A Hybrid Recommender System for Improving Automatic Playlist Continuation

Anna Gatzioura, João Vinagre, Alípio Mário Jorge, Miquel Sànchez-Marrè

Abstract—Although widely used, the majority of current music recommender systems still focus on recommendations' accuracy, user preferences and isolated item characteristics, without evaluating other important factors, like the joint item selections and the recommendation moment. However, when it comes to playlist recommendations, additional dimensions, as well as the notion of user experience and perception, should be taken into account to improve recommendations' quality.
In this work, HybA, a hybrid recommender system for automatic playlist continuation, that combines Latent Dirichlet Allocation and Case-Based Reasoning, is proposed. This system aims to address "similar concepts" rather than similar users. More than generating a playlist based on user requirements, like automatic playlist generation methods, HybA identifies the semantic characteristics of a started playlist and reuses the most similar past ones, to recommend relevant playlist continuations. In addition, support to beyond accuracy dimensions, like increased coherence or diverse items' discovery, is provided. To overcome the semantic gap between music descriptions and user preferences, identify playlist structures and capture songs' similarity, a graph model is used.
Experiments on real datasets have shown that the proposed algorithm is able to outperform other state of the art techniques, in terms of accuracy, while balancing between diversity and coherence.

Index Terms—Hybrid recommender system, Automatic playlist continuation, Music recommender systems, Latent Dirichlet Allocation, Case-Based Reasoning, Beyond accuracy dimensions.

1 Introduction

The recent capabilities of digital music production have enabled the creation and public distribution of music, with lower costs and without geographic limits. This has resulted in an increased amount of music items and information about them, easily accessible [1]. In addition, it has led to a change in the way that music is "consumed" as, more than "owning" songs in their music libraries, users tend to listen to them online [2]. Therefore, various issues, related to discovery, organisation, sharing and information services, that need to be facilitated to help users in finding and properly experiencing the music they want, have arisen [3].

Music Recommender Systems (MRSs) are an appropriate response to the information overload related to music, and help users to find relevant music items, that otherwise would be hard to unveil [3]. In general, 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 [4]. Furthermore, to support users in organising their personal collections and in continuing their existing playlists, increased focus has been lately placed on *automatic playlist generation (APG)* and *continuation (APC)* [5], [6]. *Playlists* can be defined as *sets of songs* designed to be consumed as a sequence, similar to traditional radio broadcasts [7]. Thus, more than predicting whether a music item would be highly or poorly rated, the underlying structure of *joint song selections* should be evaluated. The task becomes to recommend sets of songs to be consumed together, satisfying at the same time cognitive properties like relevance, coherence and diversity [8].

However, as there is still a lack of solid methods combining users' perception of music with sound characteristics, it becomes even more difficult to capture users' perception of playlists and specify the exact characteristics that the songs composing a "good" playlist should have. This notion can be highly subjective, depending on various parameters like the user's music knowledge, character, culture, emotional state, context and intent [6], [9]. In addition, this application domain, except from highlighting the importance of joint item selections, is also heavily affected by the so called *semantic gap* present in Music Information Retrieval (MIR) systems. In general, as shown in figure 1, users describe their needs through high level requirements (expressed through terms like "relaxing music", "smooth transitions", etc.) while the treated items are associated with low-level, sound-based characteristics (like pitch, tempo, etc.) [10]. Therefore, appropriate MRSs need to be designed, being able to capture users' needs, handle the semantic gap, and recommend relevant items that could satisfy them [11].

In this paper, we propose *HybA*, a hybrid recommender system for *automatic playlist continuation*. This system recommends *sets of songs* to complete a started playlist, being coherent and of some diversity degree, in order to provide a more complete user experience. In general, APC can be considered as a variation, or subcase, of APG. Rather than generating entire playlists based on some given target characteristics, in APC those characteristics must be inferred from the started playlist, to design sets of songs maching those, to continue the list [6].

- *A. Gatzioura is with Universitat Politècnica de Catalunya / KEMLG - IDEAI, C/Jordi Girona 1-3, 08034, Barcelona, Spain. E-mail: anna.gatzioura@gmail.com*
- *J. Vinagre and A. M. Jorge are with University of Porto / LIAAD - INESC TEC.*
- *M. Sànchez-Marrè is with Universitat Politècnica de Catalunya / KEMLG - IDEAI.*

Manuscript received December 23, 2018.

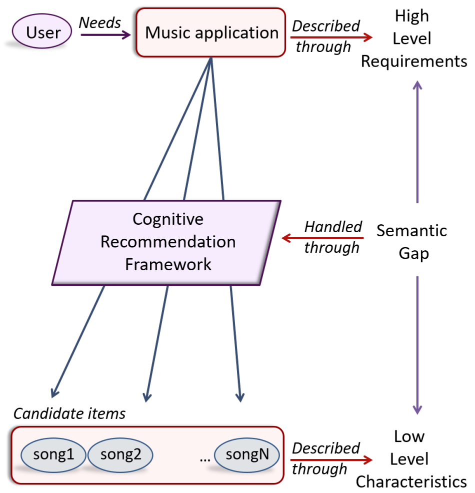

Fig. 1: The semantic gap in music (recommendation and information retrieval) applications

The proposed algorithm evaluates the similarity among playlists based on their general characteristics, and on the music styles' distributions in them. When a new list is introduced, HybA first identifies its *concept* [12] and then follows the general Case-Based Reasoning (CBR) cycle [13] to find the past most similar one(s), and uses their characteristics to construct an adequate continuation. The degree of similarity of music items is calculated based on the density of their connections in a graph model that connects songs through their common attributes and appearances in playlists.

The basic functionalities that differentiate this algorithm from the commonly used playlist generation and recommendation techniques, that also form the contributions of this work, can be summarised in the following points:

- The proposed algorithm focuses on the characteristics of *entire playlists* and the generated recommendations aim to address *similar concepts*, rather than similar users.
- In contrast to most sequential techniques, that start from a state (song) and try to predict the next one, our algorithm evaluates more than the last selected song, and recommends sets of songs.
- Two abstraction levels are used: the first evaluates more general characteristics of the playlists, and then at a second level, the songs in those are analysed.
- The designed algorithm permits beyond accuracy dimensions, related to playlists' quality –like coherence and diversity–, to be evaluated and incorporated into the final recommendations.

The rest of this paper is structured as follows: In the next section we describe the problem and set the research questions being addressed. In section 3, a brief overview of music and playlist recommendation techniques is provided. Following, in section 4, the presentation of our algorithm, the entities' modelling, the pre-processing phase and the playlists generation process, can be found. In section 5, the comparison of the proposed system with some of the currently used recommendation techniques is presented. The results show that the proposed methodology is able to improve recommendations' accuracy and quality. Finally, in the last section we conclude and present some of the topics that form our ongoing and future work.

2 Problem Description

The general scope of a recommender system (RS) is to find the items with maximum utility under a given user's expectations, at a given moment. Usually, user similarities and item characteristics are used to generate an ordered list of items that are most likely to satisfy a user [14].

However, in music (like in travelling, market basket analysis, etc.) items are usually consumed together with others, in sessions. In these domains, the "independence assumption" that users select items based only on their preferences over item characteristics, is not valid. Item interactions within a set may be crucial for the acceptance or the rejection of an item, and the quality of the set, because users use and evaluate the whole set, rather than selecting items from a top-N list [15]. For example, a user may like both classic and hard rock music, but normally would not place songs of both genres into the same playlist. More likely, the user would organise those into coherent playlists to be listened to in different occasions [16]. This example highlights the existence of an *underlying music concept* behind the joint selections of songs for a playlist [12].

