A Case-Based Reasoning Framework for Music Playlist Recommendations

Anna Gatzioura
Department of Computer Science
Universitat Politècnica de Catalunya · BarcelonaTech
Barcelona, Spain
gatzioura@cs.upc.edu

Miquel Sànchez-Marrè
Department of Computer Science
Universitat Politècnica de Catalunya · BarcelonaTech
Barcelona, Spain
miquel@cs.upc.edu

***Abstract*—Recommender Systems have recently become a fundamental part of various applications intending to support the users when searching for information, products and services that could be interested in at a given moment. While the majority of the current Recommender Systems generate *isolate item* recommendations based on user-item interactions, mainly expressