hybrid RS for automatic playlist continuation. Therefore these techniques have been presented in more detail. In addition, the necessity of implementing techniques that generate recommendations of item sets, being more than a ranked list of alternatives, has been presented along with the basic characteristics that these systems should take into account.

Chapter 3

Music Recommender Systems

In this chapter, first of all, some of the unique characteristics of music items and their consumption are presented. Then the basic concepts of Music Recommender Systems (MRSs), Music Information Retrieval (MIR) and automatic playlist generations (APG) are described, as automatic playlist continuation (APC) forms the basic application domain of the developed recommender system. Moreover, due to the unique characteristics of music items that complicate their recommendation processes, additional characteristics that influence users' music selections are presented. More specific the definitions, modelling and ways of supporting beyond accuracy parameters and contextual factors related to the user perception of playlist quality, and satisfaction, can be found.

3.1 Music Items and Consumption

Although music is present in everyday life of a lot of people, the human perception of music, the rationale behind music selections, and their emotional influences still remain undefined from a cognitive perspective.

The recent capabilities of digital music production through specialized software programs have enabled the creation and public distribution of music, without geographic limits and with lower costs, creating new opportunities for both artists and users. New, and yet less known, artists manage to produce, distribute and introduce their work to a larger audience without having to overcome the financial limitations and entry barriers of the past, usually set by record companies (Logan, 2004). On the other hand, along with the shift from CDs to mp3 formats, this has resulted in an increased amount of music items and information about them, being publicly available online for the users (Domingues et al., 2013). In addition, a change in the way that music is "consumed" has been observed. More than "owning" songs in their personal collections, users now tend to listen to them online (Schedl, Knees, & Gouyon, 2017). Furthermore, this has created numerous issues related to music access, discovery and navigation, as well as sharing and information services that need to be facilitated. There is a need for new systems, and communities, enabling users in finding new music items, or accessing specific music they are looking for, and may have difficulties in discovering. This way, like in other areas facing information overload problems, music recommender systems have evolved in order to handle these issues (Celma, 2010).

When referring to music items, those can be songs, genres, artists, albums and radio stations. Therefore, music recommendations can be addressed at different levels of abstraction, grouping items by genres, artists or albums (Schedl et al., 2015). As these items form the result of a complicated synthesis process, their analysis, in term of content characteristics, requires more in-depth domain knowledge. In addition, when it comes to the music recommendation problem, there are some unique characteristics of this domain that differentiate it from other item recommendation domains, and can be described as (Schedl, Zamani, et al., 2017):

- *Item duration*: The short *consumption time* of songs (usually being of some minutes) compared to movies (hours) or books (days or weeks), that makes them more disposable. It is also closely related to the time during which the user creates an opinion on the item.
- *Magnitude of items*: The *size of the item catalogues.* In contrast to their smaller size, per item, in comparison to other multimedia items, the number of the totally available music items is much larger, being of millions of pieces. Therefore, support for scalability becomes of higher importance.
- *Repeated recommendations*: A music item could be *consumed multiple times*, even repeatedly, while other multimedia items usually are consumed once or at most a few times. Therefore, in music recommendations, users may accept and even find useful recommendations of already known items, in a different or later time moment.
- *Sequential consumption*: Songs are rarely listened to in isolation (McFee & Lanckriet, 2011). Users rather tend to create *sessions* that consist of sequences of songs, placing more importance on the songs within a sequence and their relative order.
- *Consumption behaviour*: Music may also be consumed passively, meaning that the user is performing other activities while having music in the background. As a result, no implicit feedback or information on user preferences may be retrieved from the user's behaviour, as there is no direct interaction (skipping or liking songs).
- *Listening intent and context*: Various dimensions related to the music consumption moment and scope, like the performed activity, mood, location, time, company, etc., during the consumption moment may heavily influence the music item selections.
- *Emotional connotation*: Music is well known to evoke emotions while at the same time the users' music needs are influenced by their actual emotional situation and mood.

Pachet and Roy (Pachet & Roy, 1999) define the three main goals behind music selection and consumption as *repetition, surprise* and *exploitation of catalogues.* The "desire of repetition" is related to users' musical perception and their tendency to listen to songs they know and like, or similar to those. On the other hand, the contradictory, "desire for surprise" is related to their tendency to discover new music, in terms of songs, bands or music styles. Therefore, the key issue is to manage presenting the users music that they do not know yet, but it is similar to what they like. Finally, the exploitation of catalogues is more related with the distribution of music items within large collections in a manner that they would be accessible and also coherently organized. To this direction, for a MRS to be efficient, and deliver recommendations of improved quality, the special characteristics of music items should be understood and supported, in order to balance among the various dimensions of music consumption (Celma & Lamere, 2011).

3.2 Music Recommender Systems

Music Recommender Systems (MRSs) are a special category of recommender systems, with aim to filter and reduce the information overload related to music. These systems have their basis both in fields of recommender systems and classical multimedia information retrieval and especially in the field called *Music Information Retrieval (MIR)*, (Kim et al., 2010). They retrieve and present to a user the music items that seem more appropriate for him/her, deducted from his/her profile or based on a specific request. Therefore they have to deal with the common limitations of both areas (Casey et al., 2008; Byrd & Crawford, 2002).

Furthermore, one of the main issues of MRSs, as in MIR, and multimedia information retrieval systems in general, is the *semantic gap* between the items' content description and the users' perception of these items (Celma et al., 2006). As most of the MIR techniques use low level signal characteristics, they lack of ability to capture aspects related to music tracks that are closer to the human perception, while users tend to provide their queries through high level descriptions. Kaminskas and Ricci outline the necessity of finding systems able to represent music items at a higher representation level closer to human perception in order to provide conceptual explanations and be easier understandable by the users (Kaminskas & Ricci, 2012).

Traditional MIR techniques use content-based audio related techniques that apart from the general limitations of CB systems, like overspecialization and limited diversity, require a deeper knowledge of the application domain (Casey et al., 2008; Lamere, 2008). On the other hand, CF techniques focus mainly on isolated user ratings to predict the ratings a user would assign to a previously unknown song, based on the behavior of similar users. Thus they heavily suffer from cold-start and long tail effect limitations (Su & Khoshgoftaar, 2009). In addition, the majority of music recommender systems show a tendency towards the recommendation of popular items, and do not take into account the situational and other contextual parameters under which the selection of an item is done, which may heavily affect the final result (Park & Tuzhilin, 2008; Levy & Bosteels, 2010).

CF techniques are domain independent, thus do not take into account domain specific attributes. These techniques try to predict the ratings a user would assign to a previously unknown song only based on the ratings similar users have assigned to the specific song. As in their other application areas, these techniques are restricted by limitations like the cold-start, data-sparsity and long tail effect (Herlocker et al., 2004). An example of a well-known CF music recommender is last.fm[1] that creates personalized music recommendations starting form an artist selection from the user side. Its main drawback is that it is not able to recommend novel, not popular music. On the other hand Genius, the recommendation software introduced by iTunes[2], starting from a given song generates a playlist for a user from his/her library by comparing this library with the libraries of other iTunes users.

Apart from CF approaches that follow their neighbourhood based reasoning, and do not enhance the specific characteristics of the music domain, the other recommendation approaches used mainly focus on content and context information related to music, and to its consumption in order to generate recommendations.

CB approaches for MRSs in general require a deeper knowledge of the application domain. Therefore, given the complexity of the music items and the related information, their application is far more complicated (Shao et al., 2009). Content information on music items can be mainly of two categories, related to *audio and musical features*, like timbral, temporal and time-domain and tonal features, and *semantic annotations*, as distributions over tags, automatically extracted using machine learning techniques (Celma, 2010). Usually, in CB recommenders music items are described mainly through their low level characteristics, using the retrieval and analysis methods that will be presented below. These techniques face the general limitations imposed by CB techniques, like the overspecialization of the recommended items, and additionally, they are more exposed to the *semantic gap* of music information analysis. One of the most popular CB music recommenders is Pandora[3], which provided a song or an artist name, generates recommendations of songs that have similar characteristics with the introduced seed song. The comparison is done based on music features of the songs like tonality, instrumental arrangement and other (Shao et al., 2009).

On the other hand, metadata can be of various types, like manual annotations provided by experts, usually following some kind of taxonomy (editorial metadata including genre, artist, release year, etc.), social tags coming from user generated tags from social tagging services usually being more varied but also more noisy, or web annotations like keywords extracted directly from related web pages (Schedl et al., 2015). Bogdanov and Herrera evaluate the quantity of metadata needed in order to generate music recommendations of high quality. As especially user-generated metadata may insert additional noise, they propose the use of content-based information extracted from the music items combined with the minimum amount of genre related metadata. Based on the conducted listening experiment, while using different distance metrics, they suggest this method to be suitable also for recommendations in the long-tail (Bogdanov & Herrera, 2011).

[1] last.fm
[2] www.apple.com/itunes
[3] www.pandora.com

3.3 Music Information Retrieval

Traditional music information retrieval techniques use audio content analysis and describe music items based on low level characteristics. An audio signal analysis process is performed in order to extract the basic information about tracks. In order to retrieve and recommend music items based on a given query, the classic MIR methods use three approaches (Kaminskas et al., 2012):

- *Query by example* uses an audio signal as input while the expected output consists of the metadata related to this input audio, like title, genre, artist, etc. This methodology uses fingerprinting techniques that represent an audio as a unique set of low-level audio features. A widely known system based on this paradigm is the Shazam[4] music recognition application, which generates an audio fingerprint from 10 seconds of a recorded audio, and then searches its database of audio fingerprints for matching fingerprints and their related metadata.
- *Query by humming* takes as input a melody sung by a user and intends to find the alternative versions of a song. This kind of retrieval systems is suitable only for melodic music, as the input provided by the user is monophonic. As the majority of the existing popular west music is polyphonic, these recommendation techniques have first to retrieve from the database the various individual melodies that compose the songs included in the database, in order to be able to match them against the query.
- *Genre classification* is rather a classification process, and not a search methodology like the previous two, with the main audio signal features used for classification being timbre, melody and rhythm. The problem with this technique is the absence of a defined taxonomy of music genres. Existing libraries use different specifications, and each one has its own hierarchy in order to specify music genres. In addition, as music genres evolve, the problem becomes even more complicated.

Furthermore, when analysed regarding the level of retrieval task that they perform, MIR systems can be divided into four main categories, namely genre, artist, work and instance level. *Genre level* refers to searching for types of music, like pop, rock, etc. while, *artist level* refers to searching for similar artists to a given one. For instance, search at *work level* refers to looking for cover songs of a given song, while finally *instance level* search refers to identifying a specific song (Casey et al., 2008).

3.4 Playlists

Music items are rarely consumed in isolation but rather as sequences, like CDs or radio programs with specific properties as a whole, aiming to create a particular atmosphere (Pachet & Roy, 1999). Therefore, playlists can be defined as sets of music items designed to be consumed as a sequence, similar to traditional radio broadcasts (Bonnin & Jannach, 2013; Cunningham et al., 2006).

Bonnin and Jannach state the difference between playlists, that are arranged in a certain order, and song sets, or mixes, where songs are shuffled (Bonnin &

[4]www.shazam.com

Jannach, 2014). Playlists can be of several types, depending on their scope (target characteristics) and origin/basis (music dataset on which they were made), mainly belonging to the following categories:

- *Broadcasting radio playlists*: made by professionals in radio stations, usually being relatively homogeneous based on a certain audience and often contain popular (for the specific audience) tracks.
- *Personalized radio playlists*: generated by web music services, after users' interaction with those. Thus, their characteristics depend on user preferences and goals (for example discovery).
- *Amateur playlists*: made by music enthusiasts. Their characteristics depend on the creation scope (for example playlist for work, travel etc.) and each user's music knowledge.
- *Club playlists*: made by djs in order to be reproduced at clubs and usually contained danceable and popular songs.
- *Album tracklists*: sequences of songs of one or a group of artists, usually made by labels and artists.
- *Compilation tracklists*: similar to album tracklists but made by popular songs grouped around a given theme, or artist.

Therefore, when it comes to music consumption, more than simply predicting whether a music item would be highly rated, the underlying structure of *joint selections* should be evaluated. Furthermore, the presence of a song within a concrete concept should be captured in order to recommend sets of songs designed to be consumed together, satisfying at the same time *cognitive properties* like relevance, variety, coherence and diversity (Schedl et al., 2015). *Variety* refers to not repeating the same song or artist in the recommended sequence (or at least not often), while *coherence* refers to the order of the items within the sequence, as their changes should be smooth without destructing the user listening to them (Maillet et al., 2009).

Although a user may enjoy listening to songs of different genres or of different tempo, depending on his/her current mood, when generating a playlist like a professional dj would do, the selected songs should not be of the same artist, and more than just belonging to the same genre, should have some coherence and their changes should be smooth without destructing the users listening to them (Maillet et al., 2009). In general, song order has been reported as being less important when treating playlists built on popular items in contrast to playlists created from long tail items (Celma & Cano, 2008).

On the other hand, a large set of songs being almost equal in terms of melody and bit rates would result in the users' boredom. The *lack of diversity* in a playlist has been reported among the common limitations of the majority of automatically generated playlists. Finally, as important criteria influencing a user's opinion of the playlist quality is the user's familiarity with the songs, the creation intention and its suitability for this specific purpose, as well as personal preferences on individual songs, have been reported. Especially when it comes to songs disliked or hated, those might lead to a negative impression of the whole playlist (Schedl, Zamani, et al., 2017).

In general, a playlist in order to be considered as interesting, should have enough diversity and at the same should maintain enough uniformity between songs to avoid generating disruptive changes in its flow or mood (Andric & Haus, 2006). However, as there are no solid methods combining users' perception of music with sound characteristics, it is difficult to specify the exact characteristics that all the items of a playlist should have. Various techniques have been proposed in order to somehow infer the structure of "good" playlists, and identify the characteristics that the songs played together should have, in order to compose a nice result being pleasant and satisfactory for the listener, usually identifying music items similarities based on the metadata associated with them, or through sound related data (Shao et al., 2009).

Cunningham et al. refer to playlists' and mixes' creation as an "art", and present the basic characteristics that a playlist should have in order to be considered as "nice", according to user experiments and interviews (Cunningham et al., 2006). Playlists may be designed for a specific purpose, forming the background of an activity, thus their theme should be somehow related to this activity. On the other hand, they may be the result of a particular mood, or emotion, of their creator in a given moment, like sadness or depression. In general a good playlist or mix has been found to follow an organizing principle or to have a central theme, transmitting a message or presenting a perspective that would not have be seen otherwise. As basic principles based on which playlists are organized, have been identified their style (that can also be the artist or genre), the activity and mood to which they are related, and their message. Finally, as an additional important feature the songs' order has been identified, as users stated that consecutive songs should have complementary sounds, avoiding both boring repetitions and excessive changes, while special attention should be placed on the first and last songs in a list. Finally, mixes, more than playlists, usually should be of a specific length that in the past used to depend on the medium on which they would be recorded (ex: CD), and are usually defined by some theme.

Pichl et al. (Pichl et al., 2015) set their focus on human created playlists, rather than automatically generated ones, in order to identify the rational and patterns behind those selections. They tried to observe and explain the acoustical differences among playlists, and evaluate whether those could serve users in organizing their music. The authors use the aggregation of the characteristics of songs in playlists to represent the later, and then try to classify the playlists, based on their genre, into playlist clusters with characteristics like instrumental and acoustic, valence and danceability, speechiness and finally tempo and energy. In contrast to automatically generated playlists, users in general were found to listen to playlists of various kinds of music, on average belonging to three clusters, with some even having playlists in all clusters, where some of those were placed closer, representing more common combinations in contrast to others that never co-occurred.

3.4.1 Automatic Playlist Generation

As currently more and more online sites either incorporate some music reproduction into their environment, or focus purely on presenting music sets, automatic playlist generation (APG) and recommendation has emerged as among the interesting issues in the music recommendation domain. The scope is to help users in finding and consuming the music they would like while having a more complete experience, or/and support them when they seek for organizing appropriately their music libraries.

Automatic playlist generation refers to the automatic creation of sequences of music items based on some target characteristics. More precisely, the APG problem can be defined as follows (Andric & Haus, 2006).
Given:

- a large collection of available music pieces of any kind
- a background knowledge database
- some target characteristics for the new playlist

How to use the first two in order to construct the sequence of songs that best matches the target characteristics in order to maximize user's satisfaction.

On the other hand, *Automatic playlist continuation (APC)* which is a variation, or sub-case, of APG, consists in adding a set of music items to a playlist in a way that it would fit with its initial target characteristics. Therefore APC consists in the selection of the most appropriate tracks, among the quantity of the available, and the construction of a sequence of improved quality according to specific characteristics inferred from a started playlist (Bonnin & Jannach, 2014). APC is considered among the current challenging issues in MRSs as it could bring benefits to both listening and creating playlists, and has not yet been studied enough. Moreover, when it comes to user experience, APC would enable listeners continuing their listening sessions after a finite-length playlist, while on the same time, it could support the easier and quicker creation of playlists. The ability to first effectively infer the purpose and style of a started playlist is of high importance, as well as the capability of creating a relevant continuation of improved quality, based on it (Schedl, Zamani, et al., 2017).

However, there are various factors that may influence users' listening habits and as a consequence, the perceived satisfaction of a playlist. Some of the basic parameters influencing users' listening habits have been identified as:

- Education
- Age
- Social and cultural influences
- Individual differences (gender, character, personality)
- Social context
- Time context (mood)
- Environmental factors

In addition, when reffering to *listening habits*, there is a differentiation between *musical taste* and *music preferences*. As musical taste we refer to a person's slowly evolving long term commitment to some music idiom, while as music preference we refer to a person's temporal liking of a particular music content, in a particular period or context. Therefore, the musical taste of a user can be represented through his/her entire music collection, while the current preferences can be depicted through a playlist, that may be, or not, aligned with the style observed in long term preferences (Andric & Haus, 2006).

This differentiation highlights the necessity of developing systems that observe the current tendencies in playlists rather than entire user profiles. Probably, users' musical tastes could be evaluated through the traditional recommendation methodologies, like CF, in contrast to actual music preferences which, may change depending and being influenced by several parameters.

3.4.2 Automatic Playlist Generation Approaches

APG approaches are related with the domains of MIR, in terms of finding and ranking all the items that may address a given query, with RSs, in terms of collection recommendations and with MRSs, as song attributes have to be taken into account and properly evaluated (Bonnin & Jannach, 2014). Furthermore, APG algorithms work based on constraints or hints, being a seed item provided by the user, and can be divided mainly into the following three categories (Andric & Haus, 2006):

- *Constraint satisfaction methods* that aim to generate playlists based on some user entered criteria.
- *similarity heuristics* that given a seed music item, it may be a song or an artist, use some similarity function with aim to identify the most similar ones.
- *machine learning approaches* that use a training set of playlists in order to build a model and based on this to recommend playlists.

Some interesting approaches following different designs and coming from these categories are further presented in this section.

As important issues in music playlists are the co-occurrence of songs and the smooth transition among them, *Markov models* using songs as states, and *association rules* or *sequential patters mining* are among the techniques used in this domain. The usual recommendation approaches treating playlists as users, and comparing them through cosine similarity measures or combined with rank prediction algorithms or content based approaches finding tracks with similar musical features with the seed song, can also be used. The major limitations of these methods are that they are computationally expensive, and sometimes work based on strong assumptions, while their overall effectiveness depends heavily on the type of the used data. Finally, due to the long tail distribution present in the music domain, Bonnin and Jannach propose two popularity-based recommendations approaches that also include some artist information (Bonnin & Jannach, 2013).

Related to the more general item set recommendation problem described in chapter 2 (section 2.3), playlist generations have initially followed similar lines. Therefore, when addressing playlist recommendations as items set recommendations, mainly *graph-based* and *constraint optimization* methods have been proposed. In addition, when trying to specifically address and match the songs order, *Markovian processes* and *sequential rules* mining have been also reported.

For example, Logan refers to the song set completion (Logan, 2004). He uses the term *song set*, rather than playlist, as the order of the songs in the set is not taken into account. Furthermore, these songs are thought as representative of a specific sound that the user is seeking for. Thus, he proposes treating song sets as one long song, and using distance metrics on acoustic characteristics to identify the next song to be added to the set. He supports the use of more audio data related to songs, rather than metadata added to them by experts. Pachet and Roy also refer to music sets, and focus on the specifications that a music set should have, and then, treat those as constraints that must be satisfied (Pachet & Roy, 1999). They treat the playlist generation as a constraint satisfaction problem with explicitly defined constraints, rather than extracting those from the music patterns.

Maillet et al. propose the use of a similarity function that takes audio files as input, and generates the probability of those files to be played successively in a playlist. They use both the songs audio features and their past observations in playlists that have been made by professional radio stations (Maillet et al., 2009). The recommended playlists are then generated using a seed song selected by a user, and the transitional probabilities of this song and the rest, that arise from the similarity model and a tag cloud created by the active user to express personal preferences. Slaney and White also base their analysis on low level sound related features, that when combined, permit them to represent each song as a single point in a multidimensional genre-space (Slaney & White, 2006). The aim of their work is to approximate the diversity of user's tastes and music playlists, as it is considered as an important parameter towards user satisfaction.

Flexer et al. also base their study of APG only on songs' audio analysis, without the use of any kind of metadata (Flexer et al., 2008). In contrast to many playlist generation algorithms that start from a seed song or set of songs aiming to identify the continuation, many times leading to too uniform playlists, the authors take as input a start and an end song and the desired playlist length aim to generate playlists having an inherent sequential order, with smooth temporal transitions while requiring little user interaction. The proposed methodology can be divided into two phases, the first devoted to the pre-computation of song similarities, and the second on the creation of the playlists based on these similarities and the divergence ratios from the start and the end. As a main problem of this method can be identified the lack of a smooth transition, mainly in the middle of the generated playlists.

Ragno et al. present an approach that extracts the similarity of musical objects without measuring the distance between them (Ragno et al., 2005). The authors base their reasoning on the implicit likeliness that can be found in the playlists generated by professionals like music radio stations and djs. Usually, the items placed into those sets together with higher frequency, have some common characteristics like a particular genre, pairwise suitability, relative popularity etc. However, in order to extract this information, a large number of streams of high quality has to be exploited. In order to model the transmissions between musical items, a graph representation is used. Songs are represented as the nodes of a graph where undirected arcs are drown between adjacent songs. The number of times that each adjacency is observed is set as the arc weight. Finally after exploring the complete training set, the resulting graph is transformed into a Markov random field. In order to generate a playlist, a random walk is performed, starting from a given song and using the Markov transition probabilities. This methodology is able to introduce the desired variability into the final playlist.

McFee and Lanckriet consider playlist generations as a natural language processing, rather than an information retrieval, problem, with playlists being constructed as generative models of strings of songs of an unknown language (McFee & Lanckriet, 2011). Therefore, the quality of the resulting lists should also be evaluated using a corresponding evaluation methodology. As the general scope of a playlist algorithm is to improve user experience through music consumption, from the previously proposed evaluation strategies, namely human evaluation, semantic cohesion and sequence prediction, human evaluation seems as more appropriate to capture this notion. However, this evaluation method that consists in direct human evaluation, is being less used due to its practical limitations.

Vall et al. when addressing the problem of automatic playlist generation, also refer to language models, as they use similarly the term “song context” (Vall, Quadrana, et al., 2017). Rather than using this term like in the majority of RSs, when incorporating general contextual information, with song context they refer to the current song and the set of the previous ones. Based on those they try to rank all the song candidates, according to how likely they are to appear next in the list. Due to the popularity bias, they claim that this song context supports accurate predictions of songs belonging to the long tail.

Zhang et al. through the use of an appropriate social tag annotation of songs, propose the use of “tag clouds” for the songs, being treated as the “bag of words” of a probabilistic topic model (Y. C. Zhang et al., 2012). Therefore, a topic model is used to infer the topic distribution of the different songs found in a playlist. Based on this representation, playlists can be created and then compared given the number of songs they contain, and the number of topics in their probabilistic topic model. Their aim is to achieve a reduced dimensionality while presenting meaningfully the recommended items in a playlist.

Baccigalupo and Plaza present an interesting CBR playlist recommendation approach that aims to recommend a varied and coherent playlist based on a seed song and the desired length of the playlist (Baccigalupo & Plaza, 2006). Every playlist is treated as a case, and the relevance among those is computed based on song co-occurrences in them. However, the effectiveness of this approach could be biased by the songs’ popularity, the used sub-lists’ length, and the normalization factors applied. In addition, the recommendation process is slightly distinct from the usual CBR approaches, as the case base of the system contains only problem solutions (entire playlists) without problem descriptions. This methodology does not recommend playlists among the existing ones in the case base, but combines the items of the most similar lists in order to present a new list to the user.

These last two approaches follow some similar research lines with the recommender system presented in this thesis. However, HybA, as it will be further presented in chapter 4 in more detail, is a hybrid RS using both CBR and LDA. It treats music items at a higher abstraction level, in order to better capture the characteristics of the songs that appear in playlists of a specific concept. In addition, it aims to deliver recommendations of playlist continuations of improved quality.

3.5 Additional Dimensions

3.5.1 Contextual Factors

The majority of recommender systems treat the recommendation problem at two dimensions, simply taking into account two entities, namely users and items, and based on those, try to identify the most relevant items for a specific user. They generally ignore all the other information related to the recommendation moment, like time, location, weather, possible user's company or scope. However, this kind of data may provide additional information that highly affects the recommendation results, especially in domains where items associated with emotional dimensions are treated, like in music (Wang et al., 2013).

Incorporating contextual information into the recommendation problem results in increasing the input data dimensions from two to three, namely users, items and context, with aim to better refine the data. However, as the listening context plays an important role on song selections, when correctly evaluated and incorporated into the recommendation model, it may lead to a significant increase of accuracy (Knees & Schedl, 2013; Vall, Quadrana, et al., 2017).

Initially, context was defined as any information referring to the user's location, the people and the resources around him/her and the changes in those elements (Schilit et al., 1994). In computer systems in general, as context, we refer to any information that can be used to characterize the situation of an entity (that may be a person, a place or an object) that is considered relevant to the interaction between a user and an application (Dey et al., 2000).

Various classifications of contextual parameters have been proposed in literature with their differences coming mainly from the application domain. When referring to music consumption, context, initially defined as any *"information describing where you are, whom you are with, and what resources are nearby"* is generally mapped to the user's situation in terms of time, mood, activity and other people's presence, when consuming the music items. It can be categorized according to several criteria and purposes as:

- Fully observable, partially observable or unobservable
- Primary and secondary
- Environmental and user related

Where the primary and environmental criteria can be used to derive the secondary and user related ones, respectively. In music recommendations, as primary, or basic environmental attributes, affecting users' music selections have been identified the location, time and weather, while as secondary, or user-related, are considered the activity, emotional state or mood, social and cultural context(Dey et al., 2000; Kaminskas & Ricci, 2012).

A context-aware, or context-based, recommendation process refers to *the estimation of user contextual preferences, and based on those, on the generation of the most relevant recommendations.* These, depending on which point of the reasoning process the contextual information is taken into account (pre, post, model), can be generally categorized as (Adomavicius & Tuzhilin, 2011):

- *Contextual pre-filtering*: or contextualization of recommendation input, that refers to the selection or construction of the relevant data set that will be further used for the recommendations' generation based on the information about a specific context.

- *Contextual post-filtering*: or contextualization of recommendation output, where an initial recommendation set is generated from the entire data set using the traditional recommendation approaches, and then it is adjusted based on the contextual information of the active user.
- *Contextual modelling*: or contextualization of recommendation function, in which contextual parameters and information are inserted into the modelling technique, and are used as part of the recommendation process, or in order to transform the items into a different dimension.

Explicitly defined contextual information related to playlists may be of different types, like location, time moment, mood, or a combination of those. Pichl et al. propose the use of playlist names to identify implicit contextual information, in order to overcome the necessity of aggregating different types of contextual information. They use playlist names to extract information on user's behaviour, like the playlist creation moment or purpose based on which they create contextual clusters, that are then incorporated into the recommendation process (Pichl et al., 2015). However, the efficiency of this approach heavily depends on the data quality, as sometimes names assigned to playlists may be highly subjective and may lead to sparse clusters that are not valuable for the recommendation process. On the one hand, playlists names containing objective information like "winter", "Christmas", etc., may lead to clusters improving the recommendation accuracy, while clusters of playlist names containing words like "my favourite", "best", etc., do not provide any improvement. In addition, this approach is dataset specific, as in many cases playlists are saved only along with an id or the timestamp of their creation moment, without names or additional information.

Gillhofer and Schedl analyze the relationships between the several dimensions of context, and the resulting user listening habits and preferences, to identify whether contextual information permits the accurate prediction of music items (Gillhofer & Schedl, 2015). Furthermore, they evaluate contextual influences on different categories of music items, namely song, genre, artist and mood, and if so, which are the aspects that relate more and support this prediction. On the other hand, among the contextual attributes evaluated they focus on time, location, weather, device, phone, task, network, ambient, motion, player and activity. From those attributes, device, task, weather and time have been found as the most important, being able to cover almost the same information as all categories when combined. However, due to the data sparsity characterizing music recommendation problems, when addressing music recommendations at song level, the performance was very low, while when it comes to genre or artist prediction, the use of additional contextual information indeed was found to improve the classification accuracy.

As the context within which a music item is consumed, or a playlist is generated, is of high importance, Domingues et al. present an interesting methodology for incorporating contextual, and other additional parameters within a recommendation model. More than performing a pre- or post- filtering based on the actual context, they model contextual characteristics as additional attributes of the treated items, and represent them as "virtual items" (Domingues et al., 2011). These items are included only during the model building phase that can be performed using classical user-item recommendation algorithms, without increasing their variables from two to three. Their results show that these additional dimensions provide extra information able to improve recommendations' accuracy when combined with the usual recommendation algorithms. In addition, their results confirm that contextual modelling may also enable the access to novel, or less popular, but still relevant items (Domingues & Rezende, 2013; Cremonesi, 2009).

3.5.2 Beyond Accuracy Dimensions

Throughout many years, starting with the Netflix Prize, the main focus of the research related with recommender systems was placed on improving the *recommendation accuracy* by better deducting and predicting users' needs. However, as recommender systems evolve their scope, this dimension is not sufficient for capturing user interests and delivering recommendations of added value (Vargas & Castells, 2011). Furthermore, their scope has moved towards the improvement of the *recommendation utility*, facilitating users' access to items, information and services that may be useful for them and would not have otherwise found (Adomavicius & Kwon, 2012). As important concepts, the recommendations' *diversity* and *novelty* have been identified, in order to support three essential ground concepts related to user selection and satisfaction, namely *choice*, *discovery* and *relevance*.

More specific, generating recommendations that a user has already used, liked and obviously will keep on using may be of high accuracy. However, it does not provide any added value, as the user would also find and use these items without having been recommended them. An ideal recommendation should mimic the behaviour of a trusted friend with more knowledge on a given topic, providing recommendations balancing between the desired level of accuracy while enhancing diversity, novelty and serendipity. Thus, the challenge arising is to provide recommendations for highly relevant items that users would not discover otherwise, and give them the possibility to enjoy variety and surprising alternatives, instead of getting bored by receiving trivial recommendations. To this direction the dimensions of novelty and diversity become crucial as they may provide a wider perspective to the recommendation problem (Y. C. Zhang et al., 2012).

Diversity

Diversity refers to the variety present in a given recommendation list (Y. C. Zhang et al., 2012). Adomavicious and Kwon (Adomavicius & Kwon, 2012) define different perspectives of diversity, as individual and aggregate diversity.

- *Individual diversity*, also referred to as *intra list diversity (ILD)* or as intra list distance, measures the diversity of the recommendations for an individual user. It can be modeled as the average aggregated pairwise dissimilarity of the items in a given recommendation set. After having defined a proper distance metric $d(i, j)$ for the items in the recommendation list Lst, intra list diversity according to (Castells et al., 2015), can be calculated as:

$$ILD = \frac{1}{|Lst|(|Lst|-1)} \sum_{i \in Lst} \sum_{j \neq i \in Lst} d(i,j) \tag{3.1}$$

- *Aggregate diversity* aims to measure the total recommendations' impact. It approaches the concept of diversity across the total of recommendation sets generated by the system for different users. For example, recommending to every user the same set of n very different among them, items could lead to high individual diversity but low aggregate one (Adomavicius & Kwon, 2012).

Recommendations' diversity is considered of high importance, and especially in CB methods it could increase the recommendations' quality (Yu et al., 2009). However, many times a high value of diversity may have a negative effect on recommendations' accuracy, thus it is still among the current research issues to identify methods that would recommend a set of items being above some predefined accuracy threshold while having a certain degree of diversity (Castells et al., 2011).

Adomavicius and Kwon have shown that the use of re-ranking approaches, and especially by parametrizing the common ranking functions with a raking threshold, is able to lead to a balance between diversity and accuracy (Adomavicius & Kwon, 2012). They propose the use a graph theoretic approach where the user-item relations form the nodes of a directed graph based on which, following a maximum flow approach, the set of N highly diverse items are identified, and then used as recommendation candidates, ranked with a standard ranking approach. Shi also bases the analysis on a graph-based approach, with the users and items being the nodes of a directed graph with 1^{st} order Markovian, whose transition costs are designed to combine data related to long tail, accuracy and diversity at the same time. The recommendations are then generated based on a minimal cost flow concept (Shi, 2013).

Novelty

In recent years, novelty has also been considered as an important parameter, providing added value to a recommender system and being able to increase the recommendations' quality (Celma & Lamere, 2011). As *novelty*, we generally refer to difference between present and past experiences. From a global perspective, the novelty of an item can be defined as the opposite of its popularity, considering as novel an item that is located in the long tail of the popularity distribution, thus few people are aware that it exists (Castells et al., 2011).

Using an approach similar to the inverse document frequency, referred to as *inverse user frequency*, if U_i is the set of users that have interacted with item i, we can model the user-item interactions as $IUF = -log_2|U_i|/|U|$. Based on this definition the novelty of a recommendation set can be defined as the average novelty of the items in the list Lst, thus can be calculated as (Castells et al., 2015):

$$AvIUF = -\frac{1}{|Lst|} \sum_{i \in Lst} log_2|U_i|/|U| \tag{3.2}$$

However, also more personalized perspectives of novelty can be evaluated by calculating whether a specific item or specific items attributes have been experienced by a user in the past (Celma & Herrera, 2008). From these definitions also arises the term of user unexpectedness, that reflects how surprising an item may be for a user, given his/her past observations (Castells et al., 2015). The authors in (Vargas & Castells, 2011) differentiate between popularity and distance based novelty, where the first is related to what has been already observed, and has a negative correlation with the popularity of an item, while the latter measures the distance of an item and a context of experience, and is modelled as an attribute of an item that can be related with a set of other items from an Euclidean point of view. When referring to a recommender system, novelty can be seen as the ability of the system to present users with items that they have not experienced before, and in certain percentage it may have a positive effect on the users' trust of the system.

Novelty and diversity are related, but however they measure different dimensions of the recommendation problem. Novelty refers to the difference with respect to what a user has already seen, while diversity refers to the difference of items with respect to each other within a list. Therefore, novelty supports the notion of discovery by presenting users with items they would not otherwise have access to, while diversity intends to extend the range of presented alternatives to the user. However, although having different definitions and scopes, Zhang and Hurley provide two definitions of diversity, as average dissimilarity and average rarity. Related to the second definition, they propose that support to novelty leads to an additional amount of diversity in a set, as highly novel items are usually at least slightly different with the already selected according to some perspective (M. Zhang & Hurley, 2008).

Serendipity

Serendipity represents the unusualness, or surprise, of the recommendations leading to a positive emotional response. Serendipity is usually measured in terms of semantic distance between the recommended items, and the expected ones, in order to find the degree of deviation from a user's traditional behaviour (Adamopoulos & Tuzhilin, 2014). Some authors consider serendipity as the conjunction of novelty and relevance, that is also closely related to the long tail items distribution in the majority of recommendation problems (Castells et al., 2015). Serendipity can be assessed using some distance function on the item properties, or through its opposite, un-serendipity, that measures the average similarity between new recommendations and items in user's history (Y. C. Zhang et al., 2012), or simply the items provided by a primitive method (Ge et al., 2010). Serendipity is also closely related to the term of *unexpectedness* (Murakami et al., 2008). However, in order to correctly capture these terms, the real actual expectations of a user have to be efficiently identified, as well as the utility of a recommendation, that is closely related to its quality.

Supposing that EX is the set of expected items within a recommendation list Lst, the degree of unexpected items in general can be defined as (Castells et al., 2015; Adamopoulos & Tuzhilin, 2014),

$$Unexp = |Lst - EX|/|Lst| \tag{3.3}$$

However, only the set of useful items among the unexpected ones leads to a positive response from the user. Thus, the degree of serendipity can be measured as (Adamopoulos & Tuzhilin, 2014),

$$Serendipity = |Unexp \cap Useful/|Lst| \tag{3.4}$$

Coherence

Finally, *coherence* has been used to evaluate the transition effects among songs, especially important in automatically generated playlists. More precisely, coherence evaluates the pairwise suitability of two consequent music items in a list (Castells et al., 2011). Although the order of songs may be not as important for non-expert users, when addressing the needs of more demanding users, with higher music knowledge, the transitions between songs may highly affect the playlists' quality, as non-coherent changes may lead to a disruptive result (Shao et al., 2009). Therefore, the coherence of a playlist, as resulting from the songs co-occurring effects and their ordering is evaluated, as it has been reported among the important parameters defining playlists' quality (Jannach et al., 2015; Kamehkhosh & Jannach, 2017).

Given a local similarity metric among music items $sim(i, j)$ the coherence of a list Lst can be calculated as the average similarity among the pairs of consequent items in the list, calculated as:

$$Coherence = \frac{1}{|Lst| - 1} \sum_{i \in Lst} sim(i, i+1) \tag{3.5}$$

In contrast to novelty, that has been found to positively relate with diversity, the opposite occurs with coherence. Therefore, it is still among the main issues in APG, to generate playlists able to balance among coherence and diversity, as both have been reported as being important for the users.

3.5.3 Individual Perception

Still there is a lot left to further investigate in relation to individuals' perception of music and the way this is reflected in their music preferences and selections. For sake of completeness, even though these factors are not directly addressed in the thesis at hand, we also present shortly some basic emotional and cultural dimensions related to music perception and consumption. These factors in general are hard to be captured and analysed, therefore the majority of MRSs still does not evaluate them directly. However, their incorporation into a MRS would significantly improve the quality of the recommendation results (Ferwerda & Schedl, 2014).

Emotional

Music itself has been characterized as *the expression of emotions*. However, this can be highly subjective and difficult to model, or capture automatically, as this would require also a solid and global model of emotion. In addition there is an important distinction between the emotion(s) expressed and the emotion(s) induced by music. Both of these dimensions are also affected by the general social context and the personal motivation on a given moment (Kim et al., 2010).

Although there is an influence of ones culture on his/her judgement of musical emotions, Balkwill and Thompson state there may also be universal influences of musically expressed emotions. They define as "psychophysical dimension" any property that can be perceived independently of musical expertise, knowledge or enculturation, like speed of pulse, tempo, timbre and loudness, and show that emotional judgements and psychophysical dimensions have strong relations (Balkwill & Thompson, 1999). Furthermore, music can be considered both as a social phenomenon, as it is at the centrer of many social activities and a way to communicate specific believes or express an identity, while it is able to address needs of individuals beyond the social context. For instance, individuals may seek for particular music styles in order to regulate their emotional states (Rentfrow & Gosling, 2003).

MIR systems mainly use categorical or dimensional psychometrics, to classify emotional responses to music. The first ones find and organize related tags based on their relevance to some music pieces/styles, while the latter suggest that mood can be scaled on various dimensions. Thus it can be represented by its positioning on a space of independent axes. For example, in the two dimensional Valence-Arousal space, emotions are positioned based on their intensity and polarity. Additional sources on the music items could be used, like lyrics, web pages, social tags, etc., to derive emotional information indirectly from related sources, through the use of NLP like with other types of metadata related to music pieces. However, as emotions can be influenced by attributes like the tempo, the timbre, the harmony and the loudness of a song, emotional classification can be also done based on acoustic features like dynamics, timbre, harmony, register, rhythm and articulation (Kim et al., 2010).

Personality

Music has been also found as the second most used strategy to change, create, maintain, or enhance emotions. However, as emotion is less stable than personality, the way that users select emotionally laden music pieces under different emotional situations also depends on their personality (Ferwerda et al., 2017). Personality in general influences individuals' behaviour and preferences.

One of the widely used models to categorize personality is the five factors model (FFM) that consists of five general dimensions to characterise and classify personality. Each of those (being openness, conscientiousness, extraversion, agreeableness, neuroticism) is further associated with a cluster of primary related factors. Personality traits have been found to be related to music preferences, therefore knowing a user's personality would enable the generation of improved music recommendations (Ferwerda & Schedl, 2014; Rentfrow & Gosling, 2003).

Users may show clear preferences for specific types of music over others, suggesting that there are some links to users' personalities that define these preferences. The studies performed by Rentflow and Gosling have led to the categorisation of 14 music genres (like blues, jazz, rock etc.) according to 4 factors, namely *reflective and complex*, *intense and rebellious*, *upbeat and conventional*, *energetic and rhythmic*. In order to then map user preferences over those factors with users' personality characteristics, they have analysed and computed the correlations of music preference dimensions to these factors. For instance, based on the resulting correlations individuals who prefer intense and rebellious music (being rock, alternative and heavy metal) tend to be curious about different things, open to taking risks and physically active and consider them selves as intelligent. On the other hand, those who prefer reflective and complex music (like blues, jazz, classical and folk) tend to be inventive, with active imagination that reject conservative ideals (Rentfrow & Gosling, 2003).

On the other hand, Ferwerda et al. have conducted real experiments to measure user taxonomic choices, in terms of mood, activity and genre. related to their personalities. Based on the observed interactions they suggest that users' personality traits could be used to define their preference towards taxonomies in streaming services. For instance, they have found positive correlations between openness to experience and likeliness to choose for mood, high conscientiousness and preference for activity and finally users scoring high on neuroticism being more likely to choose for activity or genre (Ferwerda, Yang, et al., 2015). Thus these preferences could also be used to personalise music recommendations.

Cultural

In general the majority of the research held in MRSs has been done based on the so-called "western music". People in general, like with language, tend to recognise better the musical patterns that they are more familiar with (Morrison & Demorest, 2009).

However, although listening habits and music consumption may differ also subject to demographic factors, few analysis have been done towards this direction. To this direction Schedl, using the last.fm dataset presented in (Schedl, 2016), performs an analysis of music preferences per country. He constructs "country-specific" genre profiles and aims to identify listening patterns and similarities between the different countries (Schedl, 2017). The observed genre popularity ranking is quite consistent, with rock and electronic genres being the most popular in the majority of the 47 presented countries. In some cases, like in China and Japan pop music is preferred to alternative, while in Hungary and Romania the same applies to electronic music. In addition, the similarities of country genre preferences are presented showing that listeners in some groups of countries share certain listening characteristics, like for instance in USA, UK, Canada, and Australia, while in others differ significantly for the rest, and from the considered as "mainstream attitude".

Finally, Ferwerda et al. based on the same dataset, suggest that the desired levels of recommendations' diversity, related to different music variables, also vary through the analysed 47 countries. They analysed the relationship between listening preference characteristics, namely desired diversity of genres and artists, with six basic cultural characteristics along the different countries. Various correlations were identified, highlighting that there is a difference in the listening patterns across countries. This fact probably should be taken into account both when generating recommendations and when evaluating their perceived quality, especially when no other user-related or contextual data are available (Ferwerda et al., 2016).

3.6 Conclusions

In this chapter, the theoretical background and the basic functionalities of MRSs and MIR systems have been presented. Furthermore, as the focus of this thesis is on APC, an analysis of the main APG categories, the special characteristics of music items, and the additional dimensions of playlists that require special attention during their recommendation process, have been described.

As the focus is set on recommendation of playlists being relevant, but also addressing parameters *beyond accuracy*, the importance and influence of those parameters on playlist recommendations has been explained. In addition, music selections are heavily influenced by users' listening behaviour, the consumption context and emotional parameters that may be related to both the consumption context and the user's character. The general contextual dimensions affecting music consumption and some interesting approaches of capturing those, within music recommendations, have been also presented.

In table 3.1, an overview of the basic characteristics of the main approaches that have been applied to APG recommendations, with their main limitations and advantages, can be found.

Method	**Core**	**Limitations**	**Advantages**
Constraint optimization	cost function	strict constrains	address explicit requests
Markov models	transition probabilities	only previous state	coherent transitions
Sequential patterns	patterns/rules	computat. cost eval. difficulty	co-occuring patterns
NLP	probability distributions	subject to noise	reveal hidden relations
CBR	similarity function	data quality dependence	capture tendencies

Table 3.1: APG techniques' basic characteristics

Part III

Proposed Recommender System

Chapter 4

Methodology

The main goal of this thesis is the study, design and implementation of a hybrid recommendation system able to recommend playlist continuations that seems more appropriate to complete users' experience. The generated recommendations are based on the concept of the started playlist and the co-occurring patterns of the music styles included. These patterns are derived from previous playlists instead of comparing standalone song properties or trying to predict user ratings. More precisely, this approach aims to identify the specifications of items that seem able to satisfy a user based on past joint selections, and to further enable the presentation of diverse items that the user maybe was not aware of before, but after trying would be keen on using again. The design of this RS, its basic functionalities and their evolution are presented in this chapter.

4.1 Problem Description

The starting point of this thesis was the idea that human item selections in many cases are performed as sets or within sessions, thus are not affected only by the characteristics of each item or users' past preferences. Furthermore, they are based on some rationale that defines *joint item selections* in order to deliver a more complete user experience.

In most recommendation cases, when trying to determine user preferences, the focus is placed on user similarities, pairwise interactions or information on individual items, like item attributes or ratings assigned to them by users. However, there are many situations where a user needs to obtain a set (or a sequence) of items in order to cover a specific need. In such cases, simply ranking the items and presenting the user the top-N of them would not provide the optimal solution, as this set may contain many closely similar alternatives. In addition, items in a given set may interact in various ways that increase or decrease the resulting utility of the set (complementary or incompatible items, respectively). This delivered set must have some characteristics as a whole that will ensure the user's satisfaction, like *variety* and *coherence* of the included items, while being relevant with a specific request, at the same time. Moreover, especially in services that are being frequently used, the *novelty* of the proposed items also affects the user perception of the system, of course, when the recommended items are relevant and of high quality according to the active user.

The above applies also to playlist generations. Although users may like various songs, possibly of different types, those in many cases would do not provide a smooth transition from one to another. For example, somebody may like from classical to hard rock music, but normally would not put songs of these categories into the same playlist, but rather organise songs of similar styles into coherent playlists that could listen to in different occasions. This example highlights the existence of an "underlying music concept" behind the selection of songs for a playlist. Due to users' cognitive perception of music and the semantic dimensions of songs, there should be some level of *coherence* in a playlist, excluding songs that do not fit within a concept, which may be related to their lyrics, tempo, general style or purpose. Furthermore, coherence heavily depends on the kind of user, expert or novice (Lee, 2014), as expert users tend to place higher importance on the playlists' continuity compared to inexperienced users that may listen to music in shuffle mode (Liu & Hsieh, 2009). The purpose of the playlist's construction also affects the desired coherence level. For instance, in playlists used as "background music" less importance is placed on coherence, compared to playlists that are made to serve a specific purpose, like dj sets or mixes made for a party.

In addition, generating playlists being of some *diversity* may increase the user excitement. In contast, presenting only closely similar music items, like only songs from the same album or artist, might result in the user getting bored or even annoyed after having been presented a few times the same set of songs (Ziegler et al., 2005; M. Zhang & Hurley, 2008). Finally, permitting the recommendation of novel items, enabling the discovery of items that users would not have found otherwise could increase their interest and engagement to the system (Lee & Lee, 2011). Like in all recommender systems, results' accuracy and response time are among the critical parameters, for a system to gain users trust. Users in general expect to receive recommendations within a reasonable amount of time, and especially during the first uses will accept "bad" recommendations only a few times before giving up the use of the specific system, (desJardins et al., 2006).

The existence of *underlying patterns* that define the joint item selections has been initially observed in areas like market research and customer behaviour analysis. However, as described in previous chapters, the majority of the currently used recommendation techniques, when trying to predict the items that a user is going to select, focuses on isolated items or user characteristics, and does not evaluate the circumstances at the decision moment and/or the other items selected together. More than using the usual CB and CF methods that focus on single item recommendations, recommendations of sets of items have been mainly analysed by applying association or sequential rules to mine these item correlations. Recently, the application of latent factor analysis and probabilistic models has been applied to this area in order to overcome the common drawbacks of the rule-based techniques.

Playlist, and music consumption in general is considered to be highly affected by the context, and the emotional state of the user, under which they were generated and consumed (Ferwerda, Schedl, & Tkalcic, 2015; Pichl et al., 2015). However, there may be various dimensions of context, perceived differently by users depending on their personality and general preferences, thus are differently reflected in their music selections. Therefore, as the selection patterns are mainly influenced and driven by user behavioural factors they cannot be easily specified and predicted. However, more light can be set on users' habits and behaviour through the evaluation of the patterns behind the sets of co-occurring music items.

Apart from the difficulties and special issues that collection RSs have to address, in domains related to music and multimedia they also face the problem of the *semantic gap* (Celma et al., 2006), as shown in figure 4.1.

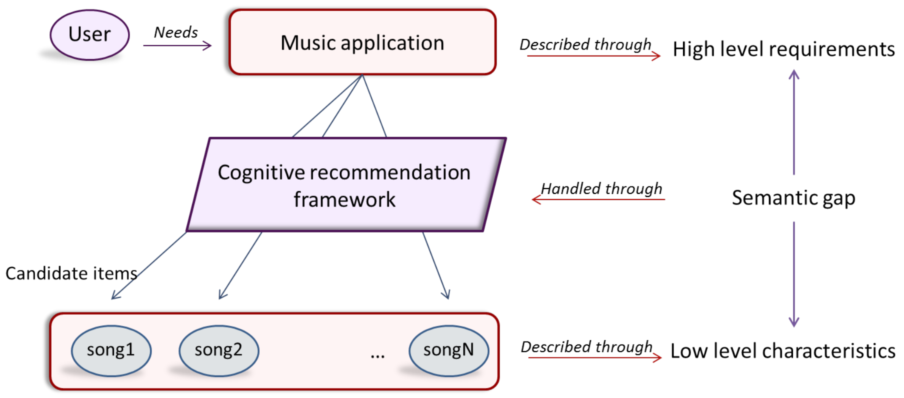


Figure 4.1: The semantic gap in music (recommendation and information retrieval) applications

The used recommendation techniques usually model music items through low-level sound-based characteristics (like beats per minute, tempo etc.), while users describe their needs through high level requirements (expressed through terms like "relaxing music", "smooth transitions" etc.) (Kim et al., 2010). In order to overcome this issues MRSs tend to relate requests to metadata associated with the music items or focus only on their ratings and usually fail to capture the structure behind users' behaviour. In order to generate meaningful recommendations, the systems first have to overcome this gap, and model user preferences and joint selections in a way that it would permit their comparison and exploration. Such a modelling would enable the observation of user selections, and provide more details on the rationale, and possibly on emotional and other factors related to music perception, leading to these selections.

In addition, there is the need of recommender systems focusing on parameters *beyond accuracy* in order to deliver a more complete user experience (Konstan & Riedl, 2012). These systems should have the ability to present users with recommendations being relevant to a given concept, that has to be deduced from the underlying data patterns. In figure 4.2, a general schema of such a *cognitive architecture* for playlist recommendations is presented.

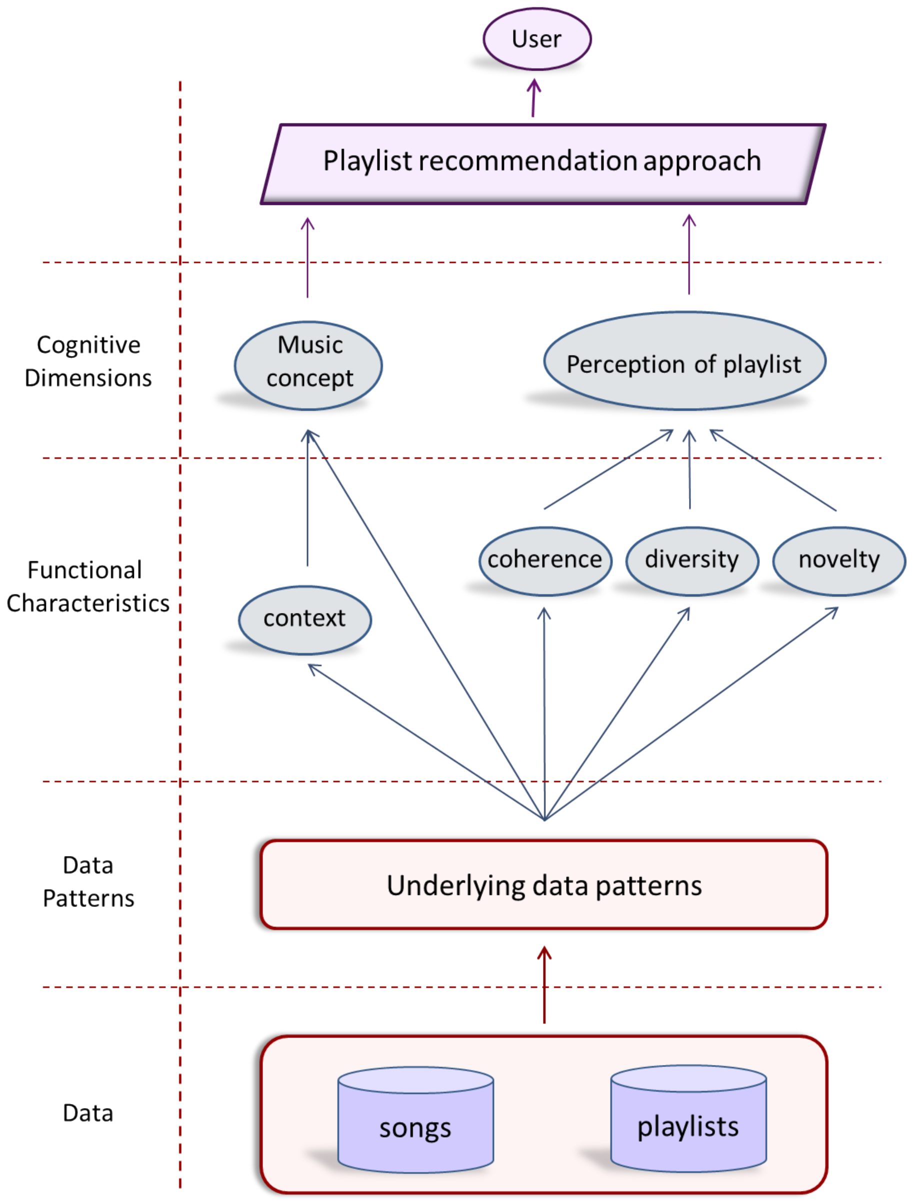


Figure 4.2: General schema behind a cognitive playlist recommendation approach

4.2 Motivation and Goal

The motivation of this research work arises from the lack of efficient recommender systems to handle the above issues, and efficiently recommend sets of items that would fit within a started concept, especially in domains associated with complex content or sentiment data, like in playlist continuation recommendations.

The designed approach aims to provide more insight into users listening habits and preferences, and based on them, to generate recommendations of music items able to complete the active user's experience. Therefore, an effective mapping between user requirements, as those have been expressed through their past selections, and sets of items with specific attributes has to be performed. In order to, not only predict whether a music item, a song or an artist, will be or not liked by a user, but also alleviate the popularity bias and long tail problems that CF recommender systems face, the specifications of the items appearing within different concepts have to be detected.

As the proposed methodology aims to address similar semantic concepts, rather than similar users, it does not require the construction of a complicated user profile, therefore it could perform better in new user, cold-start situations. More specific, the recommendations are based on the styles of the items appearing in similar concepts. In order to identify those, as well as the similarity of *item concepts*, a graph-based knowledge model is proposed. Furthermore, the evaluation of users' previous selections is not based on a an exact item match but on a *similarity metric*. Therefore, less popular items with similar specifications, that appeared in similar situations in the past, and could fit into the current, could be recommended, supporting the discovery of new items that could fit into users' expectations.

Market basket analysis (MBA) is one of the basic areas where the sets of items already selected together influence the items that will be further selected. Therefore, the starting point of this work was to model this problem and then to extend it to more complicated domains. More precisely, the initial design was based on CBR together with a hierarchical item model used to identify the properties of the items appearing in specific concepts. Further, as the intention was to extend this approach to domains with more cognitive dimensions, where the treated items may be associated also with rich content or possibly semantic data, the focus was shifted to music recommendations and especially to the automatic playlist continuation (See Chapter 3 for more details on this area). In order to handle the music items selected together in this use case, in a more flexible and less domain knowledge dependent way, the hierarchical item model was changed with a graph-based model. This model is used for the representation of songs and music styles, as well as for the computation of their similarities.

4.3 Important Issues

Among the main difficulties when treating recommendations of items collections is the fact that there are no negative ratings or opinions. When a user selects and purchases some items we infer his/her likeliness towards those items using their selection frequency. However, not selecting other items cannot be inferred as a non-preference or a low rating, as this may also result from the fact that the user does not know those item or that they do not fit within the current concept. There still has not been done enough research towards this direction of recommender systems.

Another important dimension of this problem that has to be specified is the difference between *sets of items (or collections)* and *sequences of items.* A *collection, or set, of items* is the set of items that are selected together while the *sequence* of items refers to a sorted set of items. Therefore in the case of evaluating sets of items, only the joint selection of items is evaluated, while when treating sequences of items the selection order is also important. In the case of item sequences, there is also a temporal factor that should be taken into account in order to specify when each item was selected. This could be also treated as a two-step process, first defining the set of the selected items and then their exact sequence, using for example time-stamped data.

Finally, the solution design may heavily depend on whether the items consumed together are of the same or different categories. For instance, when referring to playlist recommendations all of the items are music items, while when referring to leisure activities or market basket recommendations, the recommended sets in general consist of items of different types.

4.4 Problem Formalization and Data Modelling

In contrast to most RSs, that work mainly based on user-item interactions, captured through ratings like $U \times I \rightarrow R$, in HybA interactions are modelled through sessions (or transactions).

A *session* is defined as the set of items that have been selected together or sequentially in close time moments, by the same user. Each playlist is a session, containing a set of songs, selected by a user at some time moment under given circumstances (Vall, Eghbal-zadeh, et al., 2017). For example, a low tempo electronic playlist could be enjoyable a late winter night while a soul funk mid-tempo playlist would most probably be heard during a Sunday sunny afternoon.

Thus, first the problem data have to be transformed from selection events to sessions using the additional information available, like the events' timestamps, the playlist id or title, as the same user may construct different playlists in different time moments. The user and the explicit time moment, where available, form additional characteristics that depending on the application domain may, or not, have an effect on the playlist characteristics. The emphasis is set on the *co-occurring patterns* of music styles in playlists more than on the user or the explicit contextual information that could be associated with them.
The problem domain entities can be described as:

- a given dataset D containing a set of z songs $I = \{i_1, \ldots, i_z\}$
- a set of tags T where each song $i_j \in I$ can be represented as a distribution over tags as $i_j = \{t_{j1}, t_{j2}, \ldots, t_{jm}\}$, $t_{jk} \in T$, $k = 1, \ldots, m$
- a set of previously reproduced playlists $L = \{l_1, \ldots, l_k\}$ that each can be written as the set of songs that it consists of, being $l_j = \{i_{j1}, \ldots, i_{jn}\}$, $i_{jt} \in I$, $t = 1, \ldots, n$
- a set of users $U = \{u_1, \ldots, u_v\}$ that have formed those playlists

These entities may be associated with additional characteristics, like sound-related metadata, editorial information (genre, tempo, artist, lyrics, etc.), temporal or contextual data (event's timestamp, source of reproduction etc.) and demographic information, (age, gender, origin, etc.), respectively.

Emphasizing on the song descriptions, based on which their similarity can be calculated, the used tags could be attributes that characterise their music style, like editorial metadata, user reviews or sound features possibly coming from some proper sound or signal analysis process, like in the majority of pure MRSs. In general user generated tags, especially from non-expert users, tend to add sparsity and noise to the recommendation problem as they are highly subjective. Therefore, it is preferable to describe songs based on more accurate information (Bogdanov & Herrera, 2011).

In order to identify the possible interactions between the diverse entities, we further model the entities of the playlist generation and recommendation problem as:
An undirected graph $G = (V, E)$ where:

- V is the set of nodes (vertices) being the diverse entities, namely users, playlists and songs, thus $V = \{U, L, I\}$
- E is the set of edges connecting nodes with some kind of relationship

Depending on the kind of nodes that the edges $e \in E$ connect, as shown in figure 4.3, we may have the following, as well as their inverse, relationships:

- Users "make" playlists $u \in U$, $l \in L$, $e(u, l) \in E$
- Playlists "contain" songs $l \in L$, $i \in I$, $e(l, i) \in E$
- Songs "have characteristics" $i \in I$, $t_i \in T$, $e(i, t_i) \in E$

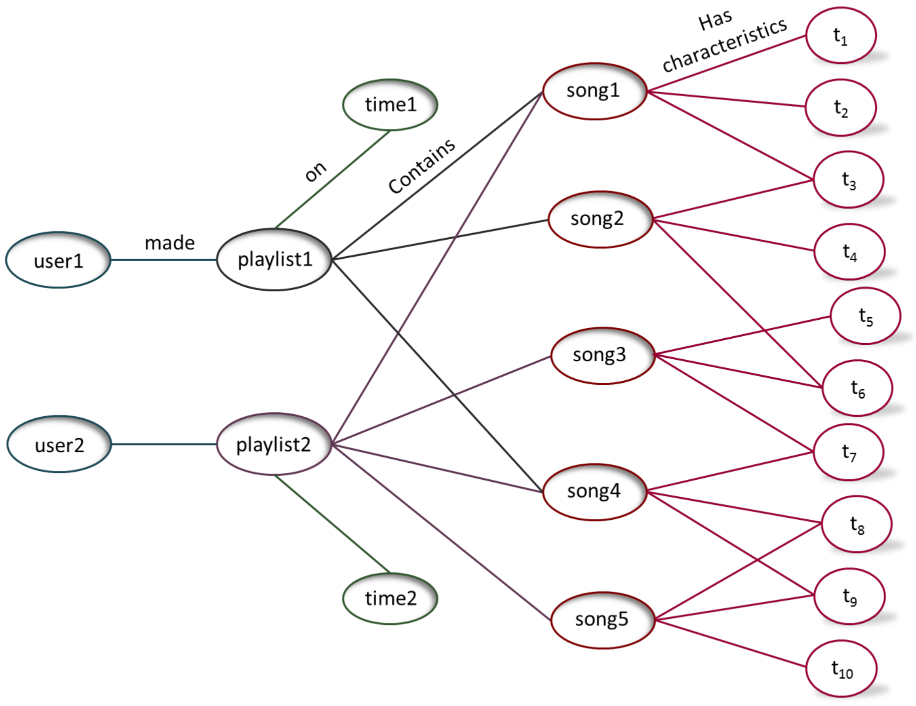


Figure 4.3: Playlist and song distributions

However, this graph model is used only for the representation of the problem entities, to identify their connections and enable the similarity computations, and not directly for generating recommendations, like in (Guerraoui et al., 2017; Interdonato et al., 2013).

4.5 Hybrid Recommender System Architecture

HybA follows the basic idea of CBR, that *"Similar problems have similar solutions"* and relies on the general CBR process (CBR Cycle) and its basic concepts of *similarity* and *retrieval*. CBR as a problem solving methodology is closely related to the basic cognitive decision making process followed by humans when facing new problems, generally referred to as new cases. In these situations, in order to find an appropriate solution people first try to identify the most similar experienced cases, in order to re-use them in an appropriate way to solve the new problem (Richter & Weber, 2013; Lopez De Mantaras et al., 2005).

Similarly, the designed algorithm, given a new playlist $l_N = \{i_{N1}, \ldots, i_{Nk}\}$ identifies the current style and looks for the past most similar ones. The characteristics of those are then used to construct, and recommend, the set of n songs $l_{rec} = \{i_{N(k+1)}, \ldots, i_{N(k+n)}\}$ that seem as the most appropriate for completing the current list. As users' taste and needs may vary along time, being in general related to the context within which they are generated, the same user may look for very different items in different moments. For example, in some cases, users seem to not care about maintaining some specific style and enjoy more diverse songs, thus the recommended set may also be more diverse. On the other hand, for users with more specific preferences more importance should be placed on the coherence of the recommended set. Therefore, this system does not base its reasoning on user related data, but it aims to identify and address similar playlist concepts.

The reasoning process of HybA, follows the basic "steps" shown in figure 4.4. It takes as input a new playlist, analyses its characteristics, and ends with the generation and recommendation of an appropriate continuation. When the new playlist is entered, the set of already selected items is compared to those in the existing cases, and the k most similar case(s) are retrieved in order to identify how cases with similar descriptions were structured in the past. The similarity of the playlists is calculated based on their general conceptual characteristics as well as on the music styles of the songs in them. After these candidate lists have been found, the music items found in them are used in order to construct the playlist continuation. An important factor is the value of k, being the number of past cases that will be retrieved, as well as the additional, beyond accuracy parameters, that are taken into account during the candidates' ranking phase and their relative importance. Depending on those factors, the items that will be used to generate the playlist continuation will be finally extracted.

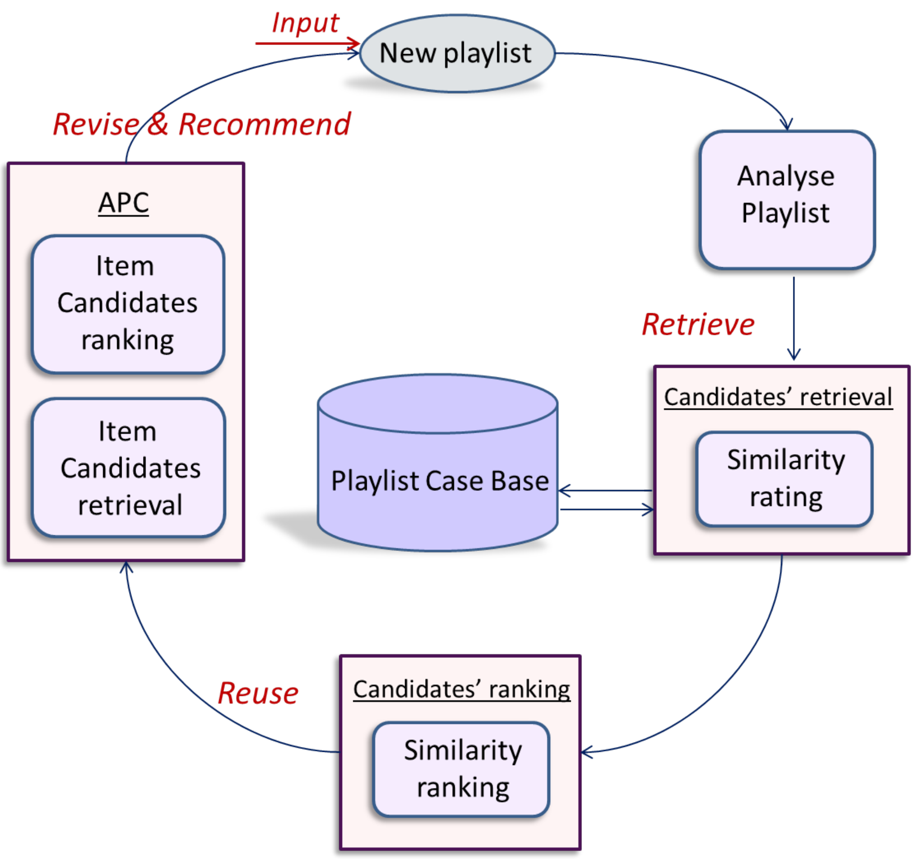


Figure 4.4: General CBR APC process

4.6 Design and Implementation Evolution

Previous to the design of the actual, and final recommender system, HybA, various approaches were proposed and tested to gradually identify vulnerabilities and limitations that had to be overcome, as well as to enrich each time the system with additional functionalities.

The starting point of HybA could be summarized as the idea of implementing an approach similar to MBA (Gatzioura & Sànchez-Marrè, 2015), where items frequently co-occurring in transactions justify the existence of some underlying patterns related to users' perception of items' joint utility. Starting from a similar model, adapted to the needs of playlist recommendations domain, till the final conceptualized hybrid recommender, the basic characteristics and purposes of the different designed recommendation approaches could be summarized as the following:

1. *MusCBR*: Initial version of a Music Case-based Recommender with focus on the playlists similarity aiming to accurately recommend sets of music items (music styles or artists) forming appropriate playlist continuations.

2. *CMusCBR*: The Contextual Music Case-based Recommender. Improves

the MusCBR algorithm by including a contextual pre-filtering based on the playlist context, the generation time moment (day time). Focusing on playlists' similarity with aim to recommend relevant sets of song clusters, of the same music style or artist.

3. *MusHR*: The Music Hybrid Recommender, extending the reasoning model of MusCBR by using a two-level reasoning model, with focus mainly on similarity aiming to recommend sets of songs forming relevant and interesting playlist continuations. Furthermore, beginning from this recommender, the emphasis shifts towards attributes beyond accuracy, like *diversity* and *long-tail* presence. Therefore, additional variations of MusHR were tested, in order to evaluate these dimensions in the recommended playlist. A maximum acceptable level of diversity deviation from the started list, and a minimum long tail percent in the recommended playlist continuations were set as target attributes.

4. *HybA*: The final hybrid recommendation system, combining the two level reasoning model initially designed for MusHR, while incorporating a pre-filtering similar to CMusCBR. Furthermore, the use of explicit context is extended to a more general *conceptual pre-filtering*, based on implicit information retrieved from songs' music styles. HybA aims to recommend playlist continuations in terms of *lists of songs*, being relevant, coherent and of some diversity level. In addition, various versions of HybA were tested before selecting the most adequate one, each time setting more emphasis, apart from the similarity between playlist patterns, on their *coherence* and/or *diversity* level. However, in this version, more than specifying the desired tendencies by threshold values or intending to maximise a specific parameter, those are extracted from the started playlist. Furthermore, the aim is to follow the tendencies observed in the new playlist.

4.7 Recommendation Approach Overview

In order to support the needs of the specific problem, the functionalities of the RS are performed in two phases and using two abstraction levels. More specific, during the *offline* (pre-processing) phase, the parameters requiring costly calculations and not changing their values during the recommendation process, like song similarities and past playlist characteristics, are calculated. Then, during the *online* (recommendations' generation) phase, based on those values and on the characteristics of the seed playlist, each time the playlist candidates are identified and recommendations are generated based on them.

Furthermore, two similarity levels are used in order to capture the similarities of items, and the similarities of playlists formed based on these items, referred to as *local* and *global similarity* respectively. The local similarities of the existing music items are calculated during the offline phase, while the global ones are approximated each time a new playlist is started.

4.7.1 MusCBR and CMusCBR

The first two approaches of the proposed RS, namely *MusCBR* and *CMusCBR*, were designed with the aim to capture the specifications of music items selected together. Their reasoning processes follow the general CBR Cycle while their emphasis was mainly placed on the recommendation of *song clusters* of some common characteristics, rather than songs. Furthermore, they have been found able to identify and recommend accurately artists or music styles that would better complete started playlists (Gatzioura & Sànchez-Marrè, 2017a, 2017b). The overviews of their core algorithms can be found in figures 4.5 and 4.6, respectively.

```
Input: Past playlists $L$, new playlist $l_N$, recommendations number $n$
Output: PlaylistContinuation $l_{rec}$

Data Pre-processing
1:  Analyse $L$: parameters, similarities
Candidates' Retrieval
2:  For ($l_R \in L$ with $|l_R| \geq n$)
3:      Calculate playlist similarities $Sim(l_N, l_R)$
4:  Rank candidate playlists in $L$
5:  If ($|L| > k$)
6:      Get $L' \leftarrow top\_k(L)$ playlists
7:  Get $\forall i' \in I', l' \in L', e(i', l') \in E$, candidate songs
Recommendations' Generation
8:  Rate candidate songs in $I'$
9:  Rank candidate songs in $I'$
10: $l_{rec} \leftarrow top\_n(I')$ songs
11: Return $l_{rec}$
```

Figure 4.5: MusCBR recommendation algorithm

CMusCBR extends MusCBR by performing, during the offline phase, a contextual clustering (using the playlist creation hour) of the playlists in the case base. After that, for each new playlist CMusCBR retrieves the most similar past playlists only from the corresponding contextual cluster, rather than searching in the whole case base.

```
Input: Past playlists $L$, new playlist $l_N$, recommendations number $n$
Output: PlaylistContinuation $l_{rec}$

Data Pre-processing
1:  Analyse $L$ : parameters, similarities, contextual clusters
2:  Retrieve $L_C \subseteq L$: playlists of the same context with $l_N$
Candidates' Retrieval
3:  For ($l_R \in L_C$ with $|l_R| \geq n$)
4:      Calculate playlist similarities $Sim(l_N, l_R)$
5:  Rank candidate playlists in $L_C$
6:  If ($|L_C| > k$)
7:      Get $L' \leftarrow top\_k(L_C)$ playlists
8:  Get $I' : \forall i' \in I', l' \in L', e(i', l') \in E$, candidate songs
Recommendations' Generation
9:  Rate candidate songs in $I'$
10: Rank candidate songs in $I'$
11: $l_{rec} \leftarrow top\_n(I')$ songs
12: Return $l_{rec}$
```

Figure 4.6: CMusCBR recommendation algorithm

4.7.2 MusHR

As two-level similarity metrics have been found to perform better than single-level metrics (Campos et al., 2017) the initial similarity and CBR core model, of MusCBR and CMusCBR, was extended into a hybrid two-level model, in order to better capture the cognitive characteristics of the lists. In more detail, in *MusHR* playlist similarities are calculated based on their general characteristics and on the music styles of the songs in them, and then, further calculations take place incorporating additional song attributes and user experience related parameters. This two-level process, except from capturing more attributes related to a playlist's style, also permits the control of additional quality related dimensions, like coherence and diversity.

The aim of the initial two level model, MusHR, extends the previous two, and the generated recommendations are playlist continuations of specific songs. More precisely, MusHR uses song clusters, of songs of the same artist or style, mainly to better identify the most similar past cases. The emphasis during the retrieval process was set only on the playlists similarity while the support to additional parameters, like diversity and long tail, was performed during the reuse phase, as it is shown in figure 4.7.

```
Input: Past playlists $L$, new playlist $l_N$, recommendations number $n$
Output: PlaylistContinuation $l_{rec}$

Data Pre-processing
1: Analyse $L$ : parameters, similarities, popularities
2: Analyse $l_N$: $l_N = \{s_{N1}, \dots, s_{Nm}\} \in L_S$
Candidates' Retrieval
3: For ($l_R \in L_s$ with $|l_R| \geq n$)
4:     Calculate playlist similarities $Sim(l_N, l_R)$
5: Rank candidate playlists in $L_S$
6: If ($|L_S| > k$)
7:     Get $L'_S \leftarrow top\_k(L_S)$ playlists
8: Get $I'$: $\forall i' \in I', l' \in L'_S, e(i', l') \in E$, candidate songs
Recommendations' Generation
9: Rate candidate songs in $I'$
10: Rank candidate songs in $I'$
11: $l_{rec} \leftarrow top\_n(I')$ songs
12: While ($QualityParameters < threshold$)
13:     Re-rank candidate songs in $I'$
14: $l_{rec} \leftarrow top\_n(I')$ songs
15: Return $l_{rec}$
```

Figure 4.7: MusHR recommendation algorithm

In MusHR the evaluation and control of the additional, beyond accuracy, parameters was done by the use of threshold values, forming soft constraints. More specific, if during the candidates' retrieval phase, no candidate items able to compose a playlist with the desired characteristics were found, then the recommended playlist continuation would be the one generated using accuracy related parameters. In contrast, in the final approach, HybA, the desired quality characteristics are incorporated into the retrieval phase and treated at a higher level, before extracting the candidate items, as it is shown in figure 4.8.

4.7.3 HybA

With aim to improve the recommendation accuracy, and also reduce the computational time during the playlist candidates' retrieval, especially when used on large datasets, a prior clustering of the case base had to be performed.

To this direction, a contextual pre-filtering based on the playlist generation moment (time of the day) had been used in CMusCBR, to test this assumption and was found to provide improved results when recommending artists or music styles (Gatzioura & Sànchez-Marrè, 2017b). In HybA, first this contextual model, taking into account the playlist generation time (the hour of the day and the month), was evaluated. However, as many times explicit contextual information, related to the playlist generation moment, is not provided, in order to further improve the performance of MusHR a more general clustering and pre-filtering of the case base was designed and incorporated in HybA. Furthermore, this model aims to capture more general, implicit, playlist characteristics, extracted from their latent topic distributions, referred to as *"playlist concept"* (Gatzioura, Jorge, et al., 2018).

```
Input: Past playlists $L$, new playlist $l_N$, recommendations number $n$
Output: PlaylistContinuation $l_{rec}$

Data Pre-processing
1:  Analyse $L$ : parameters, similarities, concepts
2:  Analyse $l_N$: $l_N = \{s_{N1}, \dots, s_{Nm}\} \in L_S$ and concept
Candidates' Retrieval
3:  Retrieve $L_C \subseteq L$: playlists of the same concept with $l_N$
4:  For ($l_R \in L_C$ with $|l_R| \geq n$)
5:      Calculate playlist similarities $Sim(l_N, l_R)$
6:  Rank candidate playlists in $L_C$
7:  If ($|L_C| > k$)
8:      Get $L'_S \leftarrow top\_k(L_C)$ playlists
Recommendations' Generation
9:  Get $\forall i' \in I', l' \in L'_S, e(i', l') \in E$, candidate songs
10: Rate candidate songs in $I'$
11: Rank candidate songs in $I'$
12: $l_{rec} \leftarrow top\_n(I')$ songs
13: Return $l_{rec}$
```

Figure 4.8: HybA recommendation algorithm

Similarly, (Pichl et al., 2015) propose a contextual clustering and filtering using Natural Language Processing (NLP) and the implicit contextual information found in playlist titles. This approach could be useful when titles contain words like "Christmas", "party" or "summer" but subjective titles like "my favourite music" or "top songs" may lead to clusters not valuable for improving the recommendation process. However, many times even the same given context may be differently perceived by the users, thus not being equally reflected on their music preferences.

An overview of the HybA approach, showing its main functionalities can be found in figure 4.8 and will be described in more detail in the following sections. Line 1 refers to the *offline data pre-processing* phase during which song and music style similarities are calculated and the characteristics of past playlists are analysed in order to find their diversity and coherence degrees. In addition, the underlying latent dimensions are used in order to identify the general playlist concepts and perform an appropriate clustering of the case base. On the other hand, line 2 refers to the analysis of the started playlist, the identification of its concept that is followed by a *"conceptual pre-filtering"* of the case base (line 3) leading to the extraction only of the playlists addressing the same concept that will be then used for further calculations.

In lines 4-8, from the cluster of playlists of the same concept, the *playlist candidates' are retrieved* based on their similarity degrees with the new playlist, while in lines 9-13 the *recommendations' generation* takes place. In line 9 the item candidates are extracted from the candidate playlists and in line 10 are rated based on the total aggregated rating of the playlist(s) in which they appeared. In this step, depending on the recommendation scope and the RS version used, the rating function evaluates the global similarity level and possibly the coherence or/and the diversity degree of the candidate playlist. Finally, in line 11 the candidates are sorted and the continuation of the playlist is constructed (line 12) and recommended (line 13).

4.8 HybA Offline Phase: Data Pre-processing

4.8.1 Song Models

The focus of the developed system is placed on the similarity among semantic concepts rather than similarities among users. This approach aims to capture the tendencies and patterns present in those concepts, as they may reflect the users' perception of music and playlist quality. Therefore, HybA aims to identify first the music styles of the songs placed together in pleasant playlists, rather than finding the exact songs. Due to the high number of existing songs that could possibly be used to serve the same, or a very similar, purpose, when focusing on the exact songs placed in playlists, meaningful information on user preferences cannot be directly extracted. In general, music style selections depend on the user's taste and the playlist creation scope, while the final song selections may be heavily influenced by songs' *popularity*.

Therefore, before analysing the playlist-song distributions it is important to identify the playlist-music styles distributions. More specific, an *appropriate clustering* of the songs, based on their styles, as defined by their characteristics, should be first performed. These characteristics could be either sound-related attributes or metadata. Thus, the starting point was to identify a proper model for the songs, enabling the representation of their characteristics, their comparison and calculation of similarity, and a proper clustering based on it. The songs in each of the resulting clusters should be closely similar, sharing the same music style, that is not restricted only to their artist and/or music genre. Each song cluster is represented by one music style, being the result of the combination of the features that its songs have in common. When personal preferences related to artists are not taken into account, these songs could be used to address the same cases and could be "interchanged" in playlists without degrading the listening result, in terms of quality.

Having items described as distributions over a set of attributes, being web-mined or user generated tags, real features or latent descriptions, would permit modelling them as vectors, graphs or through hierarchical models. Among these schemas, finally the *graph model* has been selected as the most appropriate for our goal, as it provides more flexibility than a hierarchical model while leading to same or improved results, and it is computationally more efficient than using a vector representation in domains of sparse descriptions.

Hierarchical Item Model

Initially, a hierarchical knowledge model was tried in order to categorise the items into clusters of almost identical ones, without having to analyse the values of all their low-level attributes, or using complicated domain specific methods.

This hierarchical model follows a tree structure, as in figure 4.9, where starting from the general item concept, at each level the set of existing items are divided into clusters of the more similar ones. Based on the various categories and subcategories that each item belongs to, the resulting divisions follow the paths from the tree root towards the item's position.

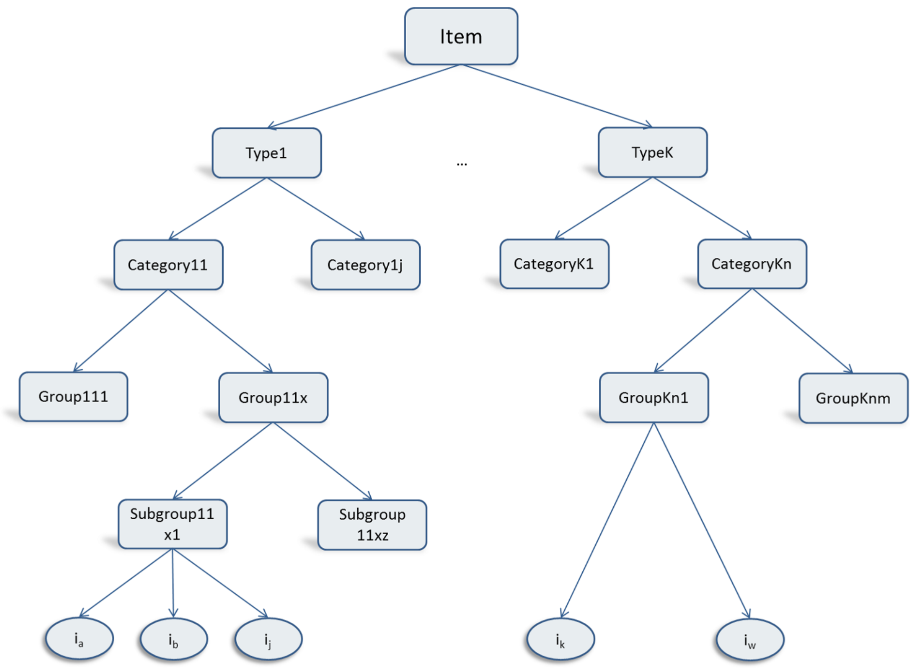


Figure 4.9: Items' general hierarchical model

At each level, the items that have common ancestors and are not differentiated at the current level are clustered together, and are treated as identical. This classification of items, depending on the application domain, may come from a proper ontology or may be extracted from a dataset structure. Given the level of abstraction that we wish in our recommendations, the level at which the items will be modelled is set. For example, as same items at the third level are regarded the items belonging to the same categories at the first two levels that have the same characteristics at the third level.

Two item representations, i_N and i_R, are thought to be *similar* to the extent to which they share the same path from the tree root to their position. Therefore, for items modelled as shown in figure 4.9 and, having specified the level of abstraction required, their *local similarity* level is calculated based on the length of their common paths, thus is given from the following equation:

$$sim(i_N, i_R) = \frac{lengthOfCommonPath(i_N, i_R)}{level(i_N)} \tag{4.1}$$

For example, in the market basket use case, a possible classification of the products sold in a store is shown in figure 4.10. For instance, in drinks we would have milk, semi-skimmed, glass bottle of 1 litre of some brand. As our intention is to capture the specifications of the items that appear within similar sets, we possibly would not care about the brand name or the distributional package but on the selection of items of a concrete type (like the selection of milk or semi-skimmed milk). Our focus, depending on the abstraction level used and the scope of the system, would be on the recommendation of groups of items described at the second and third hierarchical level (which correspond to the levels of Group and Subgroup respectively).

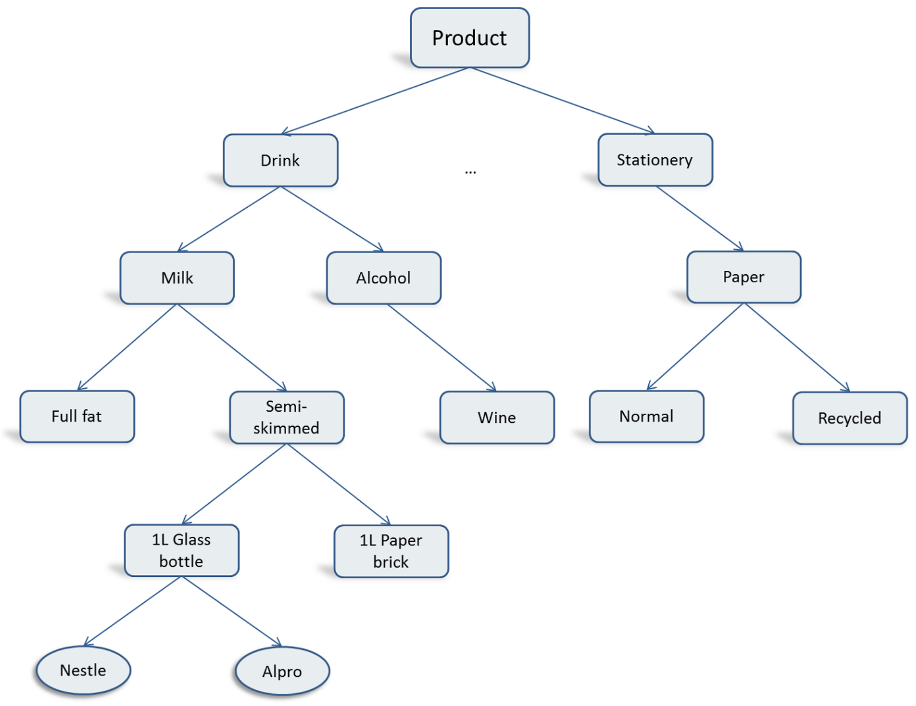


Figure 4.10: Example of product hierarchical classification

Another example of this hierarchical item classification can be found in figures 4.11 and 4.12 for music items (songs), where apart from the general type classification of songs into rock, hip hop, funk, jazz, etc., and their tempo characteristics, there can be various subtypes, like for example in rock there is indie, progressive, alternative, pop rock, hard rock, etc. However, when referring to songs, as the importance of certain characteristics may be highly subjective or may depend on the scope of a specific application, an a-priori hierarchical categorisation cannot be performed. For instance, one user may categorise songs first based on their genre, and last by their artist, like in figure 4.11, while another user would emphasize mostly on their lyrics' language and least on their tempo, like in figure 4.12.

The problem that arises from the use of an hierarchical schema in more complicated domains, like in music recommendations, is that in general, the relative importance of some attributes cannot be decided a priori without having the necessary domain knowledge. The appropriate subtypes, their definitions, as well as the importance of different attributes, may depend on subjective factors related to each user's perception of music that are not always clear. Finally, many times the differences between song styles definitions may be very small, or there may be artists that have songs of different styles, belonging to various subtypes.

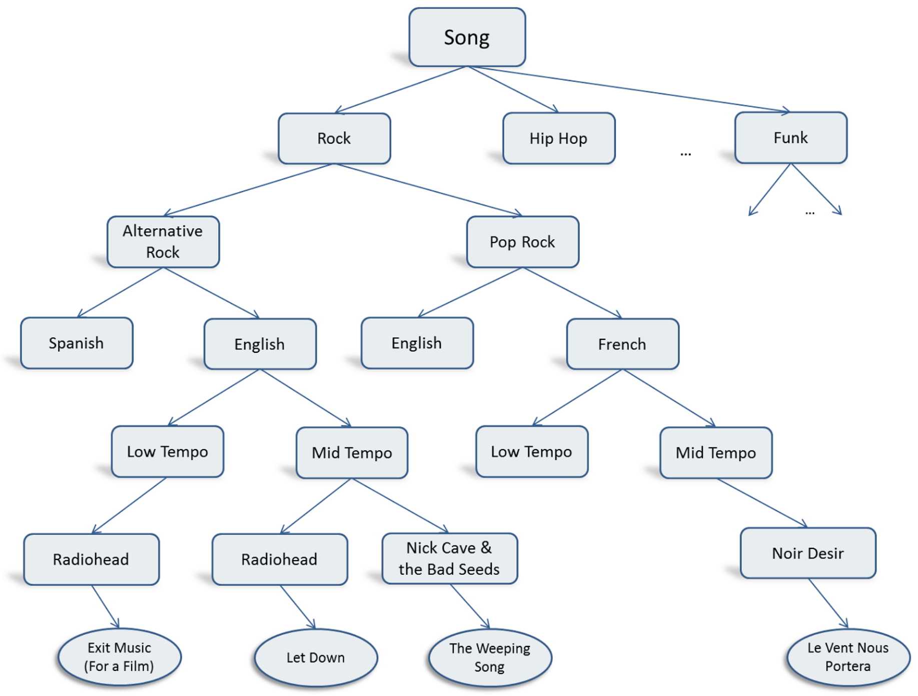


Figure 4.11: Example of first possible songs hierarchical classification

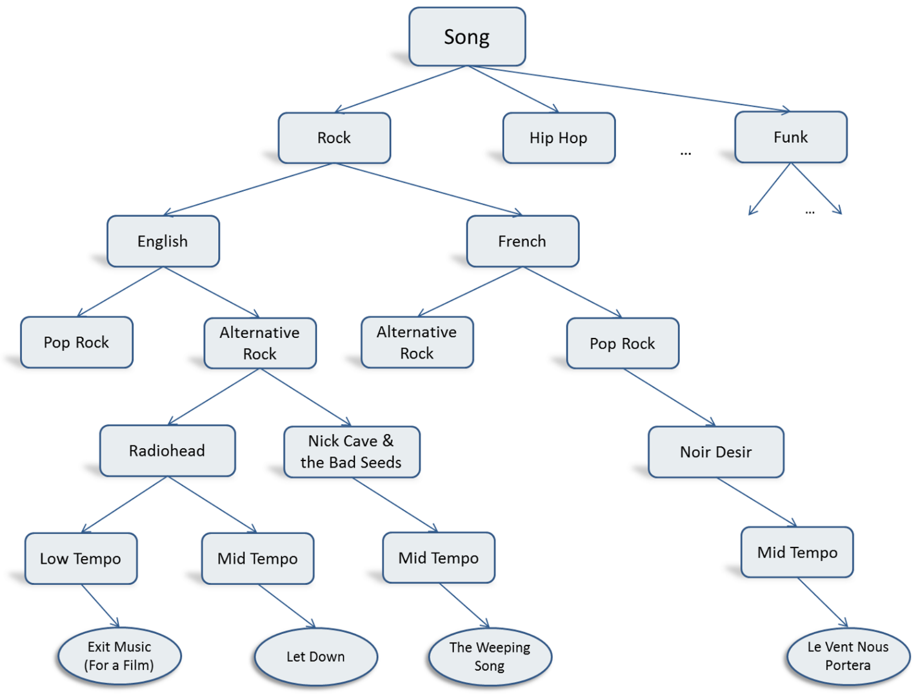


Figure 4.12: Example of second possible songs hierarchical classification

A hierarchical model requires domain knowledge specifying the relative importance of the item attributes in order to correctly categorise them. Therefore, such a model could be convenient when referring to sets of items of different types, where the parameters that mostly differentiate the items are placed higher in the hierarchy, like in market basket analysis, where it has been found to perform well (Gatzioura & Sànchez-Marrè, 2015). However, when treating items of rich content, like songs, a more flexible model has to be used.

Graph Item Model

As an alternative, a graph-based model, extending the problem entities' graph was designed. The song nodes of the graph are connected with the tags that compose their specifications, and their similarity is computed as a function of the density of common connections. In contrast to the hierarchical model, in this model the specifications of the songs are considered as of the same importance. Furthermore, when combined, the songs that have almost identical descriptions, lead to *clusters of songs* that could eventually be used in the same cases. In general we refer to the different feature combinations, each associated with a song cluster, as *music styles.*

In order to initiate the system, as part of the offline process, we parse the problem graph to identify the existing relationships and calculate the similarity levels between songs and song clusters. We model music items using metadata or latent features related to their style, that in the case of songs described through editorial information, would be category, subcategory, tempo, language, vocal, artist etc. The artist performing a song can be regarded simply as one more tag present or absent in the item description, although sometimes it may heavily influence song styles and user selections. Therefore, when available, this information could be given an increased weight.

Let T be the set of existing tags, music metadata used to describe the songs. Each song can be written as $i_j = \{t_{j1}, \ldots, t_{jm}\}$, $t_{jk} \in T$, $k = 1, \ldots, m$. Given the desired detail level, and the data sparsity, we may use all or some of the songs' tags, having thus more specific or abstract item descriptions.

The similarity of two items depends on the density of their common features and the number of unique characteristics of each. More precisely:

- Given two songs described as $i_a = \{t_{a1}, \ldots, t_{al}\}$ and $i_b = \{t_{b1}, \ldots, t_{bk}\}$
- Given that $n(a \cap b)$ is the number of tags that i_a and i_b have in common, $n(a \setminus b)$ is the number of tags associated with i_a and not i_b, while $n(b \setminus a)$ is the number of tags that only i_b, and not i_a, has
- Their similarity level, refered to as *local similarity* is calculated using the following formula (Gatzioura & Sànchez-Marrè, 2017a; Sánchez et al., 2012):

$$sim(i_a, i_b) = 1 - log_2\Big(1 + \frac{n(a \setminus b) + n(b \setminus a)}{n(a \cap b) + n(a \setminus b) + n(b \setminus a)}\Big) \tag{4.2}$$

An example can be found below, using the songs *Exit Music (For a Film)* and *Let Down* by *Radiohead, The Weeping Song* by *Nick Cave and the Bad Seeds* and *Le Vent Nous Portera* by *Noir Desir.* When using the music style characteristics assigned to them, and the artist that they are performed by, we can describe those as follows:

Let Down={Let Down, Rock, Alternative Rock, English, Male,Mid Tempo, Radiohead}
Exit Music (For a Film)={Exit Music (For a Film), Rock, Alternative Rock, English, Male, Low Tempo, Radiohead}
The Weeping Song={The Weeping Song, Rock, Alternative Rock, English, Male, Mid Tempo, Nick Cave and the Bad Seeds}
Le Vent Nous Portera={Le Vent Nous Portera, Rock, Pop Rock, French, Male, Mid Tempo, Noir Desir}

We can see that *Let Down* and *Exit Music (For a Film)*, both performed by Radiohead have almost the same characteristics but one is characterised as mid and the other as low tempo, while *Let Down* and *The Weeping Song* seem to have the same style characteristics but are performed by different artists. Therefore, depending on the decision whether the artist will form an additional feature of the song' style or not, the previously mentioned songs could be categorised in the same or in different clusters, according to their "style".

In figure 4.13, we present an example of the songs' graph for these songs, where their level of similarity can be approximated through the density of the tags they have in common.

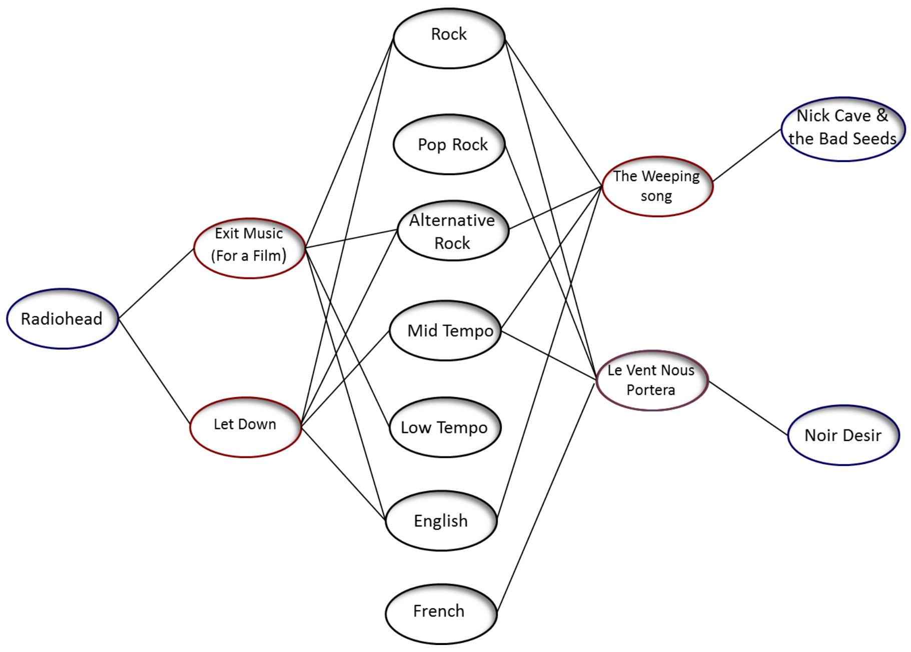


Figure 4.13: Example of songs graph based representation

Item Models' Comparison

As described previously, for the calculation of the local item similarities, and the items' clustering based on their styles, a proper item model is needed. Among the tested ones, an hierarchical model would require a prior definition of the importance of the items' attributes. The attributes that mostly differentiate the items, and define their categories should be placed at a higher hierarchical level, gradually dividing items into groups of the most similar ones. This model requires some domain related knowledge or an additional ontology or taxonomic schema that would enable its automatic extraction. Additionally, in many application domains, the notion of more "similar items" may be exposed to subjective and application related parameters. For example, in multimedia recommendations, when treating songs or movies, the novelty of an item may be of high importance for a user, clearly focusing on novel items, therefore differentiating new from old items. On the other hand, the genre may be more important according to another user who focuses basically on the item's style.

The proposed graph-based item model seems able to capture the degree of items' similarity without the need of an additional knowledge base while being more flexible. It uses the density of shared descriptions, without differentiating the importance on the various attributes. This similarity calculation enables, at the same moment, the approximation of the coherence and diversity degrees between couples of items and among entire item sets, that may be of high importance for the user experience. Additionally, it seems to be more interoperable, as any item that can be described as a distribution over the existing tags of the items in the system can be modelled according to this schema, and its similarity with the previous items can be calculated. In addition, this model is easier to integrate with the rest of the problem data. Finally, calculating item similarity based on feature vectors would be usually computationally more expensive, when sparse tags form the item descriptions, like when user generated tags are used.

An important dimension affecting the item model selection, refers to whether the treated items are of the same or different item categories. For example, when we focus on playlist recommendations, all the items in the recommended set are music items, songs, genres or artists, while when we perform market basket recommendations, we will recommend a set of items that may be of different categories, maybe alimentary and stationery items sold in the same store. Therefore, when recommending sets of items of different categories, hierarchical models may be used to model the items as their principal category is the one that mostly differentiates them. In contrast, when treating items of the same type, the construction of such a model may be highly subjective, heavily affecting the recommendation results, while requiring domain knowledge. Indeed, tests performed on an initial small scale music database showed a great variance in accuracy, depending on the selected attributes hierarchy. Thus, more flexible models are thought to be more appropriate in such domains.

However, each time that new items are inserted to the system database they would have to be tagged carefully, usually by an expert, in order to avoid adding noise to the system. The main drawback of this metric, compared with the initially proposed hierarchical model, is that it is computationally more expensive, as all the attributes of the items are evaluated, instead of evaluating only the number of common ancestors, being the number of subcategories that two items have in common. On the other hand, this is the reason why this method is able to provide recommendations of higher accuracy even when using more abstract item descriptions.

4.8.2 Playlist General Concept

A playlist is a collection of music items, reproduced as a meaningful sequence that should have some special characteristics as a whole (Bonnin & Jannach, 2014; Cunningham et al., 2006; Casey et al., 2008).

As our focus is placed on the entire playlists' characteristics, more than on the exact songs appearing in them, we want to exploit the additional semantic information related to the entire playlists. From the state of the Art review presented in chapter 3, the following categorisation of playlists' characteristics is proposed, being:

- *Internal or content related*: As "content" of a playlist we refer to the music items that it consists of. The characteristics of these items, which may be songs, artists or song clusters, define on a grand extend the style of the entire playlist.

- *External or environmental related*: As external, we refer to the all the parameters related with the playlist, that may define its content, but are not directly described in it. For example, the user that makes/reproduces the list, its purpose, the context within which this is done, that could be the time, the company, the location, the weather, etc. and the influence these factors have on the user's emotional state and music perception.

Although not always explicitly reflected in terms of formulation, the context and the other external parameters related to a playlist have been found as among the important parameters that influence its general style and the songs placed in it. However, as music perception is highly subjective, and may be heavily affected by circumstances under which is consumed, by the personality and the emotional state of the user, it is hard to establish a direct and stable connection between users, and the playlists they would enjoy in a given context.

The term *playlist concept* is introduced as a wider term to capture the general characteristics of a playlist, being the result of the combination of both *internal* and *external parameters*, beyond the explicilty defined context. In figure 4.14, that is a generalization of figure 4.3 previously presented, the important parameters related to playlists, on which the proposed approach bases its reasoning, are shown. More specific, playlists are formulated to serve a specific semantic concept, and are treated as distributions over music styles. Finally, music styles form the result of different combinations of songs' characteristics.

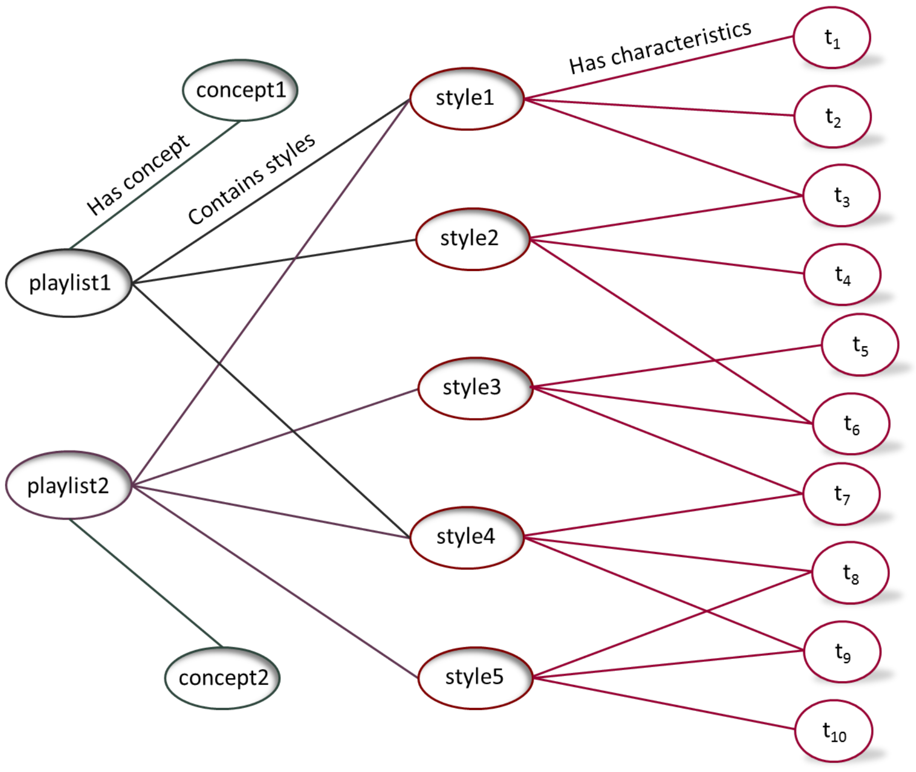


Figure 4.14: Playlist distribution over music styles

As this kind of information is not always explicitly stated, it has to be deduced from the data patterns in the consumed playlists. Therefore, in order to capture the general concept of the playlists, without focusing on their specific content or when no additional information, like their names or explicitly defined context, the playlists' latent topic distribution, based on the music styles included in them, is used.

A probabilistic topic model using *Latent Dirichlet Allocation (LDA)* (Blei et al., 2003) is built to extract the underlying latent topics from the playlists in the problem case base, and the music styles in them, like in figure 4.15.

Having every playlist described as a distribution over music styles permits us to find its probability distribution over the existing latent topics. Then, the *dominant topic(s)* of each playlist are used as a representation of its general concept. We consider the general concept of a playlist as part of its cognitive definition, related to its purpose and mood, reflected on its music perception, and usually captured through quality-related parameters (like its coherence, diversity etc.), as shown previously in figure 4.2.

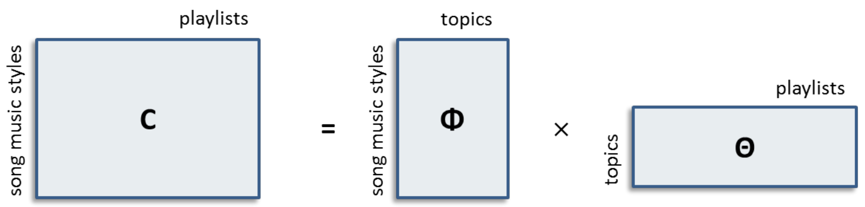


Figure 4.15: Playlists' latent distributions over music styles

In general, Probabilistic Topic Models, follow the assumption that documents are formed as probability distributions over topics that consist of words from an existing dictionary. Thus, depending on the basic idea to be expressed in a document, words from different topics are selected with different probabilities and the overall documents' probability distributions will vary (Steyvers & Griffiths, 2007; Blei, 2012). These models have been successfully used in NLP and Information Retrieval (IR) applications.

4.9 HybA Online phase: Recommendation Algorithm

The HybA RS uses a hybrid two-level model, that models entire playlists as cases, being distributions over music items, which may be songs, artists or music styles. It is based on a meta-level hybridization where first LDA is applied to the set of past playlists, described as distributions over music styles, to define their concepts. After that, CBR is applied on a refined set of cases $L_C \subseteq L$, being the result of a conceptual pre-filtering of the problem's case base. In figure 4.16, the followed CBR cycle for APC recommendation process, previously shown in figure 4.4, is presented in more details, highlighting the important functionalities and processes that take place, from the moment the new playlist is entered, till its continuation is generated and recommended.

Playlists can be treated as distributions over song styles through which they are connected to songs, or as distributions over songs, and songs as distributions over features. More specific, the algorithm uses sub-graphs of the whole problem graph (similar to figures 4.3 and 4.14) to identify the characteristics of the song clusters that the playlists are formed of, and based on them, calculates the playlists' degree of similarity. When a new playlist is started, first the most similar past ones are found, and based on them, the continuation of the new playlist is generated and recommended.

Following the general case description in CBR systems, as an ordered pair $c = (p, q)$ where p is the problem description and q the problem solution, with $p \cap q = \emptyset$, we model each music playlist as a case $l_j \equiv c = \{i_{j1}, \ldots, i_{jm}\}$. Furthermore, for the recommendation problem we use as problem description the "new" (N) playlist $l_N \equiv p = \{i_{N1}, \ldots, i_{Nk}\}$ being the set of the initially selected songs, while the solution is the recommended (rec) playlist continuation of length n, $l_{rec} \equiv q = \{i_{N(k+1)}, \ldots, i_{N(k+n)}\}$.

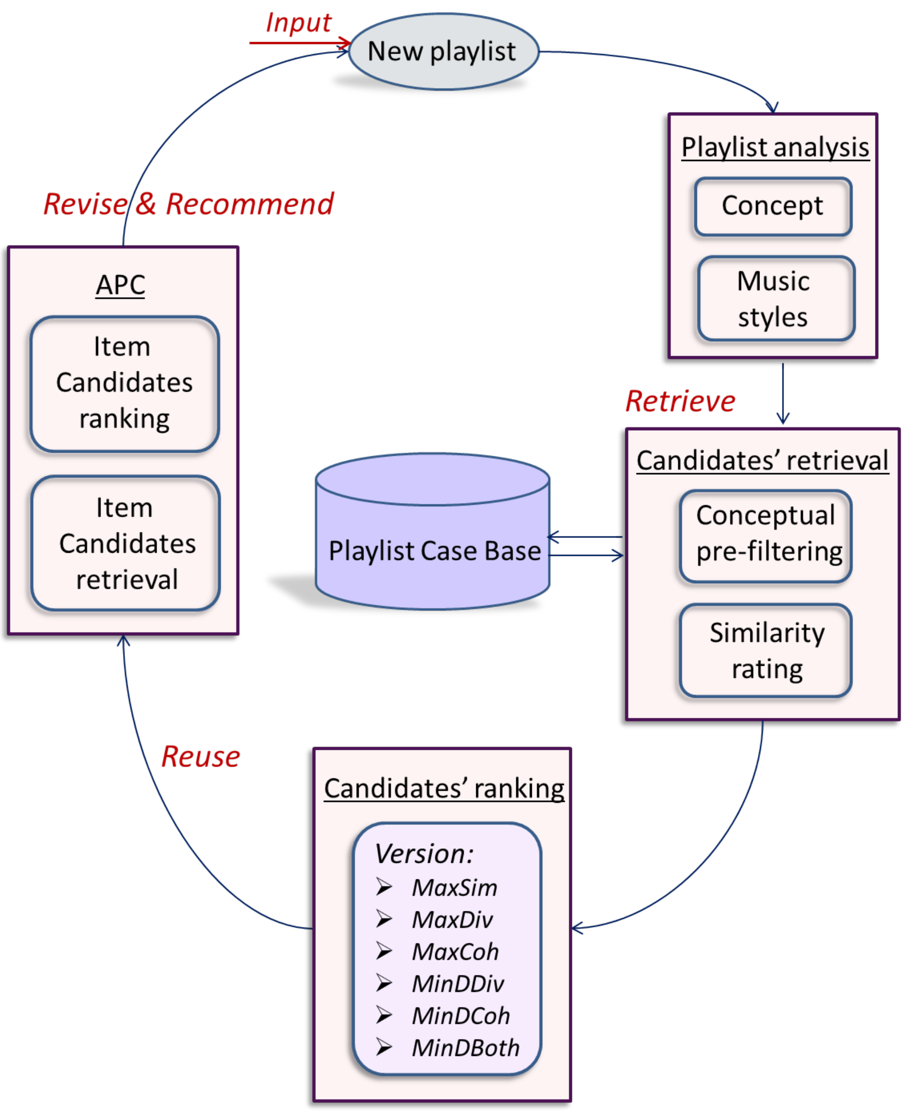


Figure 4.16: Detailed CBR APC process

4.9.1 Candidates' Retrieval

An important process of a CBR system that highly affects its performance, is its ability to identify the characteristics of a new case, and identify within an acceptable response time, the past cases that seem as most appropriate for this problem.

Therefore, it is of high importance to capture the special characteristics of a new playlist in order to efficiently retrieve the globally acceptable candidates. An appropriate clustering of the past playlists should be performed based on parameters leading to playlist clusters of some special characteristics. Then, given a new playlist l_N the algorithm retrieves only the instances of the most relevant cluster $L_C \subseteq L$, forming the input for further analysis.

From those playlists the k most similar to l_N are identified. Those are the playlists with *global similarity* that maximises the following equation,

$$l' \in L_C : \forall l_R \in L_C,\ l' = argmax\{Sim(l_N, l_R)\} \tag{4.3}$$

Candidates Pre-filtering

In order to find globally acceptable candidates for more complicated queries, in terms of user experience, various parameters related to the playlists, contextual and user related, were evaluated. The scope is to perform a pre-filtering of the case base, based on parameters capable of identifying the group of past playlists being most relevant with the new one, according to some criteria. These playlists will be used for further computations, finally leading to the identification of the k most appropriate candidates, like in figure 4.17.

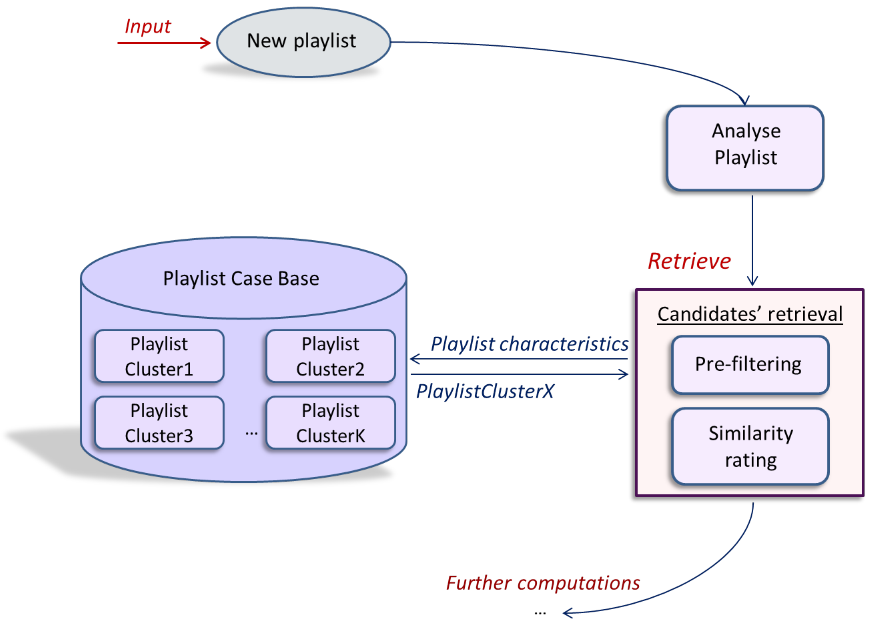


Figure 4.17: Candidates' pre-filtering

As the aim of the system is to address similar playlist concepts, not restricted to the users that constructed them, first the additional attributes associated with the playlists were evaluated. These attributes, that are not directly associated with the items included in the lists, could be the user who constructed them, and the time moment at which this happened, that were initially analysed. Users' similarity could be calculated based on both their demographic attributes, where available, and their past preferences. However, in many cases, demographic information may not be available or may contain a lot of missing values. After analysing both, on an initial testing dataset, past preferences have been found as more important towards the identification of similarities among users.

The use of contextual parameters, in general has been found to improve recommendation accuracy. In addition it has been proposed (Domingues et al., 2011) that context-aware recommenders are able to perform better for long tail and novel items. As the music datasets that were used did not provide information on the actual user status (in terms of location, activity and emotional state) while "consuming" the playlists, among the contextual parameters that can be evaluated is the time moment (date and time) at which this happened.

The initial evaluation of the above assumptions, revealed that especially the *hour of the day*, seems to influence the music playlists generated in the given time and to be related to their style. Thus, an initial contextual clustering based on the hour of the day was performed in order to identify the "neighbourhood" of each hour, in terms of the hours at which similar lists are constructed. Initially, this contextual pre-filtering was performed based on the explicit contextual information related to the playlist generation moment (construction time). This process has been found to increase the recommendation accuracy when recommending song styles or artists, and at the same moment decrease the computational effort needed, as similarity calculations take place only for the playlists in the most similar contextual cluster (Gatzioura & Sànchez-Marrè, 2017b).

Furthermore, we tested the influence of the following contextual dimensions related to the playlist consumption moment:

- the *month*, based on the notion that the general mood is closely associated with the weather that changes during the year
- the *hour* of the day, that affects both the user activities, and his/her mood, therefore are closely related with the music selections performed
- both the *hour and month* when a playlist was constructed, or consumed, are evaluated

Three contextual version of HybA were designed to evaluate the above tendencies on large scale datasets. From those, given a new playlist:

- *cntxT*, uses the cluster L_H of playlists generated in close hours
- *cntxM*, uses the cluster L_M of playlists reproduced in close months
- *cntxC*, uses the cluster L_{Comb} that combines the hour of the day and the month when playlists were made

As it will be presented in the next chapter, those methods have similar performances and the combination of month and day hour, generally leads to improved results compared to the use of only one of them (Gatzioura, Sànchez-Marrè, & Jorge, 2018).

Nevertheless, many times explicit contextual information is not available, or may have different influences on user preferences, depending on their character and their emotional state. For example, it has been found that users depending on their character and emotional state perform different music selections under similar circumstances (Ferwerda et al., 2017). People with more extrovert personalities tend to rely on "happy" music under sad or stressful situations, in order to "cheer" themselves up, while introverts tend to rely on more sad or depressive music similar to their emotional state.

Thus, explicit contextual information, even if available, could lead to the extraction of very general playlist clusters (for example, sad music on rainy days) that would not manage to capture the specific user preferences and actual music tendencies. In order to overcome this common limitation, during the retrieval phase, the contextual pre-filtering was extended to a *conceptual pre-filtering*, where the playlists' conceptual similarities are evaluated. More specific, based on the music styles' distribution in the started playlist, and the on the topics extracted from the LDA model on the case base, the topic distribution of the new list, specifying its concept, is identified. Based on this attribute, as a first part of the retrieval process, the instances of the corresponding conceptual cluster L_C are retrieved. These playlists, that address the same concept with the new started playlist, form the initial candidates' set that will be used for the rest of the computations. This conceptual pre-filtering has been found to provide important improvements of the recommendation results, in terms of both accuracy and playlists' quality, compared to the use of explicit context (Gatzioura, Sànchez-Marrè, & Jorge, 2018).

Candidates' Global Similarity

After having filtered the playlists based on their concepts, in order to find the lists that further address (4.3), HybA performs a two level similarity calculation. First the similarity degree of each item in the new list is specified, based on the similarity values found with all the items in a retrieved list, as shown in figure 4.18, and then using those values the global similarity of the two lists is calculated aggregating those values.

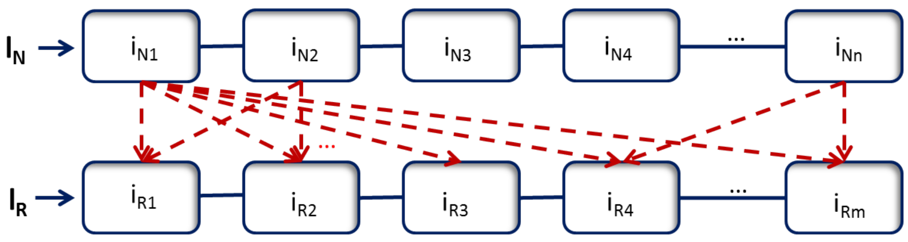


Figure 4.18: Comparing a new and a retrieved list

Let two compared playlists be, a new $l_N = \{i_{N1}, \ldots, i_{Nn}\}$ and a retrieved one $l_R = \{i_{R1}, \ldots, i_{Rm}\}, l_R \in L_C$, of length $n_N = |n|$ and $n_R = |m|$, with the *i-th* item in each being i_{Ni} and i_{Ri}, respectively.

Having found the *local similarity* $sim(i_{Ni}, i_{Rj})$ during the offline phase, the *global similarity* of two playlists, $Sim(l_N, l_R)$, is the *aggregated similarity* of the similarity levels of the items in them. As two similarity aggregations are performed, first defining the similarity level of each item, and then for the lists, we first evaluated which is the most suitable aggregation method for each level. Minimum value computations provided the worse results, therefore the following combinations of maximum and average value computations were tested:

- Max(max): for each item in the first set we find the maximum similarity value among all the items in the other set and then the maximum of those is taken as the set similarity,

$$Sim_{MM}(l_N, l_R) = max_{i=1}^{n}\Big\{max_{j=1}^{m}\{sim(i_{Ni}, i_{Rj})\}\Big\} \tag{4.4}$$

- Av(max): for each item in the first set we find the maximum similarity value among all the items in the other set and then the average value of those is taken as the set similarity,

$$Sim_{AM}(l_N, l_R) = \frac{1}{|n|}\sum_{i=1}^{n} max_{j=1}^{m}\{sim(i_{Ni}, i_{Rj})\} \tag{4.5}$$

- Max(av): we find the average similarity value of each item in the first set and all the items in the other set and then the maximum value of those is taken as the set similarity,

$$Sim_{MA}(l_N, l_R) = max_{i=1}^{n}\Big\{\frac{1}{|m|}\sum_{j=1}^{m} sim(i_{Ni}, i_{Rj})\Big\} \tag{4.6}$$

- Av(av): we find the average similarity value of each item in the first set and all the items in the other set and then the average value of those is taken as the set similarity,

$$Sim_{AA}(l_N, l_R) = \frac{1}{|n|}\sum_{i=1}^{n}\Big\{\frac{1}{|m|}\sum_{j=1}^{m} sim(i_{Ni}, i_{Rj})\Big\} \tag{4.7}$$

After having tested the above aggregation combinations, the best results were obtained when using the *average of the maximum* among the local values. Therefore, for each item in the first list we first determine its similarity level, compared to all the items in the second case, and then the global similarity $Sim(l_N, l_R)$ of the two cases is calculated as the average similarity of the similarity levels of all the items included in them, thus calculated like below,

$$Sim(l_N, l_R) = Sim_{AM}(l_N, l_R) = \frac{1}{|n|}\sum_{i=1}^{n} max_{j=1}^{m}\{sim(i_{Ni}, i_{Rj})\} \tag{4.8}$$

As it can be seen, as the sum of similarities is averaged by the length of the new list, the used similarity metric is not symmetric, meaning that two lists l_N and l_R may have $Sim(l_N, l_R) \neq Sim(l_R, l_N)$.

As explained before, HybA searches for the k playlists that maximise (4.3), thus those that fulfil the following equation:

$$l' \in L_C : \forall l_R \in L_C,\ l' = argmax\{Sim(l_N, l_R)\} \ \Rightarrow \\ l' = argmax\Big\{\frac{1}{|n|}\sum_{i=1}^{n} max_{j=1}^{m}\{sim(i_{Ni}, i_{Rj})\}\Big\} \tag{4.9}$$

When the aim is to identify the exact songs that a user would select, the data sparsity and the computational complexity, usually limit the performance of recommendation methods. On the other hand, if the focus is set on recommending relevant songs, these would possibly be the ones belonging to the same song clusters with the ones that a user had selected and not only the selected ones. Therefore, identifying first the song clusters, being artists or music styles, that would fit within a started playlist is of high importance. Thus, first the ability of the recommender to correctly identify the song clusters that would fit within started playlists, was evaluated. Finding the *k* most similar past lists using (4.9) and recommending the music items in them, has been found to be efficient when treating more abstract descriptions, like song clusters based on their music styles or artists (Gatzioura & Sànchez-Marrè, 2017a).

Extending this initial model, HybA first uses a higher abstraction level for the identification of the most similar playlists based on their general characteristics, like their concept and the music styles of the songs in them. Using the graph connections among playlists and songs, and songs with their styles, playlists can be treated as distributions over songs or as distributions over song styles (Gatzioura, Jorge, et al., 2018). Thus each playlist can be written as $l = \{s_{j1}, \ldots, s_{jm}\} \in L_S$ where,

$$i \in I,\ l \in L_C,\ s \in S :\ \forall i, e(i,l) \in E,\ e(i,s) \in E \rightarrow e(l,s) \in L_S \tag{4.10}$$

Therefore, the similarity of playlists is calculated based on the music styles in them, and the most similar past playlists are found from the following equation:

$$l' \in L_S :\ \forall l_R \in L_S,\ l' = argmax\{Sim(l_N, l_R)\} \ \Rightarrow \\ l' = argmax\Big\{\frac{1}{|n|}\sum_{i=1}^{n} max_{j=1}^{m}\{sim(s_{Ni}, s_{Rj})\}\Big\} \tag{4.11}$$

Where s_{Ni} and s_{Rj} are the music styles of the song clusters that songs i_{Ni} and i_{Rj}, belong to, respectively.

4.9.2 Candidates' Ranking

As the scope of HybA is to deliver relevant playlist continuations, while addressing additional aspects beyond accuracy, more dimensions related to playlist quality and music perception, have been incorporated into the reuse process. More precisely, in addition to playlists' similarity levels, their coherence and diversity degrees are evaluated. The aim was to evaluate and include the most adequate of them, in a manner that the overall recommendation accuracy would not decrease, while the user would perceive an added value in terms of experience.

In general, due to its reasoning model, HybA already during the retrieval phase extracts the most similar, to a started one, playlists. Thus, for example if a very diverse list has been started, playlists with similar specifications, being very diverse, will be retrieved. However, various alternatives of HybA were designed and tested, with aim to emphasize on these aspects and reach possible improvements. The quality related attributes are incorporated at the second phase of the reasoning process, after the candidate lists have been found based on their conceptual and music style similarities, to ensure that a minimum relevance level has been reached, and then its quality may be further enriched.

Variations of the HybA recommendation approach

Let $L'_S \subseteq L_C \subseteq L$ be the set of playlist candidates that have been found based on their similarity with the seed list, using equation (4.11).

HybA then assigns to each playlist $l_R \in L'_S$ a rating r_{l_R}, $0 \leq r_{l_R} \leq 1$, being a function $r(l_N, l_R)$ of characteristics of both the started and the candidate playlist, and looks for the candidate playlists with higher ratings. Therefore, ranks playlists in descending order of the corresponding rating like in equation (4.12):

$$l'' \in L'_S : \forall l_R \in L'_S,\, l'' = argmax\{r_{l_R}\} = argmax\{r(l_N, l_R)\} \tag{4.12}$$

Several variations of HybA have been designed and tested, each placing more emphasis on a different quality dimension, therefore using a different rating function. To sum up, depending on the application scope, the following versions of the HybA RS have been implemented and could be used:

- *HybA*: The emphasis is set on the similarity of the started and the retrieved playlists *(MaxSim)* and the lists in L'_S are ranked on descending order of r_{l_R}, being:
$$r_{l_R} = Sim(l_N, l_R) \tag{4.13}$$

- *HybA-d*: The emphasis is placed on both similarity and diversity *(Max-Div)*. From the playlists in L'_S those with higher diversity degrees are rated higher, using:
$$r_{l_R} = Sim(l_N, l_R)Div(l_R) \tag{4.14}$$

- *HybA-c*: The emphasis is placed on both similarity and coherence *(Max-Coh)*. Similar to HybA-d, now playlists in L'_S are rated using:
$$r_{l_R} = Sim(l_N, l_R)Coh(l_R) \tag{4.15}$$

- *HybA-dd*: The emphasis is placed on similarity and diversity. However, more than maximizing, the aim is to follow the diversity degree observed in the started playlist, l_N. More precisely, the aim is to minimise the difference of the diversity in the new l_N and a retrieved l_R list *(MinDDiv)*. Let this difference be $dDiv(l_N, l_R) = |Div(l_N) - Div(l_R)|$.
Therefore the playlists in L'_S are rated using:
$$r_{l_R} = Sim(l_N, l_R)(1 - dDiv(l_N, l_R)) \tag{4.16}$$

- *HybA-dc*: Similar to HybA-dd but with aim to follow the coherence of the started playlist *(MinDCoh)*. Let the difference in the coherence levels of the new l_N and a retrieved l_R list be $dCoh(l_N, l_R) = |Coh(l_N) - Coh(l_R)|$. The playlists in L'_S are rated using:

$$r_{l_R} = Sim(l_N, l_R)(1 - dCoh(l_N, l_R)) \tag{4.17}$$

- *HybA-db*: Similar to the previous two, but evaluating both the differences of diversity and coherence levels *(MinDBoth)*. The relative importance of these dimensions is defined by the corresponding weights w_C and w_D, with $w_C + w_D = 1$.
  The playlists in L'_S that follow both the coherence and diversity degrees of the started playlist are now rated higher, as:

$$r_{l_R} = Sim(l_N, l_R)\big(w_C(1 - dCoh(l_N, l_R)) + w_D(1 - dDiv(l_N, l_R))\big) \tag{4.18}$$

  With aim to achieve a balance between those dimensions equal importance was placed on them, setting $w_C = w_D = 0.5$.

As it can be seen, two variations of HybA aim at maximizing, apart from similarity, additional quality parameters. More specific, as the coherence of a playlist is considered as an important parameter affecting its quality, in *HybA-c*, among the most similar playlists the more coherent ones are rated higher. On the other hand, in *HybA-d* instead of coherence the more diverse among the similar playlists are rated higher.

However, each person at the moment of creating a music playlist has a specific cognitive state in mind, in terms of music styles and characteristics, affecting his/her actual preferences and cognitive perception of the playlist quality. Rather than independently selecting an optimisation factor, it would be desired to have similar tendencies in the recommended and the seed playlist. For example, if a user that has started a playlist that consists of songs only of one or two artists, might find useful the recommendation of some of their newest songs, or from some less known albums, or by a really similar artist. On the other hand, a user having started a list, with a lot of different artists and a bigger variety of styles would probably find boring the previous recommendations.

Therefore, the rest of HybA's versions tend to follow the exact tendencies observed in the started list. Instead of selecting the playlists that maximise the degree of one quality related parameter, their target value is defined by the new list from the candidate playlists. In *HybA-dc* and *HybA-dd* playlists with coherence or diversity degrees closer to the corresponding ones observed in the seed list are rated higher. Finally, in *HybA-db* candidate playlists having similar levels of both diversity and coherence with the seed list are rated higher. Then, these playlists are ranked based on their final total rating.

As it will be shown in the next chapter, all the above variations of HybA have been found to perform well and similarly. Among those, *HybA-bd* was found to have the most "balanced" performance, in terms of achieving good accuracy results while having a good trade-off between coherence and diversity. Therefore, this variation was used for further testing and comparison with other recommendation techniques.

4.9.3 Recommendations' generation

Last but not least, an important process in a case-based recommender is the way the past cases are reused, in order to formulate the actual recommendation. It may be an exact reuse of a past case solution or a composition of a new one, using the retrieved ones and a specific generative processes.

In the developed system, two reuse methodologies were initially tested, rating each time the candidate music items based on their appearance *frequency* or based on the total *rating* (depending on the system's version) of the case(s) in which those appear. The initial testing has found the rating-based one to perform better, therefore it was selected and further used.

Moreover, after having identified the set $L'' \subseteq L'_S \subseteq L_C \subseteq L$ of the k higher rated playlists, like previously described, HybA extracts the recommendation candidates, the set of items I'', that were initially placed in those playlists, as

$$\forall i'' \in I'',\ l'' \in L'' \text{ and } e(i'', l'') \in E \tag{4.19}$$

These candidate items are rated based on aggregated final rating of the candidate playlist(s) in which they appeared. Given that r_{l_j} is the rating assigned by HybA to a playlist $l_j \in L''$, the final score r_{i_k} of a candidate music item $i_k \in I''$ can be computed as:

$$r_{i_k} = \sum_{l_j \in L'', e(i_k, l_j) \in E} r_{l_j} \tag{4.20}$$

The candidate items are finally ranked based on the total score assigned to them. Depending on the HybA version, this score is a function of the similarity ratings of the candidate playlists in which the item appeared, and possibly of the diversity and/or the coherence level of those lists. These parameters are considered of high importance, as they could affect the user's perception of the playlist and influence his/her opinion on the provided recommendations.

Including additional dimensions of the problem into the recommendation model may sometimes lead to a slight decrease of the recommendation accuracy (Shi, 2013). On the other hand, depending on scope and the user's selection history, these factors are able to increase the user experience that is of equal importance in modern recommender systems. Furthermore, as it will be presented in the experimentation chapter, HybA is able to achieve a balanced performance in terms of relevance and quality. It recommends playlist continuations of coherence and diversity levels similar to the started lists, without facing an important decrease, and even sometimes showing slight improvements, in the recommendations accuracy.

4.10 Conclusions

In this chapter, the idea behind the HybA recommendation system, its goals and main functionalities have been presented. This hybrid CBR algorithm forms the final version, after gradual testing and improvements that have been also described in this chapter, of the core of a hybrid RS for automatic playlist continuation.

Starting from a notion similar to MBA and probabilistic topic models, that songs usually placed together into playlists would follow some common patterns and share cognitive characteristics, this algorithm aims to identify the set of songs to properly complete a started playlist. In contrast to most MRSs, HybA does not evaluate user-item relations, music sound attributes or user similarities, but aims to address similar cognitive concepts. More than consisting, only of relevant songs, the delivered recommendations are designed to compose a pleasant listening result for the user by addressing the *semantic concept* and the *cognitive tendencies* observed in the started list.

In order to better capture the general characteristics of a playlist, influenced by the music styles of the songs in it, and by the user's perception of the context in which it was consumed, the term *"playlist concept"* has been introduced. This term extends the notion of explicit context, and is used to perform a pre-filtering of the playlist case base with aim to improve the system's response time, as well as to address more demanding requests.

Among the *beyond accuracy* dimensions, related to the user perception of playlist quality, evaluated and incorporated to the system are the playlist diversity and coherence degrees. From the definitions and scopes of those parameters, indications of negative relations have been detected. Therefore their relative importance and their desired levels have to be specified from the started playlist, like in the HybA-db algorithm.

Chapter 5

Evaluation

The aim of this chapter is to present the evaluation results that justify that the proposed RS is able to address the research questions stated in chapter 1.

In order to evaluate the proposed RS through its different versions, various experiments were performed on real streaming and playlist datasets. First, its performance was evaluated subject to various parameters, and after having specified the optimal values of those, the designed system was compared with some of the commonly used recommendation techniques. From its initial design, the aim of HybA was the automatic generation and recommendation of playlist continuations, being relevant with the started playlist, while also forming a pleasant listening result. Therefore, for its evaluation both accuracy and quality-related metrics have been used.

The evaluation results presented in this chapter correspond to the various implemented recommendation approaches and follow the evolution of HybA, as described previously in chapter 4. For the testing of the first three approaches, namely MusCBR, CMusCBR and MusHR, a relatively small and well defined music database was used. Based on the observed results and directions, each time changes and improvements were implemented until the final version of HybA.

Furthermore, the testing of this final version took place during a research stay performed at the LIAAD[1] group associated with INESC TEC[2] Institute, located in Porto, Portugal. Larger databases were used, in order to confirm whether the initial smaller scale results and observed tendencies had a solid basis, as well as to test and improve, the scalability of the designed schema. In addition, as the initial dataset contained metadata related with the songs and additional information related to the playlists (the users and the time moment on which those were constructed), the performance of the system had to be evaluated also on datasets containing less structured information.

More information on the datasets, the compared algorithms and the used evaluation metrics, followed by the experimentation results, are presented in this chapter. The results of the three initial versions of the system can be found in section 5.3. Following, the evaluation of HybA, the final version of the proposed recommendation system is presented in section 5.4 in more detail. The performance of its variations related to contextual and conceptual dimensions, and

[1]http://www.liaad.up.pt/

[2]https://www.inesctec.pt/en

beyond accuracy parameters, is analysed in section 5.4.2, while its comparison with other recommendation techniques can be found in 5.4.3. Next, in figure 5.1, an overview of the complete experimentation process is presented.

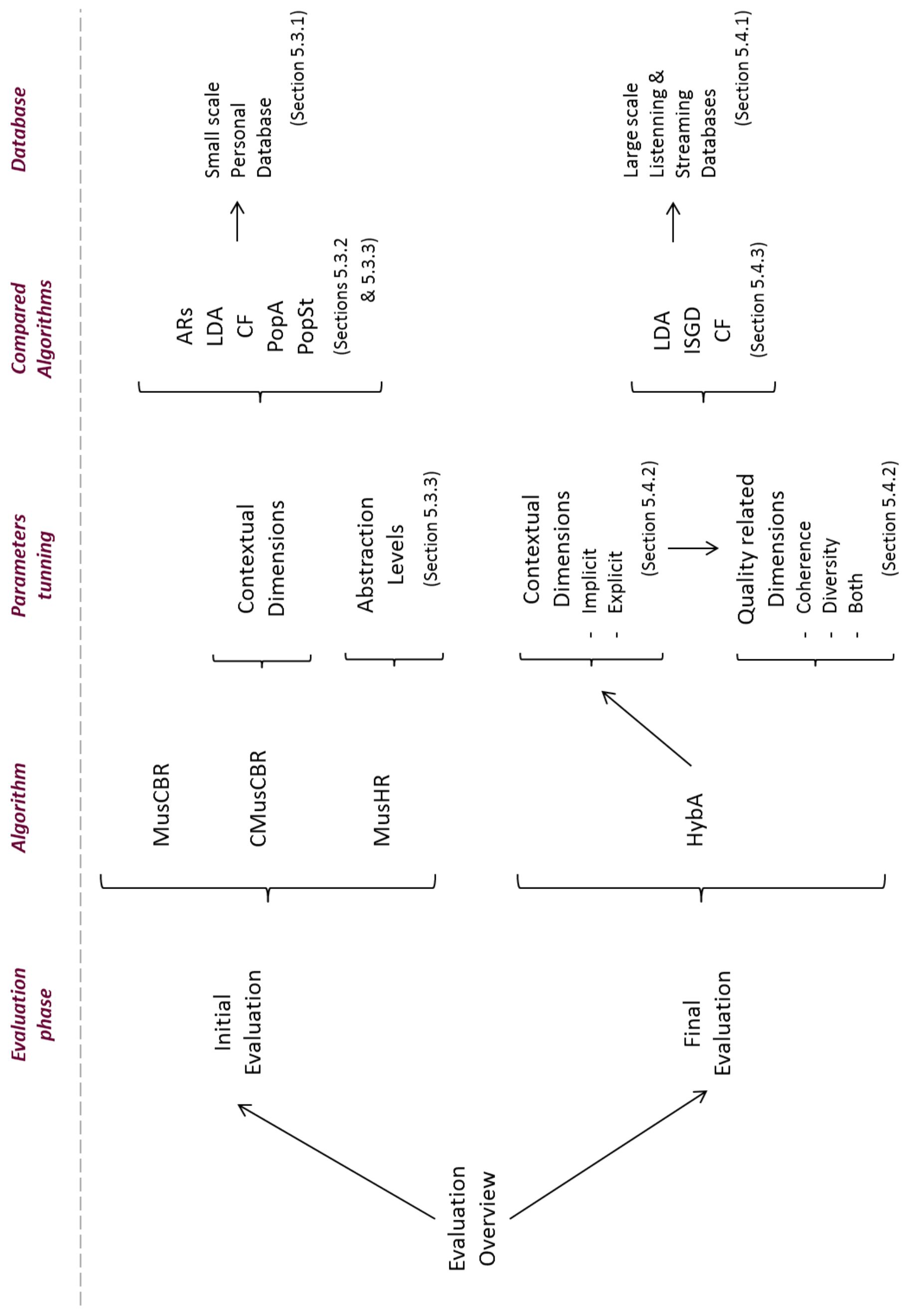


Figure 5.1: Evaluation process overview

5.1 Evaluation metrics

In general, the evaluation approaches of music playlist recommendations can be organised into the following four categories: *user studies, log analysis, objective measures and comparison with hand crafted lists* (Bonnin & Jannach, 2014). More specifically:

- *User studies* help us to determine the perceived quality of playlists by the users. Among the common drawbacks of this method may be the song popularity bias. Many times users when evaluating lists based on their metadata, focus on the songs they already know, or the styles they are more comfortable with. On the other hand, when users are asked to listen to the whole playlists in order to evaluate their quality, this methodology may be time consuming and expensive. In addition, if the users' set size is small, the results may not be of the desired confidence.

- *Log analysis* may be used as an alternative in order to implicitly obtain the acceptance of playlists by users, like for example through the times of listening to a list or the use of like/dislike expressions.

- *Objective measures* are measures that can be automatically computed for a given playlist, and intent to approximate the quality that a user would perceive from listening to it. Such a measure could be the diversity or homogeneity of a playlist, the novelty level, the consistency or smooth transitions between the included items, etc. However, as in other methods based on users' predicted utility or quality, these methods may be heavily affected by the real differences, between the objective and subjective perception of quality and users' needs, that many times may be difficult to capture.

- *Comparison of the patterns* in the generated playlists with the real ones is also being used, as APG aims to generate playlists being similar, according to some criteria, to the manually created ones. This comparison is done mainly by using information retrieval accuracy metrics, like precision, recall, hit rate, etc., or average log likelihood when the generation process is based on some probabilistic method.

5.1.1 Accuracy-based

As the scope of APC is to automatically generate playlists being similar to the manually created ones, the comparison of the patterns in the recommended and the real playlists is usually evaluated using IR accuracy metrics.

Moreover, the used datasets were divided each into a training (80%), used for building the recommendation models, and a testing (20%) part used for evaluation purposes. As user tastes may change over time, the time order of the recorded events was respected. Therefore, the first part of each dataset was used for training the recommendation models, and the rest for testing purposes. The usual cross fold validation was not performed, as it could lead to using more recent events to predict earlier ones, that would bias the experimental results. In addition, due to their reasoning processes the majority of the tested algorithms have a stable performance through experimental repetitions. As an exception, the LDA model, due to its probabilistic nature, may have slight differences in repeated experiments.

Furthermore, the playlists in the test part are divided into two parts. Each time the initial part of the playlist is kept and the rest is hidden, that would be equivalent to asking a user to submit a new playlist, and based on it recommendations are generated. The ability of the various recommendation algorithms to correctly identify the hidden items is then evaluated, and compared, using IR metrics like average precision, recall and F-measure.

- *Precision* is defined as the ratio of the number of correct recommendations: the missing items that were successfully identified, over the total number of recommendations.

$$Precision = \#RelevantRecommendedItems/\#RecommendedItems \tag{5.1}$$

- *Recall* is defined as the ratio of the number of relevant items that are present in the recommendation set, the correct recommendations, over the total number of relevant items.

$$Recall = \#RelevantRecommendedItems/\#RelevantItems \tag{5.2}$$

- *F-measure* is defined as the harmonic mean of precision and recall.

$$F - measure = \frac{2PrecisionRecall}{Precision + Recall} \tag{5.3}$$

- *Hit ratio* can be used as a more general metric to measure the percentage of times that a user would get at least one correct recommendation.

$$HitRatio = \#RelevantRecommendations/\#Recommendations \tag{5.4}$$

Although, due to the experimental settings where from the test part we “hide” and recommend the same number of items, the values of precision and recall appear very close, we rather use precision to evaluate the generated recommendations. This is due to the fact that our focus is on the accuracy of the presented items, rather than on the percent of “relevant” items that are retrieved. Moreover, the number of relevant items could be high, exceeding the number of items that were placed in a specific list.

However, the problem when using the IR accuracy-based evaluation metrics is that they are “too strict”. These methods classify recommendations as successful only when the hidden items are correctly identified. Therefore, when similar items, probably relevant, that could make good recommendations and possibly were not selected due to popularity-bias, are recommended, those are not positively evaluated. In addition, the recommendation of song clusters was also tested in the initial versions of the designed recommender, as an approximation of its ability to correctly identify relevant music styles placed within given playlists.

Furthermore, in order to have a view of “how close” the real (hidden), and the recommended lists were, based on the characteristics of the items in them, more than on their presence or absence, we propose the use of the *average* *similarity* between the real l_{rec} and the recommended l_{real} playlists, of length $|n|$. It can be calculated using equation (4.8), being:

$$Sim(l_{rec}, l_{real}) = \frac{1}{|n|} \sum_{i=1}^{n} max_{j=1}^{n} \{sim(i_{rec_i}, i_{real_j})\}$$

5.1.2 Beyond accuracy

Apart from the usual challenges and limitations of RSs, an important issue in playlist recommendations is to assess the quality of the generated playlists, in terms of coherence, variety and freshness. As the scope of these systems is to address user satisfaction, the best way to determine the quality of a playlist, as perceived by the users, would be to measure their satisfaction. This could be done through monitoring of listening times, user ratings, etc.

However, due to the limitations of these evaluation methods, more objective criteria beyond accuracy, with focus on user experience and related factors should be used. Based on users' expectations from music selections, along with the specific attributes of music items, like their immediately and consecutive consumption, metrics like the playlist's coherence, diversity or its similarity with a desired pattern, would be useful (Bonnin & Jannach, 2013).

- *Coherence* is used to evaluate the pairwise suitability of two consequent items (Castells et al., 2015). Given a song similarity metric, the coherence of a set can be calculated as the average similarity among the pairs of consequent items in a list Lst, calculated as:

$$Coherence = \frac{1}{|Lst| - 1} \sum_{i \in Lst} sim(i, i+1) \tag{5.5}$$

- *Diversity* or *Intra List Diversity (ILD)* is computed as the average aggregated pairwise dissimilarity of the items in the list. Given a similarity or a distance metric between two items, related as $d(i, j) = 1 - sim(i, j)$ the diversity of a list or set of size $|Lst|$ can be calculated as (Adomavicius & Kwon, 2012; Castells et al., 2015):

$$ILD = \frac{1}{|Lst|(|Lst| - 1)} \sum_{i \in Lst} \sum_{j \neq i \in Lst} d(i, j) \tag{5.6}$$

  Or given that item distance and similarity are symmetric we have,

$$ILD = \frac{2}{|Lst|(|Lst| - 1)} \sum_{i \in Lst} \sum_{j \in Lst, j > i} d(i, j) \tag{5.7}$$

- *F-measure of precision and diversity* (Borràs, Moreno, & Valls, 2017) could be used to evaluate the overall performance of the algorithms related to both accuracy and diversity, being:

$$F_d - measure = \frac{2PrecisionDiversity}{Precision + Diversity} \tag{5.8}$$

The balance between diversity and accuracy is still an open problem, as these parameters may have a negative effect on each other. Very diverse recommendations or many unknown items may fail to address users' expectations, being able to harm the system's reputation.

In addition, apart from the effect that increased diversity values may have on recommendation accuracy, the relations among quality-related parameters should be evaluated properly. As already mentioned in chapter 3 on MRSs, coherence and diversity arise from contradictory user goals related to music consumption. Therefore, they may have a negative relation.

From its definition, diversity is a function of the dissimilarity of all pairs of items in a set while coherence is a function of the similarity of the consequent among those pairs. Therefore, increasing the coherence of some items in the list will lead to the decrease of the total average diversity. On the other hand, as diversity depends only on the items in a set, and not on their relative order, while it takes into account the similarities of all the pairs of items in the list, for a given diversity level, still different coherence degrees could be achieved. Coherence takes into account only the pairs of consequent items. Therefore, after specifying the diversity level of a playlist, its final coherence may be further improved by selecting the appropriate items' ordering among the existing combinations.

We follow the notion that coherent lists provide a more pleasant listening result (Kamehkhosh & Jannach, 2017). Thus, they are considered as "better". On the other hand, the desired diversity degree may depend on various parameters like the user's music knowledge, familiarity with the songs' genre, current style, etc. In general, to have a pleasant result, both dimensions should be taken into account, with their relative importance and levels depending on the application domain, the actual user's general preferences or by the seed playlist. Delivering recommendations able to balance between those tendencies is still a challenge.

5.2 Compared Methods

Except from the proposed methodology and its variations, other popular recommendation techniques have been evaluated on the same datasets, in order to evaluate the comparative results, that are also presented in this chapter.

The majority of the datasets used for recommender systems research consist of user-item instances and ratings. Therefore, among the main points that have to be taken into account in this application domain, and when treating data based on transactional logs or interaction events, are:

- There are *no ratings* present. The selection frequency of an item, can be used to capture the degree of preference towards this item. However, the inverse statement, that unobserved interactions would be equivalent to negative ratings, may not be true. It may simply be a consequence of the long tail effect that causes that users focus on a small subset of the offered items, usually because they are not aware of the rest.

- Interactions are modelled as *sessions* or *transactions* that each consists of the set of items selected together, or sequentially, in close time moments by the same user. Therefore, some additional information that reveals these joint selections, like the sequence of items, the corresponding timestamps, the playlist's title or id, has to be taken into account.

Next, the different recommendation algorithms evaluated are presented. Depending on their characteristics, along with the characteristics of the evaluation datasets, they were tested on the first, on the second or on both datasets.

5.2.1 Popularity-based

Due to the long tail distribution of the items, usually observed in music recommendation domains, when focusing only on recommendations' accuracy, the idea of using item popularity seems reasonable, and especially, when addressing the needs of non-expert users it may be efficient.

Bonnin and Jannach propose the use of items popularity along with artist information in user history. They identify user's favourite artists, based on their history, and recommend the most popular items of those artists. As an improvement, they include the most popular songs of the artists usually collocated in playlists with the user's favourite artists, based on the hypothesis that artists' co-occurring frequencies in playlists may serve as a measure of their similarity (Bonnin & Jannach, 2013).

Two popularity-based recommendation approaches have been implemented based on this idea: one based on user's favourite artists (where this information is available) and one based on users' favourite styles, as defined based on the metadata related to songs, except from the artist that performs them. These methods first analyse users' past selections to identify their favourite artists (popA) and styles (popSt) respectively, and then, recommend the most popular items of those. These approaches were tested only on the first dataset, where information on songs' artists was available and there was a popularity effect. In general, those were found to perform well in terms of accuracy, and their performance improves when more abstract item descriptions are used. Especially, when recommending song styles these methods are hard to overcome, as users generally place songs of their favourite styles into their playlists. On the other hand this fact lowers the importance of those results, as they do not provide additional information on user preferences, and cannot overcome the cold start problem as they heavily depend on the analysis of user history.

5.2.2 Rule-based

Association rules' (ARs) and Sequential patterns' (SPs) mining are among the usual techniques applied when searching for co-occurrence patterns in transactional data. Given a transactional database, ARs' mining refers to identifying all the rules or patterns with support and confidence values among some predefined threshold values.

In order to generate AR-based recommendations, the A-priori algorithm was used in order to extract rules, with different minimum support and confidence values, from the playlists in the train part of the datasets. Then, based on these rules, and on the items appearing each time in a started playlist, playlist continuations were generated and recommended. However, in many cases, even with low threshold values, this method was found unable to generate accurate recommendations, probably due to the data sparsity present in the used datasets.

Some common patterns were extracted, but in many cases no significant rules were identified based on those patterns.

5.2.3 Latent Models

As already mentioned before, Semantic Analysis and Latent Models, initially used in text retrieval and classification as well as in Natural Language Processing (NLP) are now also used in other domains related to information retrieval and recommender systems. First a topic model is built based on the past playlists, using a certain number of latent topics. After, the similarity of playlists can be calculated based on the similarity of their topic distributions. Recommendations for a new playlist are generated based on its topic distribution, being the most popular items from its dominant topics.

The Latent Dirichlet Allocation (LDA) implementation of the mallet[3] framework was used to build the topic model used for recommendations. The parameters that heavily influence the performance of such a model, apart from the values of a and b of the Dirichlet model, are the number of topics that will be extracted and the number of iterations that the system is going to perform during its training phase. The *number of topics* used, affects the interpretability of the results, as selecting a low number of topics usually leads to the extraction of few general topics that do not provide any additional information, while selecting a large number may lead to idiosyncratic topics that are difficult to be interpreted. On the other hand, the *number of iterations* that the model will perform before being used for recommendations, heavily affects its stability and final state, referring the final probabilities distribution and the extracted topics. A low number of iterations may result in a less stable model, while on the other hand, an increased number of iterations results in increased computational cost of the system.

5.2.4 Collaborative Filtering

Although CF does not address exactly the given problem, as it is among the most popular and widely used recommendation techniques, its results are also listed here. Furthermore, CF approaches use user-item interactions expressed through user ratings, and generate recommendations based on user similarities without evaluating item co-occurrences in transactions.

As these algorithms first identify the "nearest neighbours" of the active user, among the parameters that mainly influence their performance is the *similarity function* used for the neighbourhood approximation, and the *neighbourhood size*, or *similarity threshold* selected, as a small number of neighbours may fail in generating sufficient recommendations, while a large one may result in many generic recommendations that will finally fail to satisfy the active user.

Users past song sessions, together with the initial part of the playlist in the test part of the databases were analysed, in order to identify users' preferences and the most similar users of each. Based on those, playlist continuations of different sizes were generated for the users in the test part. The Apache Mahout user based CF[4] recommendation algorithm was used, as it was found to have a

[3]http://mallet.cs.umass.edu/
[4]https://mahout.apache.org/users/recommender/recommender-documentation.html

better performance, according to IR metrics, than the Lenskit CF[5] algorithm.

5.2.5 Incremental Matrix Factorization

As Matrix Factorization (MF) is currently gaining ground and being more widely used, especially on large datasets, in order to overcome data sparsity and improve the performance of CF techniques, also the results of a MF algorithm were evaluated.

ISGD, the incremental MF approach presented by Vinagre in (Vinagre, 2016), that treats song appearences as sequential data and works with positive-only feedback, has been tested. ISGD has been found to outperform both other CF and incremental factorization algorithms in most cases while also having a good runtime performance (Vinagre et al., 2014). As in the case of the CF implementation, for MF also users' past sessions and the initial part of their new playlists in the test part of the databases, were used to identify their preferences. Then, the recommendations were generated based on users' song streams.

5.3 Initial Evaluation

5.3.1 Dataset details

For the initial testing we used a real music database containing approximately 3500 songs, mainly rock and alternative, and 3000 playlists of one hour, formed based on them. These lists had been constructed by the author and other radio producers of the Greek internet radio station TrollRadio[6].

In addition, for each song in this database the following information was available:

- its name
- its unique id
- the artist who performs it
- a set of metadata used to specify its style, being:
  - general music category (ex: rock)
  - subcategory (ex: alternative)
  - tempo (low/mid/up/very high)
  - origin (mainly Greek/other)
  - lyrics language (English/Greek/other)
  - vocalist type (male, female, duet)
  - release year (in some cases)

This information has been added by experienced users in the form of tags coming from a specific dictionary. Therefore, it can be used as a source of additional information about the songs. Furthermore, when only these metadata, without

[5]http://lenskit.org/documentation/algorithms/
[6]www.trollradio.gr

the artist that performs the song, are used for the songs' descriptions, these are referred to as *music style*, while when also the artist name is taken into account, those are referred to as *complete style*. Both well known and newer songs, and artists, of similar styles, thus possibly forming pleasant, coherent, playlists, were contained in this database.

The frequency with which the songs, complete music styles, music styles and the artists that perform them were placed in playlists, was also used to calculate the popularity and the long tail distribution of these terms, considering a music item as belonging to the long tail if it appeared less than in 1% of the cases. In table 5.1 below, we present more statistical information on this dataset.

	Number	**Long Tail %**
Events	38934	
Playlists	3245	
Songs	3452	92
Complete Styles	1654	74
Artists	833	51
Styles	114	44

Table 5.1: Small scale dataset

Indeed, as shown in table 5.1, the long tail percentage is lower in higher abstraction levels showing that music items that are found in the long tail are probably characterised by styles, or belong to artists, that users would place in their playlists. It was found than less than 2% of the songs in the long tail were also found in the long tail at higher abstraction levels, highlighting that there may exist songs in this dataset that belong to the long tail, while having characteristics that would probably make them fit within some of the constructed playlists. Therefore, these songs could form relevant recommendations.

In this dataset, there was also some information related to the playlists and the users that reproduced them. Apart from the set of n songs of which each playlist consists, it is also associated with the time moment on which it was made, provided by the timestamp of each song's streaming. From the set of users that formed the existing playlists, each one is associated with the playlists he/she made, along with some basic demographic characteristics, like age and gender.

Specific parameters' values

Except from MusCBR and CMusCBR, for the other recommendation techniques presented in the previous section, their best results, presented in tables 5.2–5.4, were obtained for the following values of their important parameters:

- *ARs-based (ARs)*: The A-priori algorithm was used with minimum support equal to 0.5%, and minimum confidence set to 15%.
- *Latent Dirichlet Allocation (LDA)*: The LDA model was built with 50 topics and performing 1000 iterations.
- *Collaborative Filtering (CF)*: The Pearson Correlation Similarity was used for user similarities, and for each user his/her 5 nearest neighbours were used.

5.3.2 MusCBR and CMusCBR Evaluation

In tables 5.2–5.4 the average precision values for playlist continuation recommendations of different lengths, generated by the various methods can be found. Each time music items are treated at different abstraction levels, namely complete music styles (artist and music style), artists, and music styles.

#Recommendations	**4**	**6**	**8**	**10**	**12**
AR	0.055	0.063	0.052	0.052	0.035
LDA	0.049	0.061	0.068	0.081	0.094
PopA	0.039	0.054	0.058	0.067	0.064
PopSt	0.043	0.048	0.053	0.064	0.064
CF	0.045	0.058	0.068	0.08	0.079
MusCBR	***0.078***	***0.078***	***0.076***	***0.086***	***0.083***
CMusCBR	**0.097**	**0.090**	**0.090**	**0.096**	**0.098**

Table 5.2: Average precision for complete music style recommendations

#Recommendations	**4**	**6**	**8**	**10**	**12**
AR	0.052	0.087	0.108	0.115	0.096
LDA	0.034	0.050	0.075	0.087	0.093
PopA	0.044	0.073	0.098	0.115	0.117
CF	0.048	0.078	0.102	0.118	0.119
MusCBR	***0.095***	***0.103***	***0.115***	***0.127***	***0.131***
CMusCBR	**0.096**	**0.108**	**0.124**	**0.139**	**0.135**

Table 5.3: Average precision for artist recommendations

#Recommendations	**4**	**6**	**8**	**10**	**12**
AR	0.261	0.32	0.363	0.367	0.372
LDA	0.202	0.345	0.167	0.264	0.317
PopA	0.34	0.346	0.336	0.334	0.337
PopSt	0.38	0.402	0.394	0.394	0.395
CF	0.185	0.259	0.292	0.315	0.345
MusCBR	***0.386***	***0.39***	***0.397***	***0.396***	***0.395***
CMusCBR	**0.392**	**0.403**	**0.4**	**0.401**	**0.401**

Table 5.4: Average precision for music style recommendations

When recommending songs, or when using their complete music styles that include both genre and artist descriptions, we are facing increased data sparsity, given that in most playlists there are no songs of the same artist. Therefore, the problem complexity is almost the same. In these cases, the AR methodology was not able to generate recommendations for a lot of lists, due to the low number of underlying patterns among data. On the other hand, when approaching items just through their style, it is far easier to find similar playlists, and the recommendations in terms of styles are far more accurate for all methods.

As it can be observed from the previous tables, MusCBR is able to perform equally and even better than the compared recommenders, even when using more specific item descriptions. Incorporating contextual information, about the playlist construction hour, in the pre-filtering phase of the recommendation model, further improves the algorithm's performance. CMusCBR performs better than the compared algorithms and also delivers improved recommendations compared to MusCBR, that does not evaluate any contextual dimensions of the playlists. Even for less abstract items descriptions, where the majority of the other techniques show a low performance.

However, all the recommenders show a weak performance, according to the used IR evaluation metrics, when recommending specific items, with MusCBR and CMusCBR reaching the highest, being around 5–6%. The results for the recommendations of songs' are presented in the next section for the extended model.

5.3.3 MusHR Evaluation

In this section, the experimental results for the recommendation of lists of different lengths using the MusHR algorithm, are presented, in comparison to other recommendation methodologies.

First, the possible item abstractions that could be used in the Hybrid Music Recommender (MusHR) are evaluated to identify which leads to the best performance. In table 5.5, the recommendations' average precision is shown, when no item abstraction, and when complete music style or artist abstractions, is used. Using MusHR without item abstraction would be equivalent to using MusCBR for direct recommendation of songs, instead of song clusters.

#Recommendations	**4**	**6**	**8**	**10**	**12**	**14**
Complete style	**0.239**	**0.234**	**0.218**	**0.197**	**0.164**	**0.129**
Artist	0.189	0.176	0.162	0.142	0.139	0.142
Without	0.063	0.055	0.051	0.051	0.060	0.058

Table 5.5: Recommendations' precision for MusHR with different abstractions

As it can be observed, changing the reasoning model from one to two levels, thus using more abstract item descriptions for the extraction of candidate playlists, significantly improves the performance of the recommender, both when complete styles or artists are used. Using complete style descriptions for the modelling of music items, and extracting the candidate cases based on those, performs better for almost all recommendation sizes. Therefore, it was used for the rest of the experimentations.

Below, in tables 5.6–5.8, the recommendation results of MusHR are presented in comparison to those of other recommendation algorithms, in terms of average precision, diversity and F_d measure, respectively, for playlist continuations of different lengths. Following, in table 5.9, the average long tail percent in the playlist continuations recommended by the different methods, is presented.

#Recommendations	**4**	**6**	**8**	**10**	**12**	**14**
LDA	0.048	0.065	0.081	0.085	0.084	0.115
PopA	0.018	0.028	0.031	0.033	0.033	0.040
PopSt	0.017	0.018	0.021	0.030	0.037	0.048
CF	0.025	0.031	0.038	0.042	0.053	0.063
MusHR	**0.239**	**0.234**	**0.218**	**0.197**	**0.164**	**0.129**

Table 5.6: Average precision for the different methods

#Recommendations	**4**	**6**	**8**	**10**	**12**	**14**
LDA	**0.618**	0.535	**0.639**	**0.612**	0.556	**0.635**
PopA	0.287	0.353	0.512	0.521	0.450	0.553
PopSt	0.366	0.402	0.501	0.533	0.050	0.55
CF	0.537	0.527	0.575	0.586	0.57	0.616
MusHR	0.54	**0.553**	0.562	0.568	**0.585**	0.600

Table 5.7: Average recommendations' diversity for the different methods

#Recommendations	**4**	**6**	**8**	**10**	**12**	**14**
LDA	0.089	0.116	0.144	0.149	0.146	0.195
PopA	0.034	0.052	0.058	0.062	0.061	0.075
PopSt	0.032	0.034	0.040	0.057	0.043	0.088
CF	0.048	0.059	0.071	0.078	0.097	0.114
MusHR	**0.331**	**0.329**	**0.314**	**0.293**	**0.256**	**0.212**

Table 5.8: F_d-measure of the recommended playlists for the different methods

#Recommendations	**4**	**6**	**8**	**10**	**12**	**14**
LDA	**0.731**	**0.738**	**0.723**	**0.642**	**0.526**	0.196
PopA	0.0	0.0	0.0	0.0	0.0	0.035
PopSt	0.0	0.0	0.0	0.0	0.0	0.0
CF	0.13	0.138	0.150	0.152	0.20	0.19
MusHR	0.314	0.34	0.361	0.355	0.324	**0.266**

Table 5.9: Recommendations' long tail percent for the different methods

As it can be seen from the initial recommendation results, the proposed methodology seems able to deliver results of improved accuracy while maintaining a diversity level similar to that of the real playlists, that was 55–60%. In addition, 25–35% of the items in the playlist continuations recommended by MusHR, come from the long tail, while the target was set on achieving at least 20% in each playlist.

In general, the majority of the commonly used techniques showed a low performance. Especially ARs, even when using low threshold values for minimum support and confidence to enable the extraction of rules, failed to generate relevant recommendations. This is the reason why the results of this method were finally not included. Session-based methods (LDA and MusHR), that focus on the characteristics of entire playlists, were found to significantly outperform the methods based on user preferences. Although LDA recommendations may be more diverse, and retrieve even more items from the long tail, this has as a result a significant accuracy drop. In addition, this method due to its probabilistic nature has been found to show a less stable performance. As expected, the popularity-based and CF approaches are almost not able to provide recommendation of items coming from the long tail. Although they present some diversity in their recommendations, this is mainly due to the diverse preferences of the users.

Despite the fact that MusHR does not perform neither a contextual pre-filtering nor provides additional support to diversity, given that its reasoning process is based on the similarity of playlists, in general it manages to identify the lists with tendencies similar to the list that has to be completed. However, in order to perform well also on larger datasets, and/or datasets containing songs not associated with well-defined attributes, this algorithm was further extended to HybA, whose evaluation is presented in the next section using larger scale real music datasets.

5.4 Final Evaluation

The focus of this section is two-fold: first, to present the recommendation results for the variations of HybA recommender and evaluate the influence of the different parameters on their performance, and secondly, to compare its performance with other recommendation techniques.

In section 5.4.2 we present the results of the different versions of HybA and select the most adequate, and then in section 5.4.3, its comparison with other RSs, using both accuracy and quality related metrics, is shown.

5.4.1 Datasets' details

The databases that were used for evaluation of the HybA algorithm are music databases containing information on users streaming specific tracks or putting those into their playlists.

Moreover, four datasets of the Palco Principal[7], a Portuguese music social network that gathers non-mainstream musicians with fans, were used. This website allows free music streaming and users can organise their favourite music tracks in personal playlists. It is characterised by a high long tail presence, as the majority of the songs presented are not popular songs. Furthermore, the percentage of long tail items is higher than 99% in all datasets. Thus, evaluating the number of long tail items recommended in each playlist continuation cannot be considered as an indicator about the recommendations' novelty and quality. This was also the reason why popularity-based algorithms did not provide considerable results. More information on the used datasets' statistics can be found in table 5.10.

[7]http://palcoprincipal.com/

Dataset	**Listening1**	**Listening2**	**Playlisting**	**Plc**
Events	1171849	295044	111942	508705
Songs	29786	22986	26117	25262
Users	21815	5543	10392	20875
Playlists	86174	22108	22132	253415
Songs/Playlist	13.6	13.35	5.06	2
Playlists/User	3.95	4	2.13	12.14
Users with one playlist	11544	2673	6840	7909
Test users	8166	1286	2141	3033
Test users with one playlist	5540	670	1458	273

Table 5.10: Palco Principal datasets

From these databases[8], the first two are streaming logs, containing information on users listening to songs on specific moments, while the others contain information on users adding tracks to their personal playlists at a given time moment. From those, only the forth (Plc) contained additional content information on song features, namely artist and genre. For the rest of the datasets, latent features from usage data were used to characterise the songs. On the other hand, the forth dataset did not contain information related to the time moment that the playlists were created. Therefore, it could not be used for the evaluation of the contextual dimensions of the problem.

The first two datasets (Listening1 and Listening2) have a similar distribution of events. Especially the average playlist length and the average number of playlists per user are almost equal. Additionally, from the user interactions in the first three datasets, approximately 50% of the users had only one session, while more than 50% of the users in the test part had only one interaction. In general, the high percent of "new" users, without historical data to derive their preferences forms a problem for user-based techniques. On the other hand, in the last dataset (Plc) users, in average, appear to have performed more but shorter sessions. Furthermore, there is a smaller percentage, compared to the other datasets, of users with only one sessions. Finally, from the users in the test part, less than 10% were new users users. Thus, the new user cold start problem should have less impact on this dataset. The short average length of the playlists could negatively effect the recommendation results when generating longer playlist continuations.

Although coming from platforms situated in different countries and having a significant size difference, both the initial and final testing datasets contained mainly alternative music and had the scope to enable less popular songs, and artists, to be reproduced. According to the analysis of music genre preferences accross countries, provided by Schedl, rock and alternative genres in general are the most prefered (Schedl, 2017).

Specific parameters' values

When testing the performance of the recommendation algorithms presented in section 5.2, on the large scale datasets, ARs and popularity-based were found

[8]The first three are publicly available from https://rdm.inesctec.pt/dataset

to have a very weak performance. Therefore, their results were not included in this section. This low performance is probably due to the extended long tail and data sparsity that characterises the used datasets were the majority of songs are not mainstream songs and each is listened to only by few users.

For the other methods, their best results that are presented, were obtained when the following parameters' values were used:

- *LDA*: The parameters of the LDA model were set at 200 topics and performing 1000 iterations.
- *CF*: The Euclidean Similarity function was used for the user similarities and the neighbourhood of each user is set as his/her 10 nearest neighbours.
- *ISGD*: The number of factors was set at 250, while the option of recommending already known items was enabled.

5.4.2 HybA Evaluation

We hypothesise that:

1. A "conceptual" rather than a "contextual", based on explicitly described parameters, clustering and pre-filtering of the problem case base would perform better, and
2. More dimensions related to the playlists should be incorporated and taken into account during the recommendations' generation process.

Therefore, as a starting point, the influence of the different contextual dimensions on the recommendation results was observed to confirm that the proposed conceptual pre-filtering is able to better characterise, and identify, similar playlists. Afterwards, in addition to playlists similarity, more parameters related to users' perception of playlist quality were incorporated into the recommendations generation phase, and their influence on the RS will be presented. Both accuracy (precision, hit ratio and average similarity) and quality (average coherence and diversity) related metrics are used.

Contextual Dimensions

We present graphically the results of HybA, using the proposed conceptual clustering and pre-filtering, compared with the use of three different contextual clusterings. More specific, for a new playlist, *cntxT*, uses the cluster L_H of playlists generated in close hours, *cntxM*, the cluster L_M of playlists reproduced in close months while *cntxC* uses the cluster L_B that includes playlists generated on close day hour and month with the new playlist. The tests were performed using the first three datasets that contained explicit contextual information related to the playlists' creation time moment.

Next in figures 5.2–5.4, precision, hit ratio and average similarity, respectively, of the different contextualised algorithms are presented.

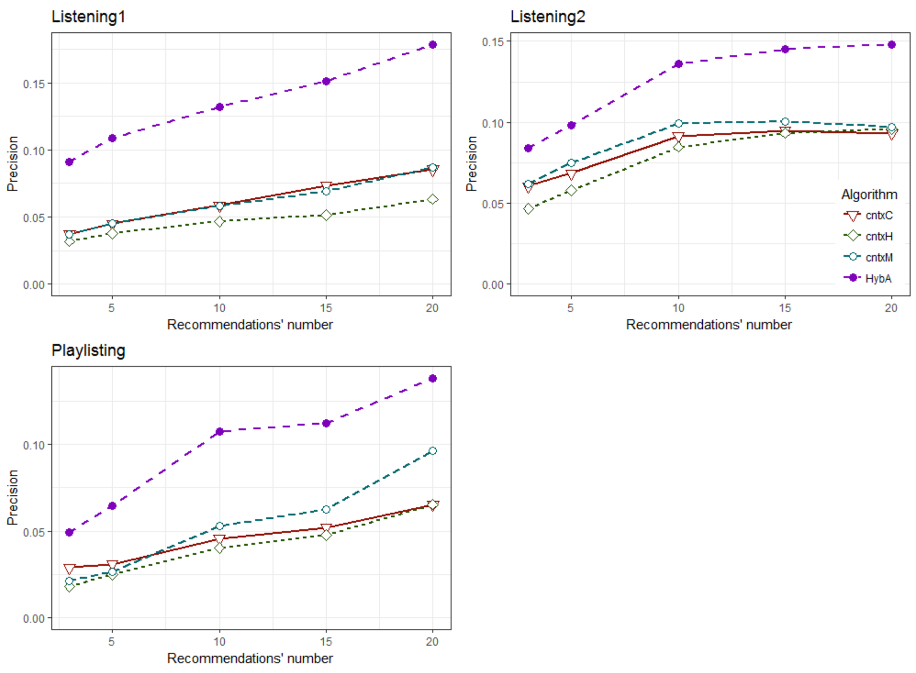


Figure 5.2: Recommendations' precision using different contextual dimensions

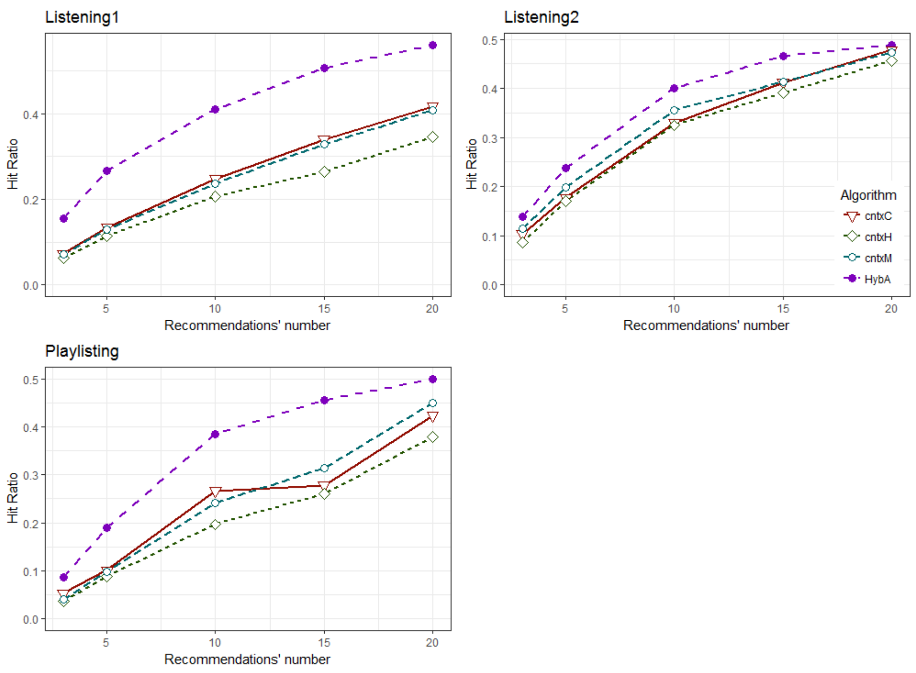


Figure 5.3: Recommendations' hit ratio using different contextual dimensions

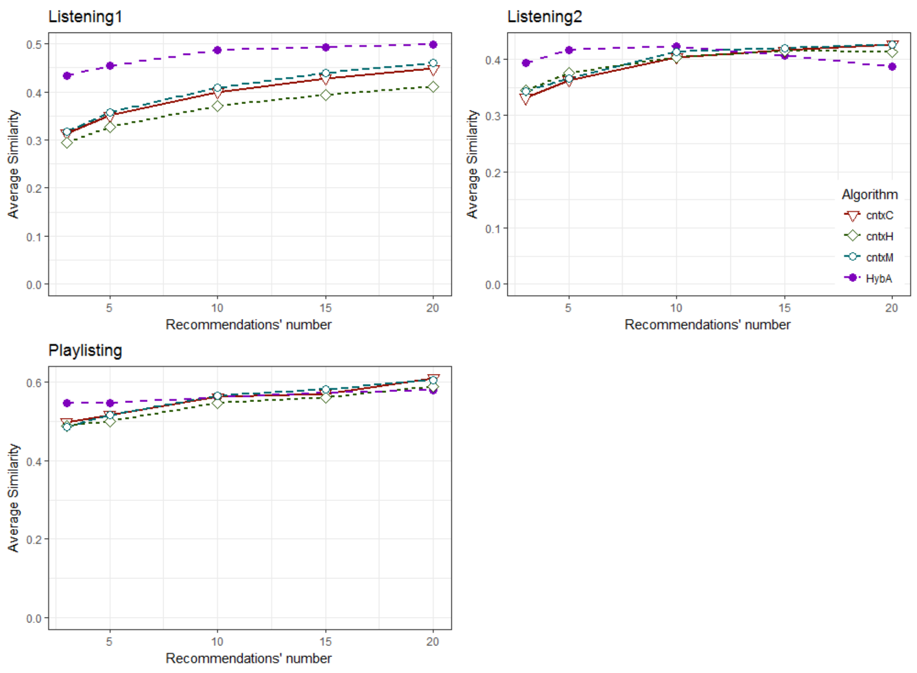


Figure 5.4: Average similarity between the real and the recommended playlist continuations when using different contextual dimensions

As it can be seen, the three pre-filtering approaches, using the explicit contextual parameters related to the playlist generation time, lead to similar results in terms of accuracy. Among those, cntxT, that evaluates only the hour of the day leads to the worst results. cntxM and cntxtC, that also evaluate the month, have a better performance, while their differences vary with the playlist continuation length and the evaluation dataset.

The proposed conceptual clustering and pre-filtering provides an important improvement in precision and hit ratio, while also having equal or better average similarity values.

Figures 5.5 and 5.6, present the average coherence and diversity of the recommended playlist continuations by the different versions. As it can be seen, cntxT, cntxM and cntxC have very similar performances at both dimensions, while HybA generates significantly more coherent playlist continuations, being of better quality.

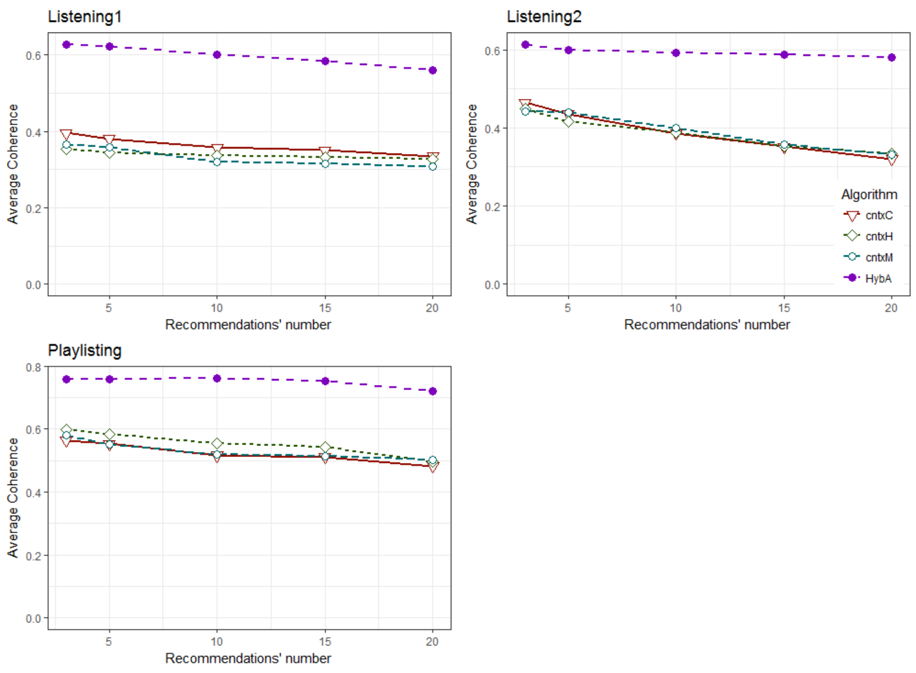


Figure 5.5: Recommendations' coherence using different contextual dimensions

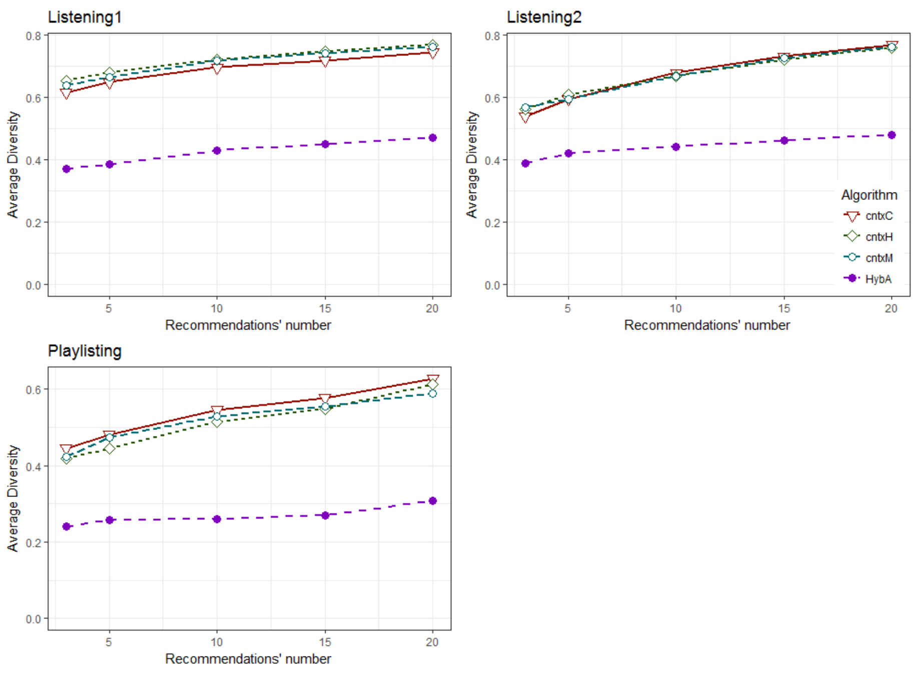


Figure 5.6: Recommendations' diversity using different contextual dimensions

Beyond Accuracy Dimensions

After having confirmed that using a conceptual pre-filtering, rather than explicit contextual information in HybA, leads to improved results, the performance of the different versions of HybA is evaluated.

Starting from accuracy related metrics, in figures 5.7 and 5.8, respectively, we present the precision and average similarity values achieved by the different versions of the RS that place increased emphasis on different quality aspects, for tests performed on the four different datasets.

From these plots, we can observe that all the HybA versions have a similar performance in terms of accuracy. When evaluating using precision (recall and F-measure are closely similar) only HybA-d has been found to perform worse than the others. However, when evaluating using the average similarity, HybA-d performs equally, and even better in one dataset, than the other algorithms. The hit ratios achieved by all the versions of HybA are closely similar therefore their graphical representations are not included.

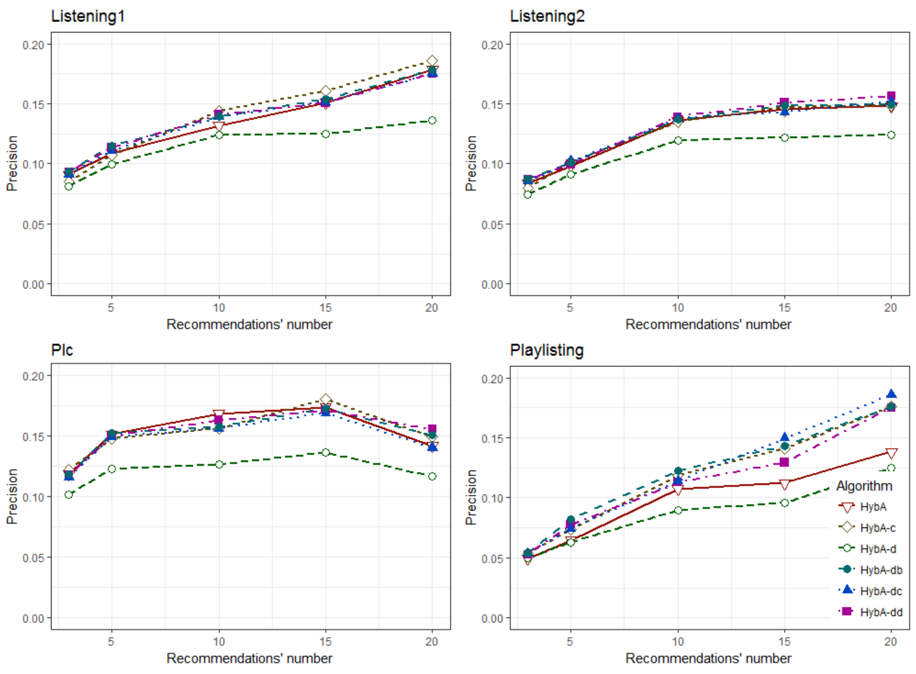


Figure 5.7: Recommendations' precision for the HybA versions

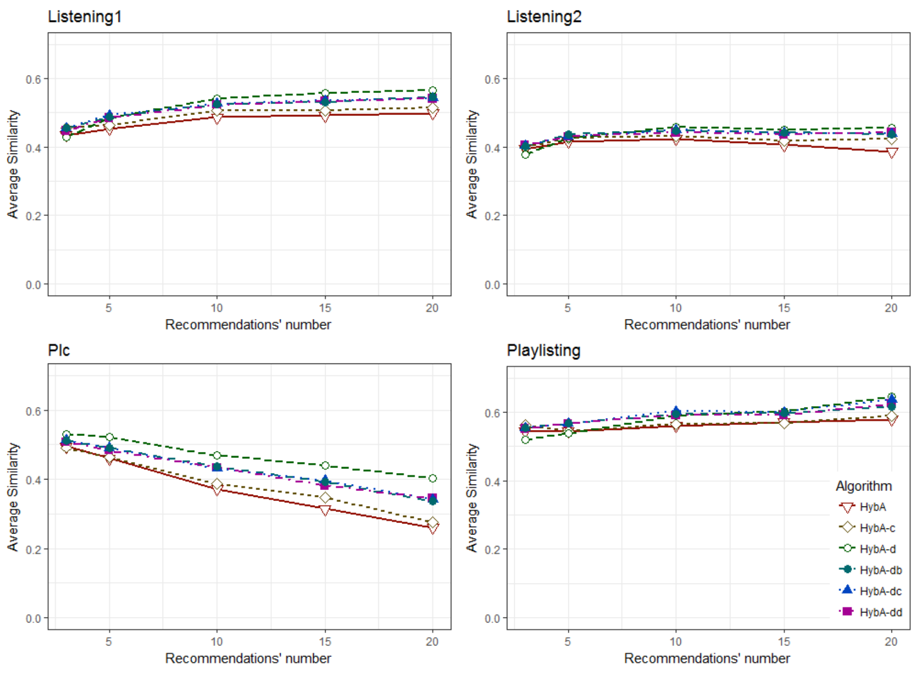


Figure 5.8: Average similarity of the real and the recommended playlist continuations by the different HybA versions

Next, in figures 5.9–5.11, the algorithms' performance as captured by the levels and the relations of quality related parameters, namely coherence and diversity, can be found.

As it can be observed from figures 5.9 and 5.10, where the levels of coherence and diversity of the recommended playlist continuations are shown, depending on the algorithm's scope, there is a clear difference in the levels of those parameters. HybA-c and HybA-d that, except from similarity, aim at maximizing an additional parameter generate playlist continuations of significantly higher coherence or diversity degrees correspondingly. As those parameters depend on the similarity, or dissimilarity, of the items in the playlist, setting the desired level of one of them, a priori, also limits the range of values that the other parameter may have.

In figure 5.11, we present graphically also the relations between average coherence and diversity as achieved by the different HybA variations.

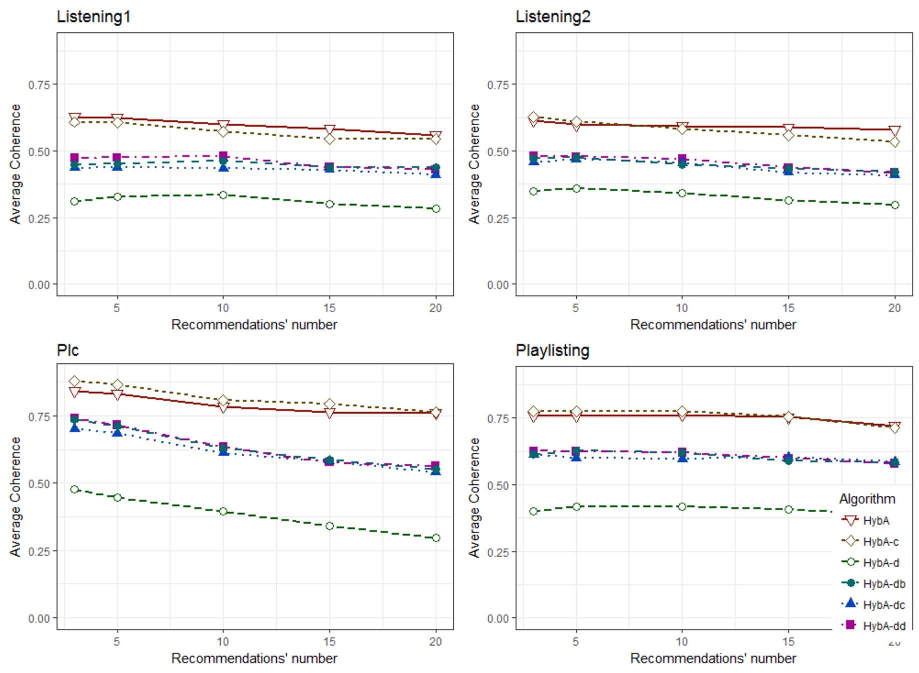


Figure 5.9: Recommendations' coherence for the HybA versions

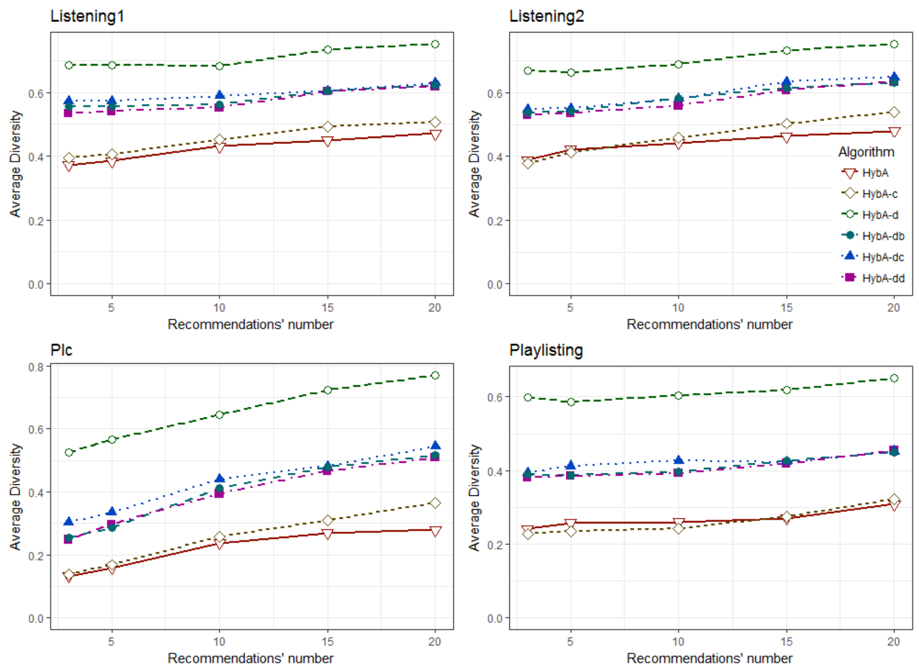


Figure 5.10: Recommendations' diversity for the HybA versions

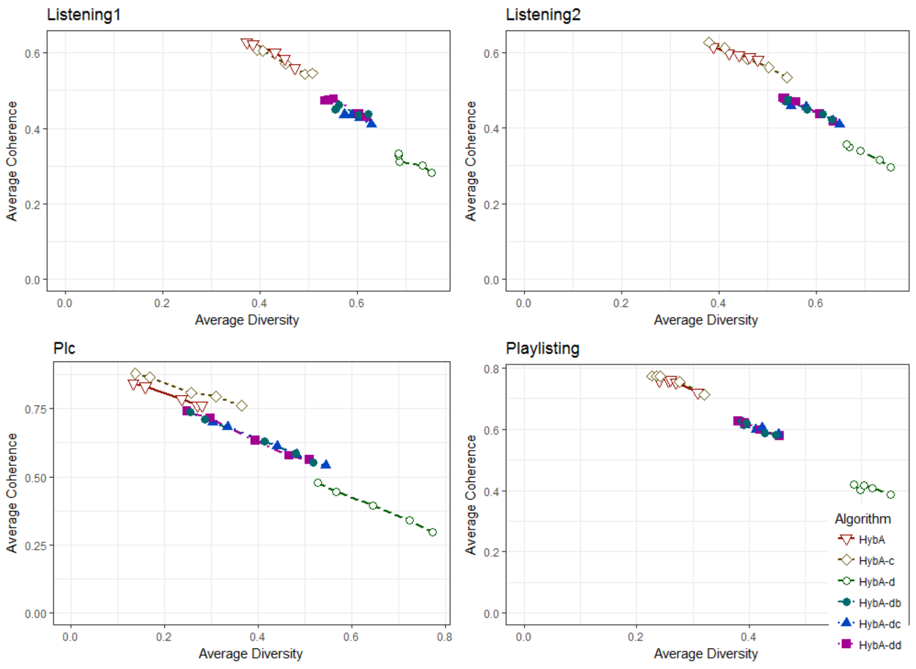


Figure 5.11: Average coherence in relation to average diversity of the various recommended playlist continuations

As it can be seen, when instead of maximizing one of those parameters, the aim is to follow the seed playlist tendencies, the three HybA variations, HybA-dc, HybA-dd and HybA-db, show a very similar, and more balanced, performance, at all dimensions. Among those, we select the HybA-db as it achieves good accuracy results while having a good level of both coherence and diversity.