Therefore, the functionalities that playlist, and collection recommenders in general, should serve are slightly different from those of the usual RSs. We believe that these algorithms should follow a different design, and an appropriate evaluation method, able to capture the additional dimensions of the problem. To this direction, we hypothesise that a hybrid solution would be more appropriate to address the problem's needs. Further than the user-item relationships, emphasis should be placed on *sessions* and the music items selected within similar sessions should be analysed, as these may be representative of a specific sound that a user is seeking for [17]. These systems should present recommendations related to a given *concept* that has to be deduced from the underlying data patterns. Finally, more than focusing only on their predictive ability, these systems should incorporate parameters related to playlists' quality and present interesting alternatives that the users would not find otherwise [18].

In figure 2, we present the general schema of a cognitive architecture that we consider as appropriate for playlist continuation recommendations. As shown in this figure, the important characteristics affecting the user perception and the quality of the playlist are extracted from the data patterns related to playlists and songs in them.

Based on the scope and the desired characteristics of the system, we set and try to address the following research questions:

Q1: Does the proposed algorithm address the needs of the specific problem in a better way than the commonly used recommendation techniques?

Q2: Are the commonly used information retrieval (IR) metrics sufficient for the evaluation of collection recommendations? Which other factors, *beyond accuracy*, should be evaluated?

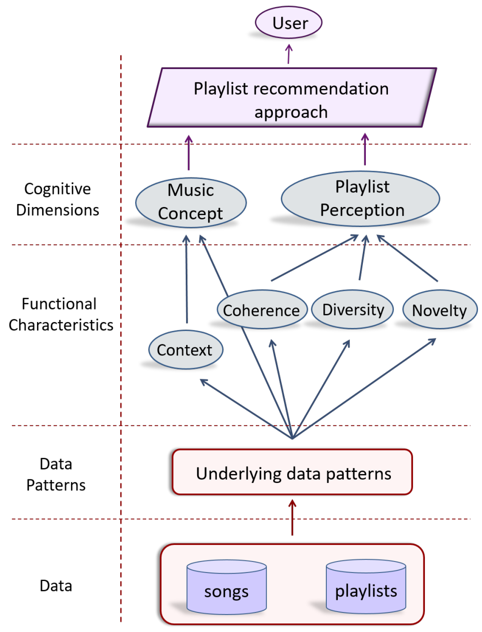

Fig. 2: General schema behind the proposed playlist continuation recommendation approach

3 Background

MRSs have their basis both in the fields of RSs and MIR. Therefore, they inherit some characteristics, and also limitations, of both areas [10], [19]. Traditional MIR approaches use content-based (CB) techniques, that apart from the general limitations of CB systems like over-specialisation and limited diversity, require a deeper knowledge of the application domain [1], [20]. On the other hand, Collaborative Filtering (CF) techniques are domain independent and focus mainly on user ratings. They aim to predict the rating a user would assign to a previously unknown song or artist (scoring) or rank the candidate items (top-n recommendations), based on the behavior of similar users. Thus, they suffer from cold-start and long tail effects [14]. In addition, the majority of MRSs shows a tendency towards popular songs, and does not properly evaluate situational and other contextual parameters under which the selection of a song is done, that may be crucial [19], [21].

However, songs are rarely listened to in isolation. Users rather create playlists (or sessions), being sets of songs, placing more importance on the music items selected and their relative order [22]. Therefore, when recommending a playlist, apart from the characteristics of each song, it must have some cognitive characteristics as a whole, like *variety, diversity* and *coherence*. Variety refers to not repeating the same song or artist in the recommended sequence (or at least not often), diversity refers to not recommending closely similar songs, while coherence evaluates their relative order, as the transitions should be smooth [16], [23]. Various techniques have been proposed to infer the structure of "good" playlists and identify their desired characteristics, to form a pleasant and satisfactory result for the listener.

APG algorithms usually work based on constraints or hints, being a seed item provided by the user, and can be divided mainly into three categories [24]. *Constraint satisfaction methods* that generate playlists based on some user entered criteria, *similarity heuristics* that given a seed music item, e.g. a song or an artist, use some similarity function to identify the most similar ones and finally *machine learning approaches* that build a model based on a training set of playlists that then generates recommendations. Bonnin and Jannach [5] review the mostly used methods for APG. As important issues in this application domain are the song co-occurrences and the smooth transitions among them, Markov models and association rules', or sequential patterns', mining are among the techniques mainly used. The usual recommendation approaches, like CF and CB, are also being used. Their major limitations, when applied to playlist recommendations, are their computational cost and the strong assumptions they rely on.

Song similarities can be identified based on the metadata associated with them or through sound related data [20]. Maillet et al. [16] propose the use of a similarity function that given some audio files generates the probability of those to be played successively in a playlist, using their audio features and past observations in playlists made by professional radio stations. Playlist recommendations are then generated starting from a seed song based on its transitional probabilities to the rest.

Usually, songs frequently placed in sessions together, tend to have some characteristics in common, like a particular genre, pairwise suitability, etc. Therefore, Ragno et al. [25] base their reasoning on the implicit likeliness that can be found in playlists generated by professionals, like music radio stations and Djs. They represent songs as nodes of a graph, while the edges among adjacent songs have a weight equal to the number of times that this transition was observed. The resulting graph is transformed into a Markov random field and a playlist can be generated as a random walk starting from a given song. Although we also use a graph in our system, this graph is used to represent the relations among the problem entities and to calculate similarities, rather than for generating recommendations.

On the other hand, Baccigalupo and Plaza [23] present an interesting Case-Based recommendation approach for playlists of a desired length, being both varied and coherent. Playlist are treated as cases, and their relevance is computed based on song co-occurrences, while recommendations are formed using the songs in the most similar lists. The authors also analyse the properties that may bias the effectiveness of their approach, namely songs' popularity and sub-lists' length. This work is probably based on a similar idea to ours, but in our system first a Latent Dirichlet Allocation (LDA) topic model is built and then CBR is applied on a refined set of cases.

Pichl et al. [26] propose the use of the implicit contextual information found in playlist titles, to perform a contextual clustering and filtering, using Natural Language Processing (NLP). This approach could be useful when titles contain "objective" information, like "Christmas" or "summer", but titles like "my favorite music" may lead to clusters not valuable for the recommendation process. Additionally, there may be various dimensions of context, perceived differently by users depending on their personality and emotional state, and thus differently reflected in their music selections. For instance, Ferwerda et al. [27] have shown that, apart from their general preferences, extroverted people in negative emotional states tend to rely on "happy" music, in contrast to more neurotic people. This subjective perception of context is among the reasons why in our previous work we have proposed the use of the term *playlist concept* to infer playlists' context and purpose [12].

Finally, Hariri et al. [28] also propose the use of an LDA topic model to implicitly capture the context of a playlist. However, they represent songs as topic distributions over tags coming from social tagging web services. The context of a playlist is then inferred from the sequences of transitions in latent topics to predict a user's next topic of interest. Rather than modelling songs, in HybA, LDA is used to model playlist characteristics and then CBR is applied to construct playlist continuations.

4 Recommendation Approach

In this work, we present *HybA*, a hybrid recommendation approach that, given a new, started, playlist, aims to construct and recommend playlist continuations, of improved quality. These recommendations do not aim to address specific users, but are related to a specific "semantic concept", inferred each time from the started playlist [5]. HybA is based on a meta-level hybridisation, LDA is applied first to the entire set of past playlists and then, for each new playlist, CBR is applied on a refined set of playlists.

LDA, and probabilistic topic models in general, are thought to be similar to the human early learning processes related to language. In these cases, we try to learn the meaning of words based on the concepts within which they are both present and absent. Furthermore, music learning processes are considered to be similar to language [29]. On the other hand, 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 try to identify the most similar experienced cases and adapt their solutions appropriately to the new problem [30].

Users' music preferences and needs may vary along time, being in general influenced by the context under which they were generated. Therefore, even the same user may look for different items in different moments. In addition, context may be differently perceived by users, thus differently reflected on their music selections [31]. For example a rainy day may be perceived as a "sad" situation by some users. Thus, HybA does not base its reasoning neither on user related data nor on explicit context, but rather aims to address similar *playlist concepts*. Furthermore, its emphasis is on the *co-occurring patterns* of music styles in playlists to infer the mood and purpose of their generation [12].

The presented system follows the general process and the basic idea of CBR, that "Similar problems have similar solutions" [13]. Similarly, given a new playlist, HybA finds the past most similar one(s), and uses their characteristics to construct the set of songs that seem as the most appropriate for completing the new playlist. This algorithm forms an extension of our previous proposal [32], that has been found able to identify and recommend accurately artists or song styles that would better fit within started playlists. We have extended that model to a two-level model to better capture the cognitive characteristics of entire playlists. Thence, first general playlist characteristics and similarities are evaluated and then additional song attributes and quality related parameters are analysed. This process permits the control of beyond accuracy dimensions, whose levels should also follow the tendencies observed in the started playlist [11].

4.1 Problem 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 as *sessions*. Each playlist is a session, being a *set of music items* that have been selected together by a user, at a given time moment.
The problem data entities can be described as:

- a dataset 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 the set of tags associated with it, as $i_j = \{t_{j1}, t_{j2}, \ldots, t_{jm}\}$, $t_{jk} \in T$, $k = 1, \ldots, m$
- a set of past playlists $L = \{l_1, \ldots, l_k\}$ where each $l_j \in L$ can be written as the set of songs it consists of, as $l_j = \{i_{j1}, \ldots, i_{jn}\}$, $i_{jt} \in I$, $t = 1, \ldots, n$
- a set of users $U = \{u_1, \ldots, u_v\}$ that have formed those playlists

In addition, each of these entities may be associated with additional characteristics, like metadata, editorial (genre, tempo, artist, lyrics, etc.), demographic (age, gender, origin, etc.), temporal or contextual (timestamp, source of reproduction etc.), information, respectively. Emphasising on the song descriptions, based on the available information related to them, the used tags could be editorial metadata, user reviews or sound features possibly coming from a proper analysis process. In general, user tags especially from non-experts are highly subjective and tend to add sparsity and noise to the recommendation problem. Thus, it is preferable to describe songs based on more accurate information, like editorial or sound-based metadata [33].

In order to identify the possible interactions, 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 problem entities, namely users, playlists, songs and tags, thus $V = \{U, L, I, T\}$
- 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 3, we may have the following, as well as their inverse, relationships:

- If $e(i_j, t_k) \in E$: Song $i_j \in I$ "has" tag $t_k \in T$
- If $e(l_j, i_t) \in E$: List $l_j \in L$ "contains" song $i_t \in I$
- If $e(u_j, l_i) \in E$: User $u_j \in U$ "made" playlist $l_i \in L$

In addition, after having analysed the whole graph we are led to a set of different tag combinations, let them be $S = \{s_1, \ldots, s_k\}$, referred to as *music styles*. This term refers to the distinct tag combinations associated with the songs in a music database, not necessarily related with the explicitly stated musical categories of the songs. Furthermore, each

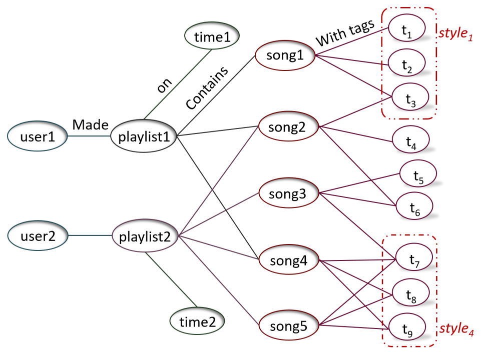

Fig. 3: Playlist and song distributions

$s_j \in S$ can be written as the set of tags it is associated with, being $s_j = \{t_{j1}, \ldots, t_{jm}\}$, $t_{jk} \in T$, $k = 1, \ldots, m$ and is finally associated with one *song cluster*. The songs in each of those clusters have almost identical descriptions, like songs 4 and 5 in figure 3, and could eventually be used in the same cases. We then aim to identify the presence, or absence, of music styles in playlists and the resulting patterns.

Nevertheless, the above 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. HybA focuses on playlist characteristics and does not evaluate user similarities. Therefore, mainly the song – tag, the playlist – song and the playlist – music style, implied from the previous two, relationships are used in its current version.

4.2 Recommendation Process Overview

In order to support the needs of the specific problem, the functionalities of the recommendation algorithm are performed in the following three basic steps, that will be described in detail in the following sections.

- *Data pre-processing*: the parameters requiring costly calculations and not changing their values during the recommendation process, like song similarities and past playlist characteristics, are computed.
- *Candidates' retrieval*: based on the characteristics of a started playlist, like its concept and the contained music styles, the relevant past playlists are retrieved and ranked.
- *Playlist Continuation Construction*: the playlist continuation is constructed by appropriately combining the music items in the top ranked playlists.

4.3 Data Pre-processing

4.3.1 Song Similarity

Our focus is placed on the playlists' characteristics and the music tendencies in them, rather than on the exact songs selected. Therefore, we aim to identify songs with the same characteristics, being of the same music style, that could be possibly "interchanged" in playlists.

Songs are associated with sets of tags –being user generated tags, real music features, latent descriptions, etc.– that allow us to model them using hierarchical models, vectors, or graphs. After evaluating those alternatives, the *graph model* has been selected as the most appropriate for our goal [11]. A hierarchical model would require domain knowledge specifying the relative importance of the item attributes to correctly categorise them. Such model could be convenient when referring to sets of items of different types, like in market basket analysis [34]. However, when referring to songs, as the importance of certain characteristics may be highly subjective, an a-priori hierarchical categorisation cannot be defined. For example, one user may categorise songs first based on their tempo while another one based on their lyrics' language. Finally, calculating item similarity based on feature vectors is usually computationally expensive when sparse tags form the item descriptions.

We model songs as nodes of a graph, connected with the tags that form their specifications (similar to figure 3), treat all attributes as being of the same importance, and calculate their similarity as a function of their unique and common connections. More precisely:

- Given two music items, described as: $i_a = \{t_{a1}, \ldots, t_{al}\}$ and $i_b = \{t_{b1}, \ldots, t_{bk}\}$
- Given that $n(a \cup b)$ is the number of tags that i_a, i_b, or both, have, $n(a \setminus b)$ 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, refered to as *local similarity*, is calculated using the following formula [32]:

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

Depending on the desired level of detail, and the existing data sparsity, we may use all, or some, of the songs' tags (i.e.: differentiating between artist and genre related data where explicitly stated). Thus, we have more specific or abstract music item descriptions, being songs or music styles.

4.3.2 Playlist General Concept

In [12], we have introduced the term *playlist concept*, as a wider term to capture the general characteristics of a playlist related with its central idea, used to implicitly capture its contextual characteristics and purpose. The use of this modelling has been found to reduce the computational time during the candidates' retrieval phase and outperform that of explicit context when incorporated to our system [11].

To capture the playlists' concepts, without focusing on their specific content or explicitly defined context, we use their latent topic distribution, based on the music styles of the songs in them. An LDA topic model [35] is built on the playlists in the problem case base, described as sets of music styles, to extract the underlying latent topics characterising the music style combinations. The number of the retrieved topics depends on the dataset size scale, normally between 200 and 300. Having every playlist described as a distribution over music styles permits us to find its probability distribution over the exctracted latent topics. Then, playlists are characterised by their *dominant topic(s)*, which are used as a representation of their general concept, considered as part of their cognitive definitions [11], [12].