Finally, we observe that the algorithms that manage increased coherence levels may also provide improved accuracy results, according to IR metrics, compared to those that emphasise on diversity, especially when treating larger lists.

5.4.3 Comparison with other techniques

After having evaluated the HybA versions and selected the HybA-db, as showing the best balance between *accuracy*, *coherence* and *diversity*, its comparison with other recommendation algorithms is presented in this section.

In figures 5.12–5.14 the graphical results for accuracy related metrics, namely *precision*, *hit ratio* and *average similarity* for recommendations of playlist continuations of different lengths, on the four different datasets, can be found. In figures 5.15 and 5.16, the average *coherence* and *diversity* of the recommendations is presented, while in figures 5.17–5.19 the combined performance of the algorithms, in relation to pairs of parameters, is evaluated.

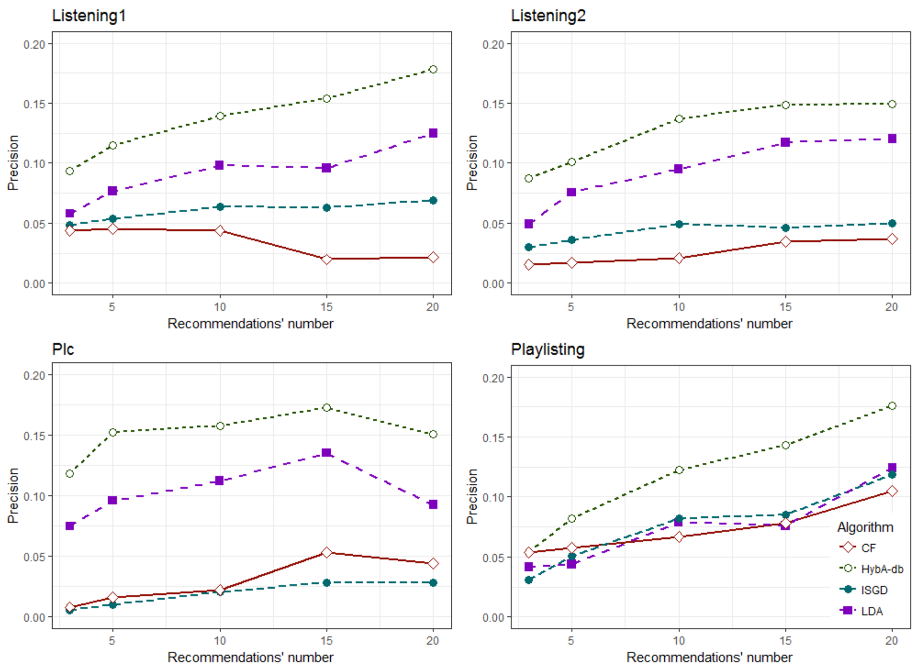


Figure 5.12: Recommendations' precision for the various techniques

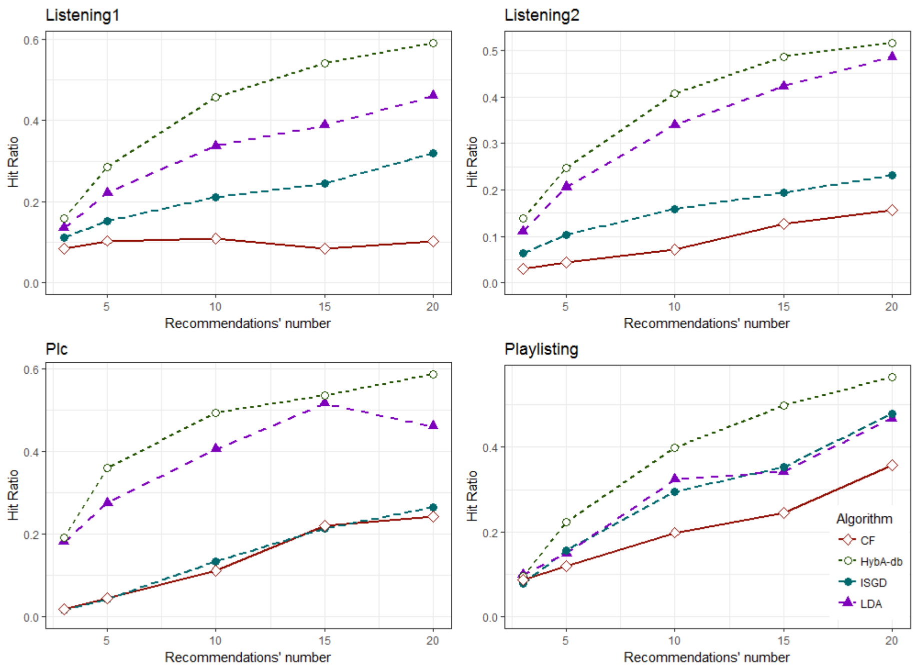


Figure 5.13: Recommendations' hit ratio for the various techniques

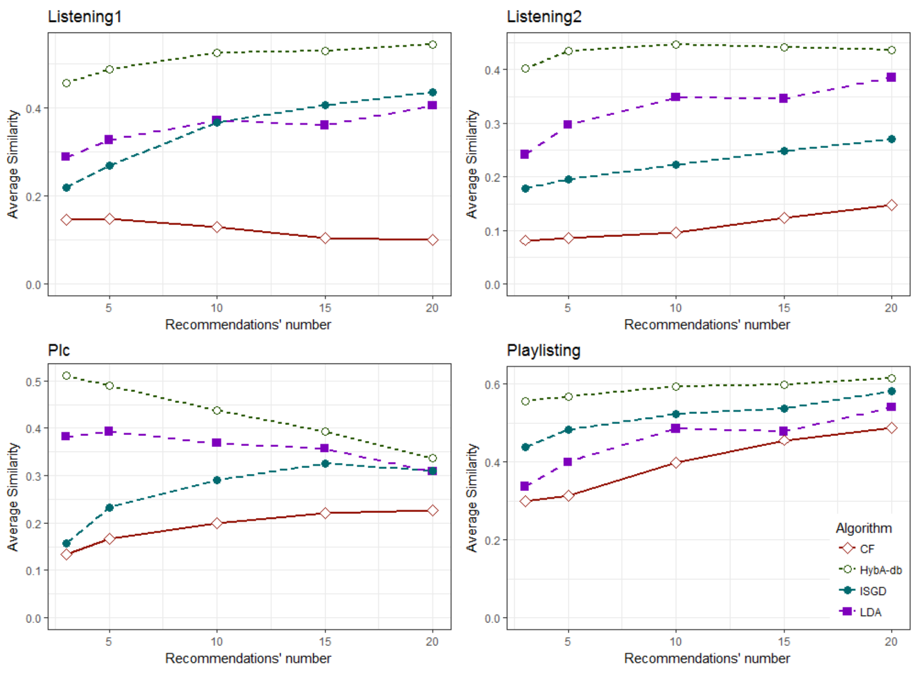


Figure 5.14: Average similarity with the real playlists for the various techniques

From figures 5.12 and 5.13, where precision and hit ratio of the recommended playlists are shown, we observe that they follow similar evolution patterns for all the evaluated algorithms. In addition, as depicted in figure 5.14, average similarity plots are in-line with those patterns, but their corresponding plots have smaller inclinations. Therefore, their values appear as less dependent on the number of recommended items. This fact supports our hypothesis that more metrics, beyond the strict IR ones, could be used to evaluate the quality of recommendations. The combination of hit ratio and average similarity seems able to capture the relevance of the recommendations presented, while being more flexible than precision or F-measure. In addition, average similarity, if correctly combined with novelty, could also serve as an indicator of serendipity.

In general, HybA and LDA, that focus on the entire sessions' characteristics, have a better performance, supporting our hypothesis that conventional recommendation techniques fail to capture the specific cognitive characteristics beyond item co-occurrences in item sets. Especially HybA-db, has been found to outperform the other algorithms in all the used datasets.

Among the compared algorithms, CF had a low performance, probably due to the fact that its focus is on the recommendation of items to users, while the current focus is on the recommendation of items fitting into specific concepts, generated by users, but not restricted to users. Therefore, more than the user-item relationships, mainly concept-item relationships and similarities between concepts have to be identified, rather than the user-concept connections. Although, the ISGD algorithm also works based on user-item relations, as its reasoning is based on data streams it is able to perform better than the pure CF. However, the increased data sparsity and long tail percent in the evaluation datasets should be also taken into account, as it significantly affects the performance of user-based algorithms.

In addition, we observe that HybA-db and LDA follow similar patterns as both methods base their reasoning on probabilistic topic models. However, HybA-db performs a Latent Analysis at a more abstract level, using songs' styles rather than songs, and is further combined with an additional item similarity model, that improves its accuracy and makes it more flexible in handling additional parameters, than pure LDA. LDA, on the other hand, based on the probability distribution of a playlist directly recommends the most popular songs of its dominant topic. The advantage of these models is that they are able to relate items based on their semantic dimensions. Therefore items that may seem as distinct at first, may be related based on their co-occurrences in common topics in the past.

Before presenting the levels of average coherence and diversity in the playlist continuations recommended by the different algorithms in figures 5.15–5.17, in table 5.11, the corresponding values of the playlists in the various datasets can be found. These threshold values were found by evaluating playlists of different lengths, specifically equal to the length of the generated recommendations (3, 5, 10, 15 and 20 music items).

Quality dimensions	**Listening1**	**Listening2**	**Playlisting**	**Plc**
Coherence	0.45-0.52	0.45-0.50	0.50-0.56	0.58-0.63
Diversity	0.53-0.66	0.57-0.66	0.50-0.60	0.45-0.65

Table 5.11: Playlists' coherence and diversity ranges in the various datasets

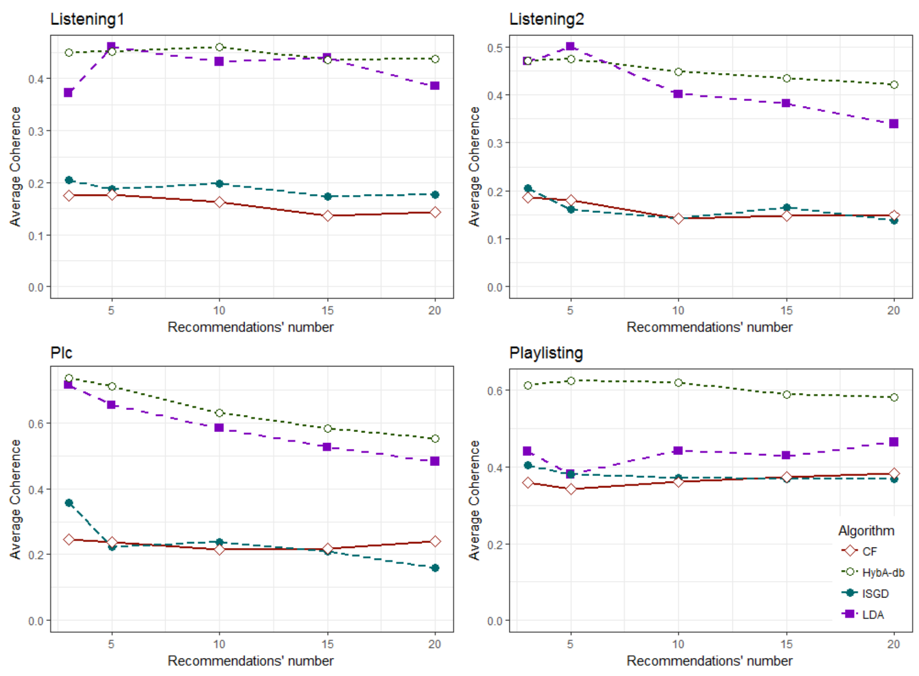


Figure 5.15: Recommendations' coherence for the various techniques

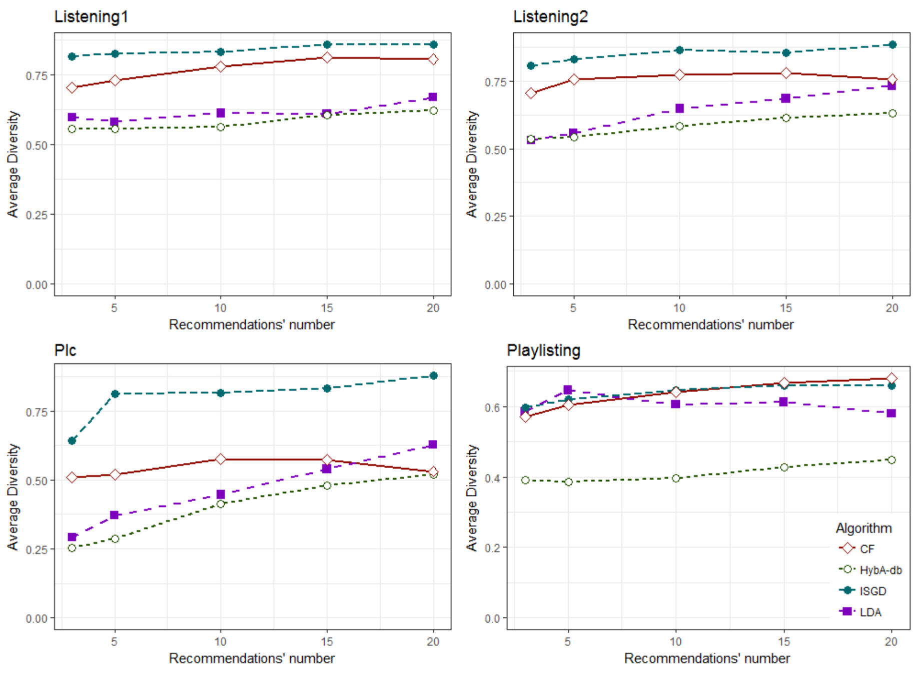


Figure 5.16: Recommendations' diversity for the various techniques

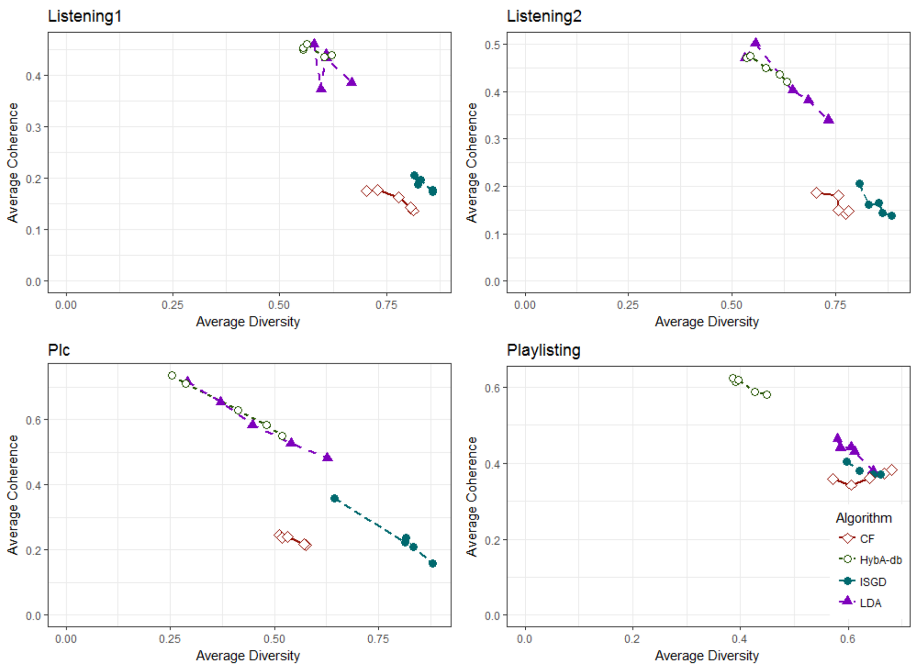


Figure 5.17: Average coherence in relation to average diversity of the various recommended playlist continuations

From figures 5.15 and 5.16, where average coherence and diversity for the recommended playlists of different lengths are shown, we see that, again, depending on the algorithm's emphasis and reasoning, there is a clear difference between the levels achieved. Among the tested algorithms, HybA-db recommends the most coherent playlists. Although being less diverse than those by the other algorithms, those playlists continuations are the ones showing the best "trade-off" between relevance, coherence and diversity. This is due to the common emphasis placed on both coherence and diversity during the recommendations' generation phase, as explained in sections 4.9.2 and 4.9.3.

In figure 5.17, the joint evolution of the diversity and coherence in the generated recommendations is presented. HybA and LDA manage to recommend lists characterised by good levels of both coherence and diversity while the user-based approaches generate less coherent playlist continuations. Shortest plots reveal a more stable behaviour, as the characteristics of the generated playlists are less affected by the length of the recommended playlist.

Additionally, the playlist continuations recommended by HybA, have diversity levels similar to the ones in the initial databases, as the algorithm's aim is to maintain the diversity level of a started list, instead of maximising it. Furthermore, although being more diverse, as they follow the diversity tendencies in user profiles, the recommendations by the user-based algorithms do not manage to combine diversity with accuracy. Their accuracy levels are significantly lower than the ones achieved by HybA-db, and their combined performance related to precision and diversity, as shown in figure 5.19, is also lower.

Finally, the F-measure and the F_d-measure for recommendations of playlist continuations of 10 items, are presented.

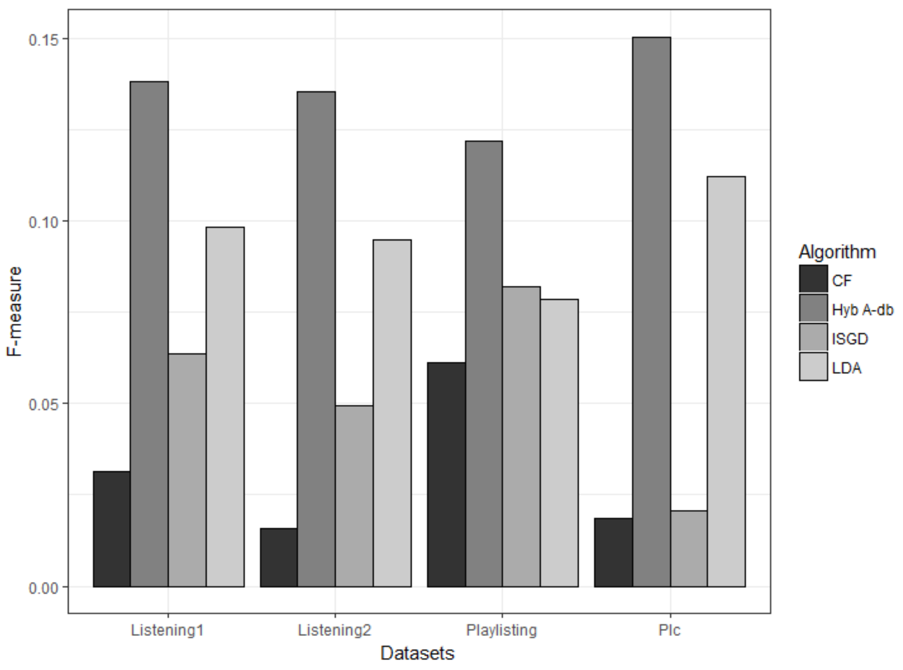


Figure 5.18: F-measure for recommendations of 10 songs

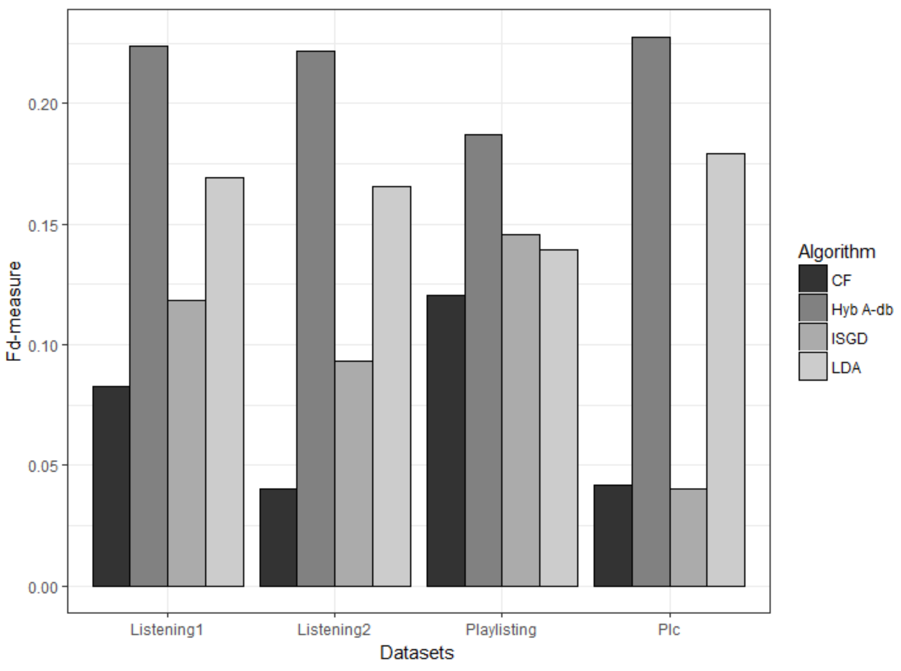


Figure 5.19: F_d-measure for recommendations of 10 songs

Thus, HybA manages to have a good performance when evaluating jointly accuracy dimensions or additional beyond-accuracy dimensions, like diversity and overperforms the other methods.

Recall@N

Finally, although our focus has been set mainly on precision when "hiding" and recommending the same number of items for every playlist, for the sake of completeness, in this section, we present also the results for recall@N. When evaluating next item recommendations for playlist continuations, or in top-N recommendation tasks, this factor has been also used to evaluate the ability of the different algorithms to identify relevant items (Cremonesi et al., 2010).

Therefore, we use a slightly different experimental setting. From the lists in the test part, we hide each time one item (the last one) and generate recommendations for playlist continuations of different lengths. Each time the ability of the recommendation techniques to correctly identify the hidden item is evaluated, that would be equal to hit ratio, the percentage of times that a user is recommended at least one relevant item. In figure 5.20, below, recall@N for different values of N, for the compared algorithms for tests performed on the four datasets is presented.

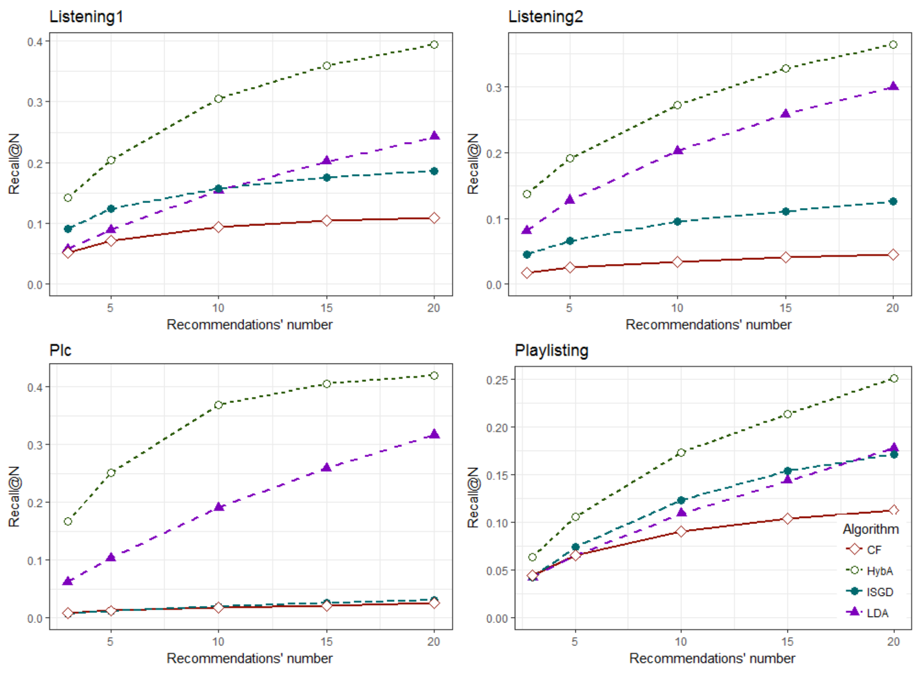


Figure 5.20: Recommendations' recall@N for the various techniques

Once again, the recommendation techniques that focus on entire "sessions" perform better than the ones that focus on users. HybA provides significantly higher recall values.

5.5 Conclusions

In this chapter, the experimentation results that have been reported, during the gradual implementations and testing of the several versions of HybA, have been presented. A relatively small music dataset containing songs with well-defined metadata, has been used for the evaluation of the initial versions of the RS. Next, four large scale real datasets have been used for the evaluation of the final version of the algorithm to ensure that it is able to efficiently address the principal research questions stated in chapter 1.

Through the evolution of the designed solution, its extension was made in order to handle more dimensions, and overcome the usual limitations of playlist recommenders. Furthermore, as the system's purpose has been gradually extended, the scope of the final evaluation part (section 5.4) has been two-fold. In general, the results presented in this section show that the proposed term of *playlist concept*, implicitly capturing the contextual dimensions of playlists, constructed under given circumstances, based on their music style combinations, better identifies playlist similarities, compared to explicit contextual dimensions. HybA has been found to perform better than the compared techniques, and in addition, to be able to have a good performance in terms of accuracy, while taking into account more *cognitive dimensions* of the playlist recommendation problem. In addition, our initial hypothesis that a hybrid approach, designed to address the characteristics of the specific problem, would be able to perform better than the usual recommendation techniques has been confirmed.

Finally, additional evaluation metrics have been described and used to provide more insight on the relevance and the quality of the recommended playlist continuations. The purpose was to provide an additional way of evaluating playlist continuation recommendations, leading to an improved performance in more dimensions.

Part IV

Conclusions and Future Work

Chapter 6

Conclusions

The majority of MRSs, and RSs in general, still treat the recommendation problem at two dimensions, namely users and items, while aiming to maximize the accuracy of their predicted behaviour. However, especially when it comes to more complicated domains, like music, the specific characteristics of the treated items, namely music items, have to be taken into account.

Music items are rarely consumed in isolation, but rather as well organized sequences designed for a specific purpose or aiming to transmit a desired message. In addition, due to their complicated nature, that both creates and is subject to emotional and cultural parameters, there is a necessity to identify user preferences under specific concepts. Furthermore, apart from personal user preferences towards songs, the designed collections must balance between the user's desire for repetition and surprise, thus being coherent and of some diversity degree. Finally, as the aim of these systems is to deliver recommendations of improved quality, additional focus should placed on attributes *beyond accuracy.*

In this thesis, we have presented several recommender systems for automatic playlist continuation aiming to address the above issues. Especially HybA, a hybrid recommendation approach that aims to generate recommendations for playlist continuations of improved quality, related to given cognitive concepts. More than evaluating the user who constructs a playlist, each time the current concept has to be extracted from the started playlist. Based on the defined scope and the characteristics of the proposed approach, in chapter 1 the following research questions had been stated. For convenience, we repeat those below along with the contribution and conclusions arising from each of them.

6.1 Contributions

1. *Are the currently used recommendation algorithms able to efficiently handle automatic playlist continuation, and item collection, recommendations? Does the proposed algorithm manage to address the needs of the specific problem in a better way?*

   Although there is the necessity of treating, and evaluating, item collections as a whole in many application domains, like in MBA, APG and APC, the mainly used recommendation techniques still lack of support towards this direction. Even when presenting a list of items to the user, these usually have not been designed to work well together, and rather form the different alternatives for a given query. Therefore, when used together, those fail to address more complicated user needs.

Furthermore, even when designed to perform well together, as a package like in APG, in APC, it is of high importance to correctly identify the purpose of a started playlist. Usually, this goes beyond the classical user preference estimation problem, as the even same user may construct different playlists under different circumstances.

Our initial hypothesis, that a solution designed to address the needs of this specific problem would perform better than an adaptation of the usual techniques, has been confirmed. As shown by the experimental results in section 5.4.3, "session-based" approaches with focus on the entire playlists perform better than the user-based ones. The recommendation results in general were of higher accuracy, while presenting also more coherent playlist continuations.

The proposed algorithm, combines LDA and CBR for the playlists' modelling, in order to capture their underlying semantic concepts and calculate their similarities. It has been found to over-perform the other commonly used recommendation techniques, in terms of accuracy, while enhancing more parameters related to user listening experience, like playlist coherence.

2. *How does the use of additional parameters, related to the quality characteristics and user perception of the sets of music items, affect the recommendation results?*

   In general, support to some of the beyond accuracy dimensions, like diversity, in many cases has been found to negatively affect recommendation accuracy. Furthermore, depending on the purpose of the applied recommendation algorithm, usually a clear emphasis is placed on some of the parameters that can be detected in the corresponding results. As it can be seen in the results presented in the previous chapter, in section 5.4.3, the majority of the algorithms used, have clear tendencies, coming from their reasoning process, and clearly boost one of those dimensions. Especially user-based techniques, due to their focus on the entire user histories and the preferences of similar users, were find to generate very diverse playlists. This increased diversity, that overtook the average diversity tendencies observed in the test datasets, had a negative effect on their recommendation accuracy.

   However, as it has been shown, it is possible to achieve balanced solutions that perform well from both accuracy and quality perspectives. Especially support to coherence, if correctly captured, may result in accuracy improvements. Furthermore, when adequately combined, like in the case of HybA, both a good level of coherence and diversity in a playlist can be achieved. Thus, an acceptable solution in terms of quality may be provided, without decreasing the results' relevance.

3. *Are the commonly used evaluation metrics suitable when recommending item collections? Which other factors should be evaluated?*

The commonly used IR evaluation metrics focus only on the accuracy of recommendations without providing any information on the collections' quality. Furthermore, these techniques are "too strict" to be applied to application domains, characterised by contextual and sentimental dimensions, as they only evaluate the presence, or absence, and sometimes the position, of a specific item within a set.

In actual application domains, due to the existing data sparsity and the popularity bias, it is more important to capture "how close" a recommendation is to the user's expectations, in terms of item characteristics. In the case of collection recommendations, this would be mapped to collection characteristics. This is the reason why we proposed the use of the average similarity between playlists combined with the achieved hit ratio. The first, to evaluate the ability of a system to identify interesting alternatives that, as a whole, are close to the user's expectations, while the other captures the general accuracy level achieved.

In addition, when recommending sets of items, apart from their relevance, their quality should be evaluated. When referring to a playlist, to be considered as a nice result, it should be coherent, as the smoothness of song transitions influences its effect on listeners. However, due to the fundamental goals behind music consumption, it should also be of some diversity degree to generate some excitement to the listener. Therefore, the levels of these parameters should be also evaluated. However, as the perception of these dimensions, especially diversity, is highly subjective, being subject to user actual preferences, and possibly additional domain specific parameters, their exact values cannot be set a priori. Regarding APC, the levels of these parameters depend on the corresponding ones in the started playlist, because a playlist continuation, apart from its quality, has to fit well within the started playlist.

In general, coherent playlists are thought as better. Thus, the coherence level could serve as a metric of quality. In addition, coherence has been found to possibly have a positive relation with recommendations' accuracy, while it could be also combined with long tail presence.

6.1.1 Limitations

Although the described approach has been found to outperform the used as baseline recommendation techniques, it still comes with some limitations that may affect its performance.