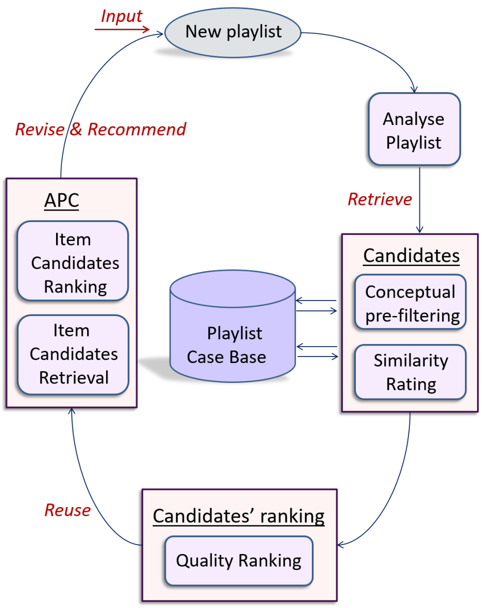

Fig. 4: APC process

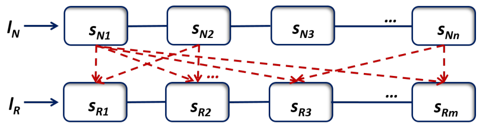

Fig. 5: Comparing a new and a retrieved list

4.4 Recommendations' Generation Phase

In HybA entire playlists are treated as cases, that may be *sequences of songs* or, by using a higher level of abstraction, as *sequences of music styles*. HybA uses sub-graphs of the whole problem graph to identify the music styles of the songs that the playlists are formed of, and based on them, calculates the playlists' degree of similarity.

In figure 4, the followed CBR cycle is presented, highlighting the important processes that take place, from the moment the new playlist is entered, until its continuation is 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 $l_j \equiv c$. For the recommendation problem the "new" (N) playlist $l_N \equiv p = \{i_{N1}, \ldots, i_{Nk}\}$, being the set of the initially selected songs, is used as problem description, while the solution is the recommended (rec) playlist continuation of length n, $l_{rec} \equiv q = \{i_{N(k+1)}, \ldots, i_{N(k+n)}\}$.

4.4.1 Candidates' Retrieval

Given a new playlist l_N, a set of songs that a user has already selected, we first identify the topic distribution of the music styles of those songs, being the concept of this list. Then a conceptual pre-filtering of the problem case base takes place, as a way to capture the implicit contextual dimensions reflected in the playlists and to also reduce the computational time. Only the playlists $L_C \subseteq L$, of the same concept with the started playlist, are retrieved and used for the rest of the computations. From those, the k most similar to l_N, that maximise the playlist similarity, also referred to as *global similarity*, are identified using equation (2).

$$l' \in L_C : \forall l_R \in L_C,\ l' = argmax\{Sim(l_N, l_R)\} \quad (2)$$

Let a new playlist be $l_N = \{s_{N1}, \ldots, s_{Nn}\}$, and a retrieved one $l_R = \{s_{R1}, \ldots, s_{Rm}\}, l_R \in L_C$, of length $n_N = |n|$ and $n_R = |m|$, with the *i-th* item in each being s_{Ni} and s_{Rj}, referring to the music style of songs i_{Ni} and i_{Ri}, respectively.

In order to find the lists that address equation (2) our algorithm performs a two level similarity calculation. First the similarity degree of each item in the new list is specified based on its local similarity values with the items in the other list. Then the global similarity of the two lists is calculated as an aggregation of those values.

The *local similarity* of music styles is calculated during the data pre-processing phase using equation (1). The *global similarity* of two playlists, $Sim(l_N, l_R)$, is the *aggregated similarity* of the similarity levels of the music items in them. Among the combinations of aggregation methods tested, the best results are achieved when using equation (3) [11],

$$Sim(l_N, l_R) = \frac{1}{|n|} \cdot \sum_{i=1}^{n} max_{j=1}^{m}\{sim(s_{Ni}, s_{Rj})\} \quad (3)$$

Where $max_{j=1}^{m}\{sim(s_{Ni}, s_{Rj})\}$ finds the maximum similarity degree between each music style in the new playlist and those in a retrieved one, like in figure 5, and then their average is set as the global similarity of the two playlists. As the sum of similarities is averaged by the length of the new playlist, the used similarity metric is not symmetric. Furthermore, a playlist l_R may be among the most similar to l_N, without necessary the opposite being true.

HybA is based on the identification of the playlists that maximise (2). Therefore, the following equation is used,

$$l' \in L_C : \forall l_R \in L_C,$$
$$l' = argmax\Big\{\frac{1}{|n|} \cdot \sum_{i=1}^{n} max_{j=1}^{m}\{sim(s_{Ni}, s_{Rj})\}\Big\} \quad (4)$$

Finding the k most similar past playlists using (4), and recommending directly the music items in them, has been found to be efficient when recommending relevant "song clusters", like music styles or artists that would fit within the started playlist [32]. However, when the aim is to efficiently recommend songs, the data sparsity usually would limit the performance of such methods. Thus, our algorithm first identifies the most similar playlists based on more general characteristics, like their concept and the music styles in them, and then identifies the adequate songs.

4.4.2 Playlist Continuation Construction

Let $L'_S \subseteq L_C \subseteq L$ be the set of the k most similar playlists to l_N, that have been identified using equation (4). HybA then assigns to each playlist $l_R \in L'_S$ a rating $r_{l_R} = r(l_N, l_R)$, being a function of the characteristics of both the started and the candidate playlist, with values $0 \leq r_{l_R} \leq 1$. The candidate playlists with the higher ratings are finally identified, like in equation (5):

$$l'' \in L'_S : \forall l_R \in L'_S,\ l'' = argmax\{r(l_N, l_R)\} \quad (5)$$

The evaluation of playlist candidates is performed in two steps, first based on similarity, to ensure that a minimum relevance level is achieved, and then additional emphasis is placed on quality related parameters.

Given a similarity, or a distance, metric between two items, related as $sim(i,j) = 1 - d(i,j)$, for a list Lst of size $|Lst|$, those are defined as:

- Coherence: the average similarity among pairs of consecutive music items in the list, being [7]:

$$Coherence = \frac{1}{|Lst|-1} \cdot \sum_{i \in Lst, i<|Lst|} sim(i, i+1) \tag{6}$$

- Diversity: the average pairwise dissimilarity of the music items in the list, being [7], [36]:

$$Diversity = \frac{1}{|Lst|(|Lst|-1)} \cdot \sum_{i \in Lst} \sum_{j \in Lst, j \neq i} d(i,j) \tag{7}$$

From their definitions, coherence and diversity have contradictory scopes and may have a negative relation, as one depends on the similarity of consecutive items and the other on the dissimilarity of the included item pairs. Increasing the coherence of some items will result in a decrease of the total average diversity. On the other hand, as diversity does not depend on the items' relative order, for a given diversity level, coherence still could be improved by selecting an appropriate items' ordering.

Six variations of HybA have been designed and tested [11], each prioritising different dimensions – like similarity, or *beyond accuracy dimensions*, like *coherence* and/or *diversity*– thus using different rating functions. Three of them had as aim to maximise global similarity, coherence or diversity in the retrieved list, and three the aim to follow the corresponding tendencies present in the started playlist. Among those, the best performance is achieved by *HybA-db*, where rather than maximising or minimising one of these parameters, the aim is to follow the tendencies in the started playlist.

Let the coherence in the new and a retrieved list be $Coh(l_N)$ and $Coh(l_R)$ respectively, while their difference is $Cd(l_N, l_R) = |Coh(l_N) - Coh(l_R)|$. On the other hand, let $Dd(l_N, l_R) = |Div(l_N) - Div(l_R)|$ be the difference of the respective diversities. and their relative importance is defined by the corresponding weighting factors w_C and w_D, with $w_C + w_D = 1$.

In HybA-bd both weighting factors are set to $w_C = w_D = 0.5$ and the playlists in L'_S are rated, as:

$$r_{l_R} = Sim(l_N, l_R) \cdot \\ \big(w_C \cdot (1 - Cd(l_N, l_R)) + w_D \cdot (1 - Dd(l_N, l_R))\big) \tag{8}$$

Setting $w_C = 1$ or $w_D = 1$ leads to the version of HybA that follows only the coherence or diversity of the started list, respectively. Due to the relation between coherece and diversity, these three versions have similar performances. In addition, these are more balanced than those maximising coherence or diversity, which results in reduction of the other factor and sometimes also in accuracy decrease [11].

After having identified the set $L'' \subseteq L'_S \subseteq L_C \subseteq L$ of the k higher rated playlists from (8), HybA extracts the set of songs I'', that were initially placed in those. These songs are then assigned the aggregated final rating of the candidate playlist(s) in which they appeared. Given that r_{l_j} is the rating of a playlist $l_j \in L''$, the final score r_{i_k} of a candidate song $i_k \in I''$, can be computed as:

$$r_{i_k} = \sum_{l_j \in L'', e(i_k, l_j) \in E} r_{l_j} \tag{9}$$

Finally, songs in I'' are ranked in descending order of r_{l_j} and the top sub-list, of desired length, is recommended.

TABLE 1: Palco Principal datasets

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

5 Evaluation

To evaluate the proposed algorithm we performed experiments on real datasets, and compared it with some of the commonly used recommendation algorithms, through both accuracy and quality metrics. More information on the datasets, the evaluation metrics, the compared algorithms and the experimentation results are presented in this section.

5.1 Evaluation Datasets

The datasets that were used for the evaluation are music databases that contain information about users streaming or putting specific tracks into their playlists at a given time moment. More specifically, four datasets[1] of Palco Principal[2], 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 tracks in personal playlists, while it is characterised by a high long tail presence, as the majority of the songs are not popular songs. From those, only the fourth (Plc) contained content information on songs, namely their artist and genre. For the rest, latent features from usage data were used. More information is presented in Table 1.

5.2 Evaluation Approach

In general, the evaluation approaches of music playlist recommendations can be organised into four categories, namely: user studies, log analysis, objective measures and comparison with hand crafted lists [37]. As our scope is to automatically generate playlists being similar to the manually created ones, we have chosen the *comparison of the patterns* in the recommended and the real playlists that is usually evaluated through IR accuracy metrics.

The used datasets were divided into a training part (80%), used for building the recommendation models, and

1. The first three datasets are publicly available from https://rdm.inesctec.pt/dataset
2. http://palcoprincipal.com/

a testing part (20%). As user tastes may change over time, to avoid biasing the results, we respected the time order of the recorded transactions and used the first part of each dataset for training and the rest for testing purposes. From the playlists in the test part, we keep each time the initial part, and hide the rest, that would be equivalent to asking a user to submit a new playlist and based on it to generate continuations of the desired length. We evaluate and compare the ability of the various algorithms to correctly identify the hidden items, using IR metrics like precision, recall and F-measure, defined as:

$$\begin{aligned} Precision &= \#RelevantRecItems/\#RecommendedItems \\ Recall &= \#RelevantRecItems/\#RelevantItems \qquad (10) \\ F-measure &= \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \end{aligned}$$

However, the problem when using these evaluation metrics is that they are “too strict” as they only classify recommendations as successful when the hidden songs are correctly identified. Therefore, when recommending similar, relevant songs, that could make good recommendations and possibly were not selected due to popularity bias, those are not positively evaluated. We propose, and test, the use of hit ratio together with average similarity, as indicators of recommendations’ relevance. The total hit ratio could be used as a more general metric, of the times that a user would get at least one correct recommendation. It is calculated as:

$$HitRatio = \#RelevantRecs/\#Recommendations \qquad (11)$$

To have a view of “how close” the real and the recommended lists were, we use the *average similarity* of the two lists, calculated as their global similarity, being (3):

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

In addition, as in MRSs the focus shifts to user experience, those metrics are combined with *beyond accuracy metrics* to evaluate the quality of the recommended playlist continuations [2]. Therefore, the *coherence* and *diversity* of the recommended playlist continuations are calculated using equations (6) and (7), and are also presented.

We follow the hypothesis that coherent lists provide a more pleasant listening result [38]. Thus, they are considered as “better”. On the other hand, the diversity degree depends on the user’s current taste as it can be deduced from the started list. As coherence and diversity may have a negative relation, in order to have a pleasant result there should be a balance among them. In addition, the balance between accuracy and diversity still forms an open problem, as very diverse recommendations, or many unknown items, may fail to address users’ expectations, thus may harm the system’s reputation [39]. To evaluate the overall performance of the algorithms related to both accuracy and diversity, we use the F-measure of precision and diversity [40],

$$F_d - measure = \frac{2 \cdot Precision \cdot Diversity}{Precision + Diversity} \qquad (12)$$

5.3 Comparison with other Recommendation Techniques

We further compare HybA-db (with 200 topics), the version of HybA with the best overall performance, with the following recommendation algorithms, that have been used in commercial RSs applications or proposed in the literature:

- *Latent Dirichlet Allocation (LDA)*: Playlists are treated as “documents” formed as probability distributions over topics, and songs as “words” coming from (200) different topics. A LDA topic model was built on the playlists in the training part and recommendations were generated each time based on the topic distribution of the songs in started list [41].
- *Collaborative Filtering (CF)*: Although CF does not focus exactly the same problem dimension, as it does not evaluate song co-occurrences, its results (10 nearest neighbours) are also listed here, as it is among the most popular recommendation techniques. As no song ratings were available, user preferences were inferred from songs’ selection frequencies.
- *Incremental Matrix Factorisation (ISGD)*: This Matrix Factorisation algorithm, presented in [42], works with positive-only feedback and treats songs in playlists as sequential data, more than evaluating past entire listening sessions. It has been found to outperform other CF and incremental factorisation algorithms in most cases, while also having a good runtime performance.

In figures 6–10 we present the graphical results for precision (recall and F-measure values are closely similar), hit ratio, average similarity, coherence and diversity, for recommendations of playlists of different lengths, on the four datasets.

As it can be seen from figures 6 and 7, *precision* and *hit ratio* follow similar evolution patterns for all the evaluated algorithms. In addition, as depicted in figure 8, *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. These facts support our hypothesis that those metrics could be used as an alternative way to evaluate playlist 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, if correctly combined with beyond accuracy metrics, average similarity could serve as an indicator of serendipity.

In general, session-based techniques (HybA and LDA) have been found to perform better, highlighting that conventional recommendation techniques fail to capture the specific cognitive characteristics found in item co-occurrences. This supports our hypothesis that a model focusing on entire sessions’ characteristics is more adequate for this problem. 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. Therefore, more than the user-item relationships,