First of all, due to its reasoning model, this approach highly depends on the data used for its training and reasoning processes, both at song and playlist level. If the metadata related to songs is of high quality, the resulting song clusters and music styles, will be informative enough, and could be used to capture the tendencies observed in playlists. In contrast, if this information is sparse and noisy, the resulting song clusters and the computed similarity degrees will probably fail to reflect the real similarity levels. On the other hand, the number and the quality of the playlists in the case base, has an influence on the resulting conceptual clusters. These clusters should reflect the existing music tendencies in order to effectively address future user needs. This is due to the fact that, when a new playlist continuation is required, first the concept of the started playlist is identified. Then, the playlists of the same conceptual cluster are retrieved, and form the basis for further computations, that lead to the recommendations' generation.

In relation with the computational cost, this is high mainly during the offline phase of the system, as the pairwise similarities of the existing items styles have to be calculated. Therefore, the corresponding song clusters are found, and then the playlist latent topics and resulting concepts are analysed. When new items are added to the system, if they share the characteristics of an existing song cluster no additional computations are required. Otherwise, its similarities with all the existing music styles have to be identified.

On the other hand, although appearing able to improve the recommendations quality, it is difficult to capture the "real" user's perception of playlist quality without having performed experiments with real users. In general, there are many factors that impact music consumption, and are not being recorded, in the majority of music datasets, in a way that it could support the design on more user perception oriented methods. However, this has an impact on the majority of current MRSs.

6.2 Future Work

The limitations identified in the previous section form also the basic topics of our ongoing and scheduled future work, towards the improvement of the developed RS. In addition, some other functionalities are planned to be evaluated and possibly incorporated in order to extend the system.

When treating music recommendations, an important factor is the possible degree of discovery and excitement, that may be caused to a user by presenting yet unknown but relevant items. Therefore, support to novelty is among the principal parameters that will be analysed in depth, in order to identify an adequate way of incorporating it to the model (Celma & Herrera, 2008). In contrast to coherence and diversity, novelty could have a positive relation with both coherence and diversity (Vargas & Castells, 2011). Furthermore, support to novelty could be one way of increasing the diversity of the recommended playlist continuation (M. Zhang & Hurley, 2008).

The relation between coherence and diversity will be further monitored in order to investigate possible ways of achieving an optimal "equilibrium" among them. As these factors are related with the user perception of quality, of the recommended playlist, identifying a standard "point" among them could be used also for evaluation purposes. Finally, the design of real user experiments, in order to capture users' real perception of quality, and the attributes that mostly influence them, has been defined among the important future steps. At the moment, the system focuses on the three first steps of the CBR cycle. Having users' feedback on the generated recommendations could support the implementation of the forth, the retain, step.

List of Related Publications

Gatzioura, A., & Sànchez-Marrè, M. (2015). A Case-Based Recommendation Approach for Market Basket Data. *IEEE Intelligent Systems*, *30*(1), 20–27.

Gatzioura, A., & Sànchez-Marrè, M. (2017a). A Case-Based Reasoning Framework for Music Playlist Recommendations. In *4th IEEE International Conference on Control, Decision and Information Technologies, CoDIT 2017, Barcelona, Spain, April 5-7, 2017* (pp. 242–247).

Gatzioura, A., & Sànchez-Marrè, M. (2017b). Using Contextual Information in Music Playlist Recommendations. In *Recent Advances in Artificial Intelligence Research and Development - Proceedings of the 20th International Conference of the Catalan Association for Artificial Intelligence (CCIA 2017), Deltebre, Terres de l'Ebre, Spain, October 25-27, 2017* (pp. 239–244).

Gatzioura, A., Sànchez-Marrè, M., & Jorge, A. M. (2018). A Study on Contextual Influences on Automatic Playlist Continuation. In *Proceedings of the 21th International Conference of the Catalan Association for Artificial Intelligence (CCIA 2018), Roses, Spain, October 8-10, 2018.*

Gatzioura, A., Jorge, A. M., Sànchez-Marrè, M., & Vinagre, J. (2018). A Hybrid Recommendation Algorithm for Improving Automatic Playlist Continuation. *Submitted to: Cognitive Computation, Special Issue on : Bridging Cognitive Models and Recommender Systems*.

References

Aamodt, A., & Plaza, E. (1994). Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI communications*, *7*(1), 39–59.

Adamopoulos, P., & Tuzhilin, A. (2014). On unexpectedness in recommender systems: Or how to better expect the unexpected. *ACM TIST*, *5*(4), 54:1–54:32.

Adomavicius, G., & Kwon, Y. (2012). Improving aggregate recommendation diversity using ranking-based techniques. *IEEE Trans. Knowl. Data Eng.*, *24*(5), 896–911.

Adomavicius, G., Manouselis, N., & Kwon, Y. (2011). Multi-criteria recommender systems. In *Recommender systems handbook* (pp. 769–803). Springer.

Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, *17*(6), 734–749.

Adomavicius, G., & Tuzhilin, A. (2011). Context-aware recommender systems. In *Recommender systems handbook* (pp. 217–253). Springer.

Agrawal, R., Imieliński, T., & Swami, A. (1993). Mining association rules between sets of items in large databases. In *Acm sigmod record* (Vol. 22, pp. 207–216).

Amatriain, X., Jaimes, A., Oliver, N., & Pujol, J. M. (2011). Data mining methods for recommender systems. In *Recommender systems handbook* (pp. 39–71).

Andric, A., & Haus, G. (2006). Automatic playlist generation based on tracking user's listening habits. *Multimedia Tools Appl.*, *29*(2), 127–151.

Baccigalupo, C., & Plaza, E. (2006). Case-based sequential ordering of songs for playlist recommendation. *Advances in Case-Based Reasoning*, 286–300.

Balkwill, L.-L., & Thompson, W. F. (1999). A cross-cultural investigation of the perception of emotion in music: Psychophysical and cultural cues. , *17*(1), 43–64. doi: 10.2307/40285811

Bell, R., Koren, Y., & Volinsky, C. (2009, 08). Matrix factorization techniques for recommender systems. *Computer*, *42*, 30-37.

Bergmann, R. (1991, 01). An introduction to case-based reasoning. , *6*.

Blei, D. (2012, 4). Probabilistic topic models. , *55*, 77-84.

Blei, D., Ng, A., & Jordan, M. (2003, 3). Latent dirichlet allocation. , *3*, 993-1022.

Bogdanov, D., Haro, M., Fuhrmann, F., Gómez, E., & Herrera, P. (2010). Content-based music recommendation based on user preference examples. In *Womrad 2010 workshop on music recommendation and discovery* (p. 6).

Bogdanov, D., & Herrera, P. (2011, 01). How much metadata do we need in music recommendation? a subjective evaluation using preference sets. In (p. 97-102).

Bollen, D., Knijnenburg, B. P., Willemsen, M. C., & Graus, M. (2010). Understanding choice overload in recommender systems. In *Proceedings of the fourth ACM conference on recommender systems - RecSys ́10.* ACM Press.

Bonnin, G., & Jannach, D. (2013). A comparison of playlist generation strategies for music recommendation and a new baseline scheme. In *Workshops at the twenty-seventh aaai conference on artificial intelligence.*

Bonnin, G., & Jannach, D. (2014). Automated generation of music playlists: Survey and experiments. *ACM Comput. Surv.*, *47*(2), 26:1–26:35.

Borràs, J., Moreno, A., & Valls, A. (2017, Nov 01). Diversification of recommendations through semantic clustering. *Multimedia Tools and Applications*, *76*(22), 24165–24201.

Braunhofer, M., Kaminskas, M., & Ricci, F. (2013). Location-aware music recommendation. *IJMIR*, *2*(1), 31–44.

Bridge, D. G., Göker, M. H., McGinty, L., & Smyth, B. (2005). Case-based recommender systems. *Knowledge Eng. Review*, *20*(3), 315–320.

Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, *12*(4), 331–370.

Burke, R. D. (2004). Hybrid recommender systems with case-based components. In *Advances in case-based reasoning, 7th european conference, ECCBR 2004, madrid, spain, august 30 - september 2, 2004, proceedings* (pp. 91–105).

Byrd, D., & Crawford, T. (2002, March). Problems of music information retrieval in the real world. *Inf. Process. Manage.*, *38*(2), 249–272.

Campos, R., Dias, G., Jorge, A. M., & Nunes, C. (2017). Identifying top relevant dates for implicit time sensitive queries. *Inf. Retr. Journal*, *20*(4), 363–398.

Casey, M. A., Veltkamp, R. C., Goto, M., Leman, M., Rhodes, C., & Slaney, M. (2008). Content-based music information retrieval: Current directions and future challenges. *Proceedings of the IEEE*, *96*(4), 668–696.

Castells, P., Hurley, N. J., & Vargas, S. (2015). Novelty and diversity in recommender systems. In *Recommender systems handbook* (pp. 881–918).

Castells, P., Wang, J., Lara, R., & Zhang, D. (2011). Workshop on novelty and diversity in recommender systems-divers 2011. In *Proceedings of the fifth acm conference on recommender systems* (pp. 393–394).

Cavique, L. (2007). A scalable algorithm for the market basket analysis. *Journal of Retailing and Consumer Services*, *14*(6), 400 - 407. (Data Mining Applications in Retailing and Consumer Services)

Celma, Ò. (2010). *Music recommendation and discovery.* Springer Berlin Heidelberg.

Celma, Ò., & Cano, P. (2008). From hits to niches? In *Proceedings of the 2nd KDD workshop on large-scale recommender systems and the netflix prize competition - NETFLIX ́08.* ACM Press.

Celma, Ò., & Herrera, P. (2008). A new approach to evaluating novel recommendations. In *Proceedings of the 2008 ACM conference on recommender* *systems - RecSys ́08.* ACM Press.

Celma, Ò., Herrera, P., & Serra, X. (2006). Bridging the music semantic gap.

Celma, Ò., & Lamere, P. (2011, oct). If you like radiohead, you might like this article. *AI Magazine*, *32*(3), 57.

Chen, Y.-L., Tang, K., Shen, R.-J., & Hu, Y.-H. (2005). Market basket analysis in a multiple store environment. *Decision Support Systems*, *40*(2), 339 - 354.

Christidis, K., Apostolou, D., & Mentzas, G. (2010). *Exploring customer preferences with probabilistic topics models.*

Cremonesi, P. (2009). Top-n recommendations on unpopular items with contextual knowledge..

Cremonesi, P., Koren, Y., & Turrin, R. (2010). Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth acm conference on recommender systems* (pp. 39–46). New York, NY, USA: ACM.

Cunningham, S. J., Bainbridge, D., & Falconer, A. (2006). 'more of an art than a science': Supporting the creation of playlists and mixes. In *ISMIR 2006, 7th international conference on music information retrieval, victoria, canada, 8-12 october 2006, proceedings* (pp. 240–245).

Deshpande, M., & Karypis, G. (2004, jan). Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, *22*(1), 143–177.

desJardins, M., Eaton, E., & Wagstaff, K. L. (2006). Learning user preferences for sets of objects. In *Proceedings of the 23rd international conference on machine learning* (pp. 273–280). New York, NY, USA: ACM.

Dey, A. K., Salber, D., Abowd, G. D., & Futakawa, M. (2000). *Providing architectural support for context-aware applications* (Tech. Rep.).

Domingues, M. A., Gouyon, F., Jorge, A. M., Leal, J. P., Vinagre, J., Lemos, L., & Sordo, M. (2013, Mar 01). Combining usage and content in an online recommendation system for music in the long tail. *International Journal of Multimedia Information Retrieval*, *2*(1), 3–13.

Domingues, M. A., Jorge, A. M., & Soares, C. (2011). Using contextual information as virtual items on top-n recommender systems. *CoRR*, *abs/1111.2948*.

Domingues, M. A., & Rezende, S. O. (2013, Oct). The impact of context-aware recommender systems on music in the long tail. In *2013 brazilian conference on intelligent systems* (p. 119-124).

Ferwerda, B., & Schedl, M. (2014). Enhancing music recommender systems with personality information and emotional states: A proposal. In *Umap workshops.*

Ferwerda, B., Schedl, M., & Tkalcic, M. (2015). Personality & emotional states: Understanding users' music listening needs. In *Posters, demos, late-breaking results and workshop proceedings of the 23rd conference on user modeling, adaptation, and personalization (UMAP 2015), dublin, ireland, june 29 - july 3, 2015.*

Ferwerda, B., Tkalcic, M., & Schedl, M. (2017). Personality traits and music genres: What do people prefer to listen to? In *Proceedings of the 25th conference on user modeling, adaptation and personalization, UMAP 2017, bratislava, slovakia, july 09 - 12, 2017* (pp. 285–288).

Ferwerda, B., Vall, A., Tkalcic, M., & Schedl, M. (2016). Exploring music diversity needs across countries. In *Proceedings of the 2016 conference on* *user modeling adaptation and personalization* (pp. 287–288). New York, NY, USA: ACM.

Ferwerda, B., Yang, E., Schedl, M., & Tkalcic, M. (2015). Personality traits predict music taxonomy preferences. In *Proceedings of the 33rd annual acm conference extended abstracts on human factors in computing systems* (pp. 2241–2246). New York, NY, USA: ACM.

Finnie, G. R., & Sun, Z. (2002). Similarity and metrics in case-based reasoning. *Int. J. Intell. Syst.*, *17*(3), 273–287.

Flexer, A., Schnitzer, D., Gasser, M., & Widmer, G. (2008). Playlist generation using start and end songs. In *ISMIR 2008, 9th international conference on music information retrieval, drexel university, philadelphia, pa, usa, september 14-18, 2008* (pp. 173–178).

Gatzioura, A. (2013, 06). *Design and implementation of a customer personalized recommender system.*

Gatzioura, A., Jorge, A. M., Sànchez-Marrè, M., & Vinagre, J. (2018). A hybrid recommendation algorithm for improving automatic playlist continuation. *Cognitive Computation, Special Issue on: Bridging Cognitive Models and Recommender Systems*.

Gatzioura, A., & Sànchez-Marrè, M. (2015). A case-based recommendation approach for market basket data. *IEEE Intelligent Systems*, *30*(1), 20–27.

Gatzioura, A., & Sànchez-Marrè, M. (2017a). A case-based reasoning framework for music playlist recommendations. In *4th ieee international conference on control, decision and information technologies, codit 2017, barcelona, spain, april 5-7, 2017* (pp. 242–247).

Gatzioura, A., & Sànchez-Marrè, M. (2017b). Using contextual information in music playlist recommendations. In *Recent advances in artificial intelligence research and development - proceedings of the 20th international conference of the catalan association for artificial intelligence (ccia 2017), deltebre, terres de l'ebre, spain, october 25-27, 2017* (pp. 239–244).

Gatzioura, A., Sànchez-Marrè, M., & Jorge, A. M. (2018). A study on contextual influences on automatic playlist continuation. In *Proceedings of the 21th international conference of the catalan association for artificial intelligence (ccia 2018), roses, spain, october 8-10, 2018.*

Ge, M., Delgado-Battenfeld, C., & Jannach, D. (2010). Beyond accuracy. In *Proceedings of the fourth ACM conference on recommender systems - RecSys 10.* ACM Press.

Gillhofer, M., & Schedl, M. (2015). Iron maiden while jogging, debussy for dinner? In X. He, S. Luo, D. Tao, C. Xu, J. Yang, & M. A. Hasan (Eds.), *Multimedia modeling* (pp. 380–391). Cham: Springer International Publishing.

Golbeck, J., & Hansen, D. L. (2011). A framework for recommending collections. In *Proceedings of the workshop on novelty and diversity in recommender systems, divers 2011, at the 5^{th} ACM international conference on recommender systems, recsys 2011, chicago, illinois, usa, 23 october 2011* (pp. 35–42).

Goldberg, D., Nichols, D. A., Oki, B. M., & Terry, D. B. (1992). Using collaborative filtering to weave an information tapestry. *Commun. ACM*, *35*(12), 61–70.

Guerraoui, R., Merrer, E. L., Patra, R., & Vigouroux, J. (2017). Sequences, items and latent links: Recommendation with consumed item packs. *CoRR*, *abs/1711.06100*.

Hansen, D. L., & Golbeck, J. (2009). Mixing it up: recommending collections of items. In *Proceedings of the 27th international conference on human factors in computing systems, CHI 2009, boston, ma, usa, april 4-9, 2009* (pp. 1217–1226).

Haruechaiyasak, C., & Damrongrat, C. (2008). Article recommendation based on a topic model for wikipedia selection for schools. In *Digital libraries: Universal and ubiquitous access to information, 11th international conference on asian digital libraries, ICADL 2008, bali, indonesia, december 2-5, 2008. proceedings* (pp. 339–342).

He, X., Zhang, H., Kan, M.-Y., & Chua, T.-S. (2016). Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of the 39th international acm sigir conference on research and development in information retrieval* (pp. 549–558). New York, NY, USA: ACM.

Herlocker, J., Konstan, J., Terveen, L., & Riedl, J. (2004). Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, *22*(1), 5–53.

Hofmann, T. (1999). Probabilistic latent semantic analysis. In *Proceedings of the fifteenth conference on uncertainty in artificial intelligence* (pp. 289–296). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Hofmann, T. (2001, Jan). Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, *42*(1), 177–196.

Huang, S. (2011). Designing utility-based recommender systems for e-commerce: Evaluation of preference-elicitation methods. *Electronic Commerce Research and Applications*, *10*(4), 398–407.

Interdonato, R., Romeo, S., Tagarelli, A., & Karypis, G. (2013). A versatile graph-based approach to package recommendation. In *2013 IEEE 25th international conference on tools with artificial intelligence, herndon, va, usa, november 4-6, 2013* (pp. 857–864).

Iwata, T., & Sawada, H. (2013). Topic model for analyzing purchase data with price information. *Data Min. Knowl. Discov.*, *26*(3), 559–573.

Iwata, T., Watanabe, S., Yamada, T., & Ueda, N. (2009). Topic tracking model for analyzing consumer purchase behavior. In *IJCAI 2009, proceedings of the 21st international joint conference on artificial intelligence, pasadena, california, usa, july 11-17, 2009* (pp. 1427–1432).

Jannach, D., Lerche, L., & Kamehkhosh, I. (2015). Beyond "hitting the hits": Generating coherent music playlist continuations with the right tracks. In *Proceedings of the 9th acm conference on recommender systems* (pp. 187–194). New York, NY, USA: ACM.

Jannach, D., Zanker, M., Felfernig, A., & Friedrich, G. (2010). *Recommender systems: An introduction.* Cambridge Univ Pr.

Kamehkhosh, I., & Jannach, D. (2017). User perception of next-track music recommendations. In *Proceedings of the 25th conference on user modeling, adaptation and personalization* (pp. 113–121). New York, NY, USA: ACM.

Kaminskas, M., Fernández-Tobías, I., Ricci, F., & Cantador, I. (2012). Knowledge-based music retrieval for places of interest. In *Proceedings of the second international ACM workshop on music information retrieval* *with user-centered and multimodal strategies, MIRUM '12, nara, japan, october 29 - november 02, 2012* (pp. 19–24).

Kaminskas, M., & Ricci, F. (2012, may). Contextual music information retrieval and recommendation: State of the art and challenges. *Computer Science Review*, *6*(2-3), 89–119.

Kaminskas, M., & Ricci, F. (2017). Emotion-based matching of music to places. In *Emotions and personality in personalized services - models, evaluation and applications* (pp. 287–310).

Kim, Y. E., Schmidt, E. M., Migneco, R., Morton, O. G., Richardson, P., Scott, J., ... Turnbull, D. (2010). Emotion recognition: a state of the art review. In *11th international society for music information and retrieval conference.*

Knees, P., & Schedl, M. (2013, dec). A survey of music similarity and recommendation from music context data. *ACM Transactions on Multimedia Computing, Communications, and Applications*, *10*(1), 1–21.

Kolodner, J. L. (1992). An introduction to case-based reasoning. *Artif. Intell. Rev.*, *6*(1), 3–34.

Konstan, J. A., Miller, B. N., Maltz, D., Herlocker, J. L., Gordon, L. R., & Riedl, J. (1997). Grouplens: Applying collaborative filtering to usenet news. *Commun. ACM*, *40*(3), 77–87.

Konstan, J. A., & Riedl, J. (2012, Apr 01). Recommender systems: from algorithms to user experience. *User Modeling and User-Adapted Interaction*, *22*(1), 101–123.

Krestel, R., Fankhauser, P., & Nejdl, W. (2009). Latent dirichlet allocation for tag recommendation. In *Proceedings of the 2009 ACM conference on recommender systems, recsys 2009, new york, ny, usa, october 23-25, 2009* (pp. 61–68).

Lamere, P. (2008, jun). Social tagging and music information retrieval. *Journal of New Music Research*, *37*(2), 101–114.

Landauer, T. K., Foltz, P. W., & Laham, D. (1998). An introduction to latent semantic analysis. *Discourse Processes*, *25*(2-3), 259-284.

Leake, D. B. (1996). Cbr in context : The present and future..

Lee, K. (2014). Using dynamically promoted experts for music recommendation.

Lee, K., & Lee, K. (2011). My head is your tail: applying link analysis on long-tailed music listening behavior for music recommendation. In *Proceedings of the 2011 ACM conference on recommender systems, recsys 2011, chicago, il, usa, october 23-27, 2011* (pp. 213–220).

Levy, M., & Bosteels, K. (2010). Music recommendation and the long tail. In *Womrad 2010 workshop on music recommendation and discovery.*

Liao, T. W., Zhang, Z., & Mount, C. (1998). Similarity measures for retrieval in case-based reasoning systems. *Applied Artificial Intelligence*, *12*(4), 267–288.

Linden, G., Smith, B., & York, J. (2003). Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, *7*(1), 76–80.

Liu, N. H., & Hsieh, S. J. (2009). Intelligent music playlist recommendation based on user daily behavior and music content. *Advances in Multimedia Information ProcessingPCM 2009*, 671–683.

Logan, B. (2004). Music recommendation from song sets. In *ISMIR 2004, 5th international conference on music information retrieval, barcelona, spain,* *october 10-14, 2004, proceedings.*

López de Mántaras, R. (2001). Case-based reasoning. In *Machine learning and its applications, advanced lectures* (pp. 127–145).

Lopez De Mantaras, R., McSherry, D., Bridge, D., Leake, D., Smyth, B., Craw, S., . . . others (2005). Retrieval, reuse, revision and retention in case-based reasoning. *The Knowledge Engineering Review*, *20*(03), 215–240.

Lops, P., de Gemmis, M., & Semeraro, G. (2010, oct). Content-based recommender systems: State of the art and trends. In *Recommender systems handbook* (pp. 73–105). Springer US.

Lorenzi, F., & Ricci, F. (2005). Case-based recommender systems: A unifying view. In *Lecture notes in computer science* (pp. 89–113). Springer Berlin Heidelberg.

Maillet, F., Eck, D., Desjardins, G., & Lamere, P. (2009). Steerable playlist generation by learning song similarity from radio station playlists. In *Proceedings of the 10th international society for music information retrieval conference, ISMIR 2009, kobe international conference center, kobe, japan, october 26-30, 2009* (pp. 345–350).

McFee, B., & Lanckriet, G. R. G. (2011). The natural language of playlists. In *Proceedings of the 12th international society for music information retrieval conference, ISMIR 2011, miami, florida, usa, october 24-28, 2011* (pp. 537–542).

Melville, P., & Sindhwani, V. (2010). Recommender systems. In *Encyclopedia of machine learning* (pp. 829–838).

Mild, A., & Reutterer, T. (2002). An improved collaborative filtering approach for predicting cross-category purchases based on binary market basket data..

Morrison, S. J., & Demorest, S. M. (2009). Cultural constraints on music perception and cognition. In J. Y. Chiao (Ed.), *Cultural neuroscience: Cultural influences on brain function* (Vol. 178, p. 67 - 77). Elsevier.

Murakami, T., Mori, K., & Orihara, R. (2008). Metrics for evaluating the serendipity of recommendation lists. In K. Satoh, A. Inokuchi, K. Nagao, & T. Kawamura (Eds.), *New frontiers in artificial intelligence* (pp. 40–46). Berlin, Heidelberg: Springer Berlin Heidelberg.

Pachet, F., & Roy, P. (1999). Automatic generation of music programs. In *Principles and practice of constraint programming - cp'99, 5th international conference, alexandria, virginia, usa, october 11-14, 1999, proceedings* (pp. 331–345).

Park, Y.-J., & Tuzhilin, A. (2008). The long tail of recommender systems and how to leverage it. In *Proceedings of the 2008 ACM conference on recommender systems - RecSys ́08.* ACM Press.

Pichl, M., Zangerle, E., & Specht, G. (2015). Towards a context-aware music recommendation approach: What is hidden in the playlist name? In *IEEE international conference on data mining workshop, ICDMW 2015, atlantic city, nj, usa, november 14-17, 2015* (pp. 1360–1365).

Prasad, B. (2003). Intelligent techniques for e-commerce. *J. Electron. Commerce Res.*, *4*(2), 65–71.

Ragno, R., Burges, C. J. C., & Herley, C. (2005). Inferring similarity between music objects with application to playlist generation. *Proceedings of the 7th ACM SIGMM international workshop on Multimedia information retrieval MIR 05*, 73–80.

Rentfrow, P., & Gosling, S. (2003, 07). The do re mi's of everyday life: The structure and personality correlates of music preferences. , *84*, 1236-56.

Resnick, P., Iacovou, N., Sushak, M., & Bergstrom, P. (1994). Grouplens – an open architecture for collaborative filtering of netnews. In *Proceedings of the computer supported collaborative work conference (acm conference).*

Ricci, F. (2010). Mobile recommender systems. *J. of IT & Tourism*, *12*(3), 205–231.

Ricci, F., Arslan, B., Mirzadeh, N., & Venturini, A. (2002). ITR: A case-based travel advisory system. In *Advances in case-based reasoning, 6th european conference, ECCBR 2002 aberdeen, scotland, uk, september 4-7, 2002, proceedings* (pp. 613–627).

Ricci, F., Cavada, D., Mirzadeh, N., & Venturini, A. (2006). Case-based travel recommendations. *Destination Recommendation Systems: Behavioural Foundations and Applications*, 67–93.

Ricci, F., Rokach, L., & Shapira, B. (2010, oct). Introduction to recommender systems handbook. In *Recommender systems handbook* (pp. 1–35). Springer US.

Ricci, F., & Werthner, H. (2001, 01). Case base querying for travel planning recommendation. , *4*, 215-226.

Richter, M., & Weber, R. (2013). *Case-based reasoning: a textbook.*

Sánchez, D., Batet, M., Isern, D., & Valls, A. (2012). Ontology-based semantic similarity: A new feature-based approach. *Expert Syst. Appl.*, *39*(9), 7718–7728.

Schafer, J., Konstan, J., & Riedl, J. (2001). E-commerce recommendation applications. *Data mining and knowledge discovery*, *5*(1), 115–153.

Schedl, M. (2016). The lfm-1b dataset for music retrieval and recommendation. In *Icmr.*

Schedl, M. (2017, Mar 01). Investigating country-specific music preferences and music recommendation algorithms with the lfm-1b dataset. *International Journal of Multimedia Information Retrieval*, *6*(1), 71–84.

Schedl, M., Knees, P., & Gouyon, F. (2017). New paths in music recommender systems research. In *Proceedings of the eleventh ACM conference on recommender systems, recsys 2017, como, italy, august 27-31, 2017* (pp. 392–393).

Schedl, M., Knees, P., McFee, B., Bogdanov, D., & Kaminskas, M. (2015). Music recommender systems. In *Recommender systems handbook* (pp. 453–492).

Schedl, M., Yang, Y., & Herrera-Boyer, P. (2017). Introduction to intelligent music systems and applications. *ACM TIST*, *8*(2), 17:1–17:8.

Schedl, M., Zamani, H., Chen, C., Deldjoo, Y., & Elahi, M. (2017). Current challenges and visions in music recommender systems research. *CoRR*, *abs/1710.03208*.

Schilit, B., Adams, N., & Want, R. (1994, Dec). Context-aware computing applications. In *1994 first workshop on mobile computing systems and applications* (p. 85-90).

Shao, B., Ogihara, M., Wang, D., & Li, T. (2009). Music recommendation based on acoustic features and user access patterns. *IEEE Trans. Audio, Speech & Language Processing*, *17*(8), 1602–1611.

Shi, L. (2013). Trading-off among accuracy, similarity, diversity, and long-tail. In *Proceedings of the 7th ACM conference on recommender systems*

- *RecSys 13.* ACM Press.

Si, X., & Sun, M. (2009). Tag-lda for scalable real-time tag recommendation. , *6*(1), 23–31.

Slaney, M., & White, W. (2006). Measuring playlist diversity for recommendation systems. In *Proceedings of the 1st acm workshop on audio and music computing multimedia* (pp. 77–82). New York, NY, USA: ACM.

Smyth, B., & McClave, P. (2001). Similarity vs. diversity. *Case-Based Reasoning Research and Development*, 347–361.

Steyvers, M., & Griffiths, T. (2007). Probabilistic topic models. *Handbook of latent semantic analysis*, *427*(7), 424–440.

Su, X., & Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. *Adv. Artificial Intellegence*, *2009*, 421425:1–421425:19.

Takács, G., Pilászy, I., Németh, B., & Tikk, D. (2008). Matrix factorization and neighbor based algorithms for the netflix prize problem. In *Proceedings of the 2008 ACM conference on recommender systems - RecSys 08.* ACM Press.

Vall, A., Eghbal-zadeh, H., Dorfer, M., Schedl, M., & Widmer, G. (2017). Music playlist continuation by learning from hand-curated examples and song features: Alleviating the cold-start problem for rare and out-of-set songs. In *Proceedings of the 2nd workshop on deep learning for recommender systems, dlrs@recsys 2017, como, italy, august 27, 2017* (pp. 46–54).

Vall, A., Quadrana, M., Schedl, M., Widmer, G., & Cremonesi, P. (2017). The importance of song context in music playlists. In *Proceedings of the poster track of the 11th ACM conference on recommender systems (recsys 2017), como, italy, august 28, 2017.*

Vargas, S., & Castells, P. (2011). Rank and relevance in novelty and diversity metrics for recommender systems. In *Proceedings of the fifth ACM conference on recommender systems - RecSys 11.* ACM Press.

Vinagre, J. (2016, 06). *Scalable adaptive collaborative filtering.*

Vinagre, J., Jorge, A. M., & Gama, J. (2014). Fast incremental matrix factorization for recommendation with positive-only feedback. In *User modeling, adaptation, and personalization - 22nd international conference, UMAP 2014, aalborg, denmark, july 7-11, 2014. proceedings* (pp. 459–470).

Wang, M., Kawamura, T., Sei, Y., Nakagawa, H., Tahara, Y., & Ohsuga, A. (2013). Context-aware music recommendation with serendipity using semantic relations. In *Semantic technology - third joint international conference, JIST 2013, seoul, south korea, november 28-30, 2013, revised selected papers* (pp. 17–32).

Xie, M., Lakshmanan, L. V. S., & Wood, P. T. (2012). Composite recommendations: from items to packages. *Frontiers of Computer Science*, *6*, 264-277.

Xiong, N., & Funk, P. (2006). Building similarity metrics reflecting utility in case-based reasoning. *Journal of Intelligent and Fuzzy Systems*, *17*(4), 407–416.

Yu, C., Lakshmanan, L., & Amer-Yahia, S. (2009). It takes variety to make a world: Diversification in recommender systems. In *Proceedings of the 12th international conference on extending database technology: Advances in database technology* (pp. 368–378). New York, NY, USA: ACM.

Zhang, M., & Hurley, N. (2008). Avoiding monotony. In *Proceedings of the 2008 ACM conference on recommender systems - RecSys 08.* ACM Press.

Zhang, Y. C., Séaghdha, D. Ó., Quercia, D., & Jambor, T. (2012). Auralist. In *Proceedings of the fifth ACM international conference on web search and data mining - WSDM ́12.* ACM Press.

Ziegler, C.-N., McNee, S. M., Konstan, J. A., & Lausen, G. (2005). Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on world wide web - WWW ́05.* ACM Press.