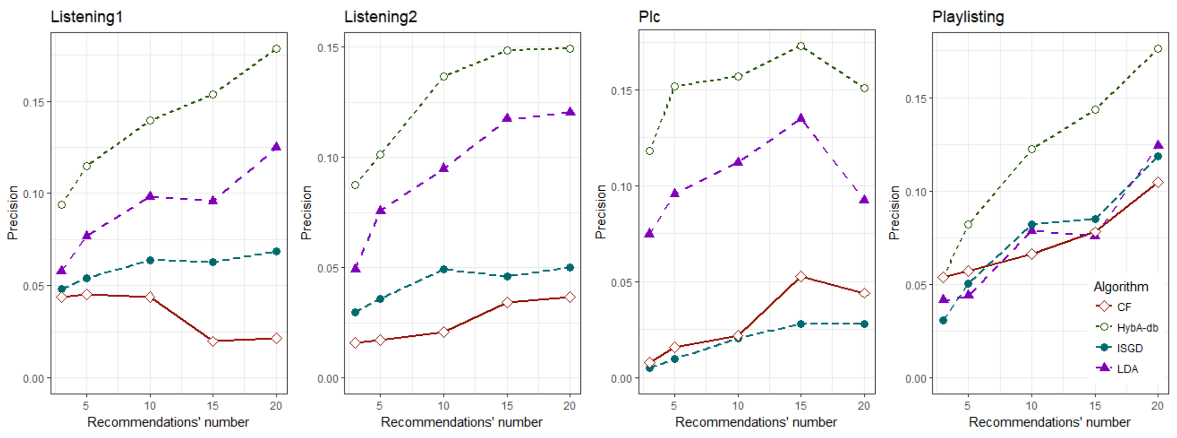

Fig. 6: Recommendations' precision for the various techniques

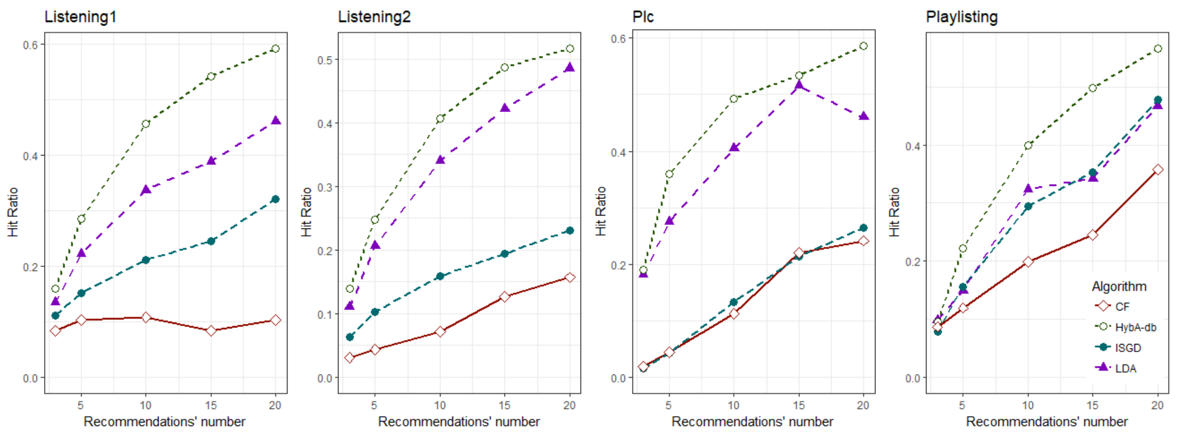

Fig. 7: Recommendations' hit ratio for the various techniques

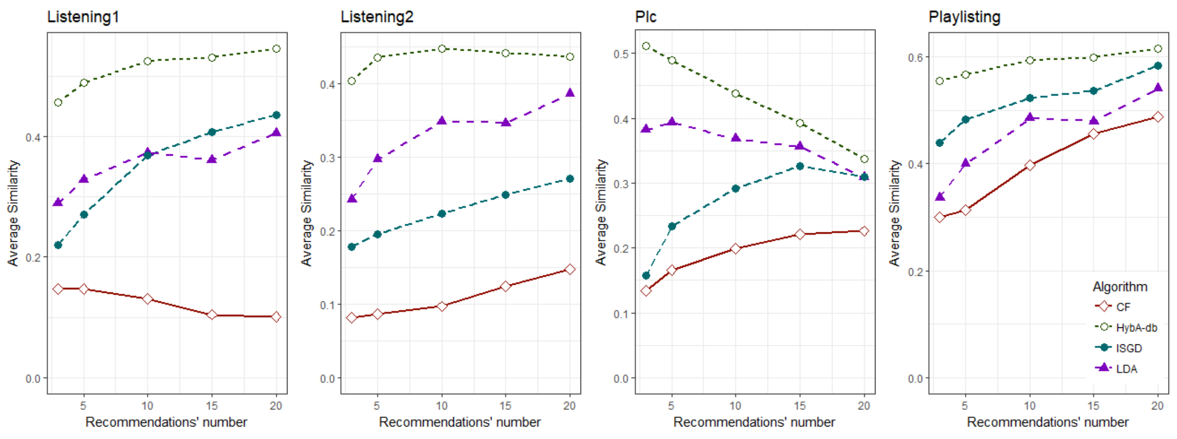

Fig. 8: Average similarity of the real (hidden) an the recommended playlists by the various techniques

the user-concept and concept-item relationships have to be identified. Although, ISGD also works based on user-item relations, as its reasoning is based on data streams, it is able to capture the actual tendencies in user profiles, reflected in playlists. Therefore, it performs better than pure CF. The increased data sparsity and long tail percent in the datasets should be also taken into account when comparing the algorithms. The performance of the algorithms that need a representative enough user profile in order to perform well, is expected to decrease more due to those phenomena.

In figures 6 and 7 we observe that HybA-db and LDA follow similar patterns, as both methods base their reasoning on LDA probabilistic topic models. However, HybA performs a latent analysis at a more abstract level, using song styles rather than songs, which makes it more flexible, and is further combined with a more complex item similarity model that improves its accuracy. LDA, on the other hand, based on the probability distribution of a playlist directly

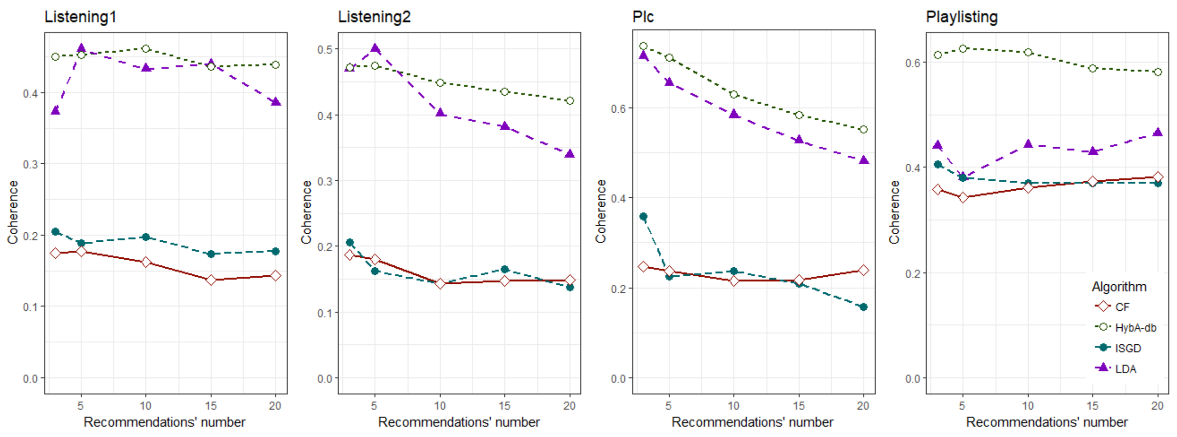

Fig. 9: Recommendations' coherence for the various techniques

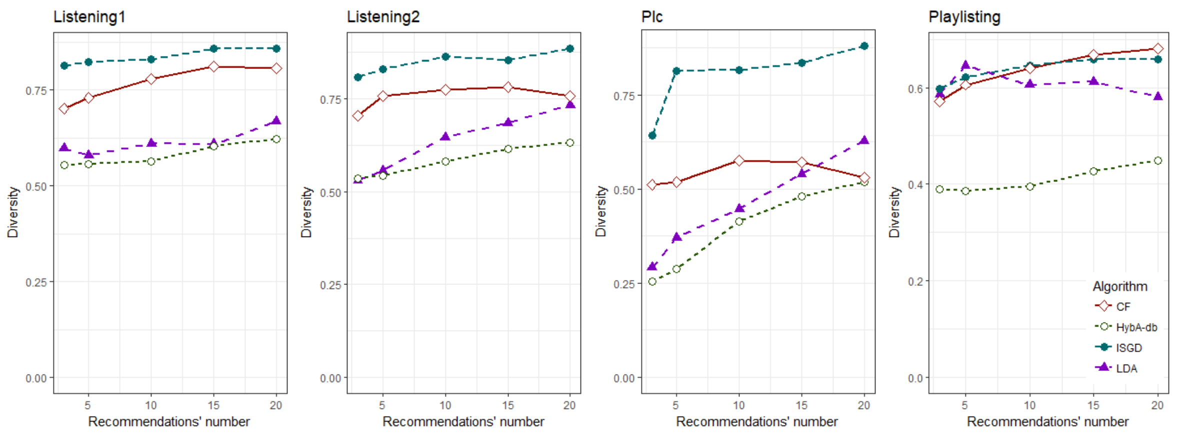

Fig. 10: Recommendations' diversity for the various techniques

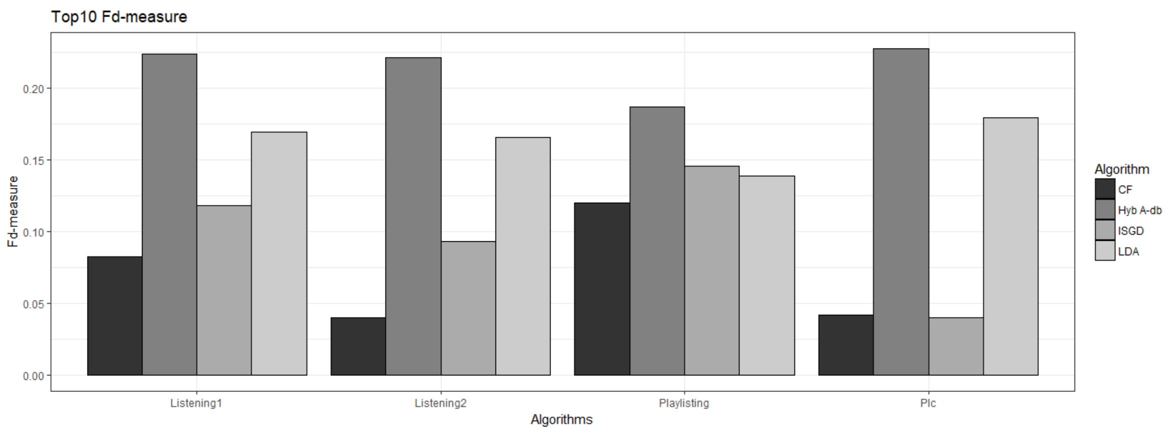

Fig. 11: F_d-measure for recommendations of 10 songs

recommends the most popular songs of its dominant topic.

From figures 9 and 10, where average *coherence* and *diversity* are shown, we observe that depending on the algorithm's emphasis there is a clear difference in the levels of coherence and diversity, achieved. In general, session-based techniques focus more on the coherence of playlists while user-based techniques follow the diversity of user profiles. Among the tested algorithms, HybA-db recommends the most coherent playlists. Although being less diverse, these playlists have diversity levels similar to the ones of the real playlists. Although presenting more diverse recommendations, the compared algorithms do not manage to combine diversity with relevance, as their accuracy levels are significantly lower than the ones achieved by HybA-db. As shown in figure 11, were the F_d-measure for recommendations of playlist continuations of 10 items is presented, HybA-db manages to best balance between accuracy and diversity.

In general, HybA has been found to outperform the compared ones, in terms of accuracy while having a good trade-off with parameters beyond accuracy, in all the used datasets. This confirms our initial hypothesis that, a hybrid approach designed for the specific problem would be able to perform better than the usual recommendation techniques, in terms of accuracy, while taking into account more cognitive dimensions related to playlist quality.

6 Conclusions and Future Work

In this paper, we have presented HybA, a hybrid recommender system for automatic playlist continuation, one of the current challenges that MRSs are facing. When provided with a started playlist, this algorithm aims to recommend the most appropriate continuation [6].

HybA focuses on the analysis of entire playlists. To overcome the existing semantic gap and capture more cognitive dimensions of the playlists, this analysis is performed at two levels. A graph-based model is used to describe songs based on tags, being metadata, or other information available related to their styles. Song similarity is then calculated based on the density of their common connections. On the other hand, recommendations are generated through a hybrid approach, combining LDA and CBR, to identify the similarity of playlists, in terms of music concepts and styles usually enjoyed together. Additional support to playlist coherence and diversity is provided to capture more characteristics related to playlist quality and user perception.

The main contributions of this work can be summarised as the following:

- A hybrid algorithm for APC, performing better than the usually applied for this kind of problems, techniques, has been proposed. This algorithm allows the control of more qualitative dimensions that affect users' perception of playlist quality, like coherence and diversity.
- We have extended our previous single-level [32] to a two-level model, that first identifies appropriate music styles to complete a new playlist and then recommends relevant song sequences. Additionally, we have incorporated a conceptual pre-filtering to implicitly capture the context and more cognitive aspects of playlists [12].
- We set the basis for a methodology that could be used for multimedia recommendations without being so heavily affected by the semantic gap, as does not require a mapping of user queries to song characteristics or vice versa. In addition, as the implemented algorithm addresses similar concepts it does not require well defined user profiles to perform well.
- We proposed and tested additional evaluation metrics that could be used for APC and similar domains.

The incorporation of more quality dimensions into the recommendation algorithm is being evaluated. Songs' novelty and long tail support, that are considered as parameters able to increase users' excitement [21], are among the candidate parameters. In addition, the design of proper user experiments, to capture the real perception of the recommended playlists, is among our future plans. Finally, additional ways to efficiently extract songs' music styles from sparse, and especially user generated, tags will be analysed.

Acknowledgment

This work has been partially supported by the Catalan Agency for Management of University and Research Grants (AGAUR) (2017 SGR 574), by the European Regional Development Fund (ERDF), through the Incentive System to Research and Technological development, within the Portugal2020 Competitiveness and Internationalization Operational Program –COMPETE 2020– (POCI-01-0145-FEDER-006961), and by the Portuguese Foundation for Science and Technology (FCT) (UID/EEA/50014/2013).

References

[1] M. A. Casey, R. C. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney, "Content-based music information retrieval: Current directions and future challenges," *Proceedings of the IEEE*, vol. 96, no. 4, pp. 668–696, 2008.

[2] M. Schedl, P. Knees, and F. Gouyon, "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*, 2017, pp. 392–393.

[3] Ò. Celma and P. Lamere, "If you like radiohead, you might like this article," *AI Magazine*, vol. 32, no. 3, p. 57, oct 2011.

[4] M. Schedl, P. Knees, B. McFee, D. Bogdanov, and M. Kaminskas, "Music recommender systems," in *Recommender Systems Handbook*, 2015, pp. 453–492.

[5] G. Bonnin and D. Jannach, "Automated generation of music playlists: Survey and experiments," *ACM Comput. Surv.*, vol. 47, no. 2, pp. 26:1–26:35, 2014.

[6] M. Schedl, H. Zamani, C. Chen, Y. Deldjoo, and M. Elahi, "Current challenges and visions in music recommender systems research," *CoRR*, vol. abs/1710.03208, 2017.

[7] P. Castells, N. J. Hurley, and S. Vargas, "Novelty and diversity in recommender systems," in *Recommender Systems Handbook*, 2015, pp. 881–918.

[8] M. Ge, C. Delgado-Battenfeld, and D. Jannach, "Beyond accuracy," in *Proceedings of the fourth ACM conference on Recommender systems - RecSys 1́0*. ACM Press, 2010.

[9] M. Schedl, "Investigating country-specific music preferences and music recommendation algorithms with the lfm-1b dataset," *International Journal of Multimedia Information Retrieval*, vol. 6, no. 1, pp. 71–84, Mar 2017.

[10] Y. E. Kim, E. M. Schmidt, R. Migneco, O. G. Morton, P. Richardson, J. Scott, J. A. Speck, and D. Turnbull, "Emotion recognition: a state of the art review," in *11th International Society for Music Information and Retrieval Conference*, 2010.

[11] A. Gatzioura, "A hybrid approach for item collection recommendations: An application to automatic playlist continuation," Ph.D. dissertation, 11 2018.

[12] A. Gatzioura, M. Sànchez-Marrè, and A. M. Jorge, "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*, 2018.

[13] R. López de Mántaras, "Case-based reasoning," in *Machine Learning and Its Applications, Advanced Lectures*, 2001, pp. 127–145.

[14] F. Ricci, L. Rokach, and B. Shapira, "Introduction to recommender systems handbook," in *Recommender Systems Handbook*. Springer US, oct 2010, pp. 1–35.

[15] J. Golbeck and D. L. Hansen, "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*, 2011, pp. 35–42.

[16] F. Maillet, D. Eck, G. Desjardins, and P. Lamere, "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*, 2009, pp. 345–350.

[17] B. Logan, "Music recommendation from song sets," in *ISMIR 2004, 5th International Conference on Music Information Retrieval, Barcelona, Spain, October 10-14, 2004, Proceedings*, 2004.

[18] P. Adamopoulos and A. Tuzhilin, "On unexpectedness in recommender systems: Or how to better expect the unexpected," *ACM TIST*, vol. 5, no. 4, pp. 54:1–54:32, 2014.
[19] M. Kaminskas, I. Fernández-Tobías, F. Ricci, and I. Cantador, "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*, 2012, pp. 19–24.
[20] B. Shao, M. Ogihara, D. Wang, and T. Li, "Music recommendation based on acoustic features and user access patterns," *IEEE Trans. Audio, Speech & Language Processing*, vol. 17, no. 8, pp. 1602–1611, 2009.
[21] Ò. Celma and P. Cano, "From hits to niches?" in *Proceedings of the 2nd KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition - NETFLIX 0́8*. ACM Press, 2008.
[22] D. Jannach, L. Lerche, and I. Kamehkhosh, "Beyond "hitting the hits": Generating coherent music playlist continuations with the right tracks," in *Proceedings of the 9th ACM Conference on Recommender Systems*, ser. RecSys '15. New York, NY, USA: ACM, 2015, pp. 187–194.
[23] C. Baccigalupo and E. Plaza, "Case-based sequential ordering of songs for playlist recommendation," *Advances in Case-Based Reasoning*, pp. 286–300, 2006.
[24] A. Andric and G. Haus, "Automatic playlist generation based on tracking user's listening habits," *Multimedia Tools Appl.*, vol. 29, no. 2, pp. 127–151, 2006.
[25] R. Ragno, C. J. C. Burges, and C. Herley, "Inferring similarity between music objects with application to playlist generation," *Proceedings of the 7th ACM SIGMM international workshop on Multimedia information retrieval MIR 05*, pp. 73–80, 2005.
[26] M. Pichl, E. Zangerle, and G. Specht, "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*, 2015, pp. 1360–1365.
[27] B. Ferwerda, M. Schedl, and M. Tkalcic, "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.*, 2015.
[28] Negar Hariri, Bamshad Mobasher, and Robin D. Burke. Context-aware music recommendation based on latenttopic sequential patterns. In *Sixth ACM Conference on Recommender Systems, RecSys '12, Dublin, Ireland, September 9-13, 2012*, pages 131–138, 2012.
[29] S. J. Morrison and S. M. Demorest, "Cultural constraints on music perception and cognition," in *Cultural Neuroscience: Cultural Influences on Brain Function*, ser. Progress in Brain Research, J. Y. Chiao, Ed. Elsevier, 2009, vol. 178, pp. 67 – 77.
[30] M. Richter and R. Weber, *Case-based reasoning: a textbook*, 09 2013.
[31] B. Ferwerda, M. Tkalcic, and M. Schedl, "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*, 2017, pp. 285–288.
[32] A. Gatzioura and M. Sànchez-Marrè, "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*, 2017, pp. 242–247.
[33] D. Bogdanov and P. Herrera, "How much metadata do we need in music recommendation? a subjective evaluation using preference sets." 01 2011, pp. 97–102.
[34] A. Gatzioura and M. Sànchez-Marrè, "A case-based recommendation approach for market basket data," *IEEE Intelligent Systems*, vol. 30, no. 1, pp. 20–27, 2015.
[35] D. Blei, "Probabilistic topic models," vol. 55, pp. 77–84, 4 2012.
[36] G. Adomavicius and Y. Kwon, "Improving aggregate recommendation diversity using ranking-based techniques," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 5, pp. 896–911, 2012.
[37] G. Bonnin and D. Jannach, "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*, 2013.
[38] I. Kamehkhosh and D. Jannach, "User perception of next-track music recommendations," in *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*, ser. UMAP '17. New York, NY, USA: ACM, 2017, pp. 113–121.
[39] L. Shi, "Trading-off among accuracy, similarity, diversity, and long-tail," in *Proceedings of the 7th ACM conference on Recommender systems - RecSys 1́3*. ACM Press, 2013.
[40] J. Borràs, A. Moreno, and A. Valls, "Diversification of recommendations through semantic clustering," *Multimedia Tools and Applications*, vol. 76, no. 22, pp. 24 165–24 201, Nov 2017.
[41] D. Blei, A. Ng, and M. Jordan, "Latent dirichlet allocation," vol. 3, pp. 993–1022, 3 2003.
[42] J. Vinagre, A. M. Jorge, and J. Gama, "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*, 2014, pp. 459–470.

Anna Gatzioura is a researcher at the Knowledge Engineering & Machine Learning Group (KEMLG) at Universitat Politècnica de Catalunya. She obtained both the BSc. and MSc. degrees from the School of Electrical and Computer Engineering of the National Technical University of Athens and a PhD in Artificial Intelligence from the UPC. Her research interests focus on recommender systems and user preference modelling, and especially on music and playlist recommendations and beyond accuracy dimensions in them.

João Vinagre is a researcher at the Laboratory of Artificial Intelligence and Decision Support (LIAAD) of INESC TEC and invited professor at the University of Porto. He obtained his PhD from the MAPi joint doctoral programme between the Universities of Minho, Aveiro and Porto. His research focuses on recommender systems and user modeling, with special emphasis on stream-based algorithms, evaluation issues and incremental ensemble models. He has published several papers on these subjects and co-organized the ORSUM workshop, held at The Web Conference 2018.

Alípio Mário Jorge is an associate professor of the University of Porto, head of the Computer Science Department and coordinator of LIAAD (Artificial Intelligence and Decision Support Lab) at INESC TEC. He is PhD in Computer Science by U. Porto and MSc. by the Imperial College. His research interests are Data Mining, Machine Learning, in particular recommender systems and information extraction. He co-chaired international conferences such as ECML/PKDD 2015 and 2005, Discovery Science 2009 and EPIA 01.

Miquel Sànchez-Marrè received the BSc. and MSc. degrees in Computer Science both from the Barcelona School of Informatics and a PhD in Computer Science from the Universitat Politècnica de Catalunya. He is Associate Professor in the Computer Science Department of UPC. His main research topics are case-based reasoning, machine learning, intelligent decision support systems, recommender systems, dynamic learning, knowledge engineering and integrated AI architectures. He is a member of the Editorial Board of IJAI and an Associate Editor of the Environmental Modelling & Software journal. He is a Fellow of the iEMSs since 2005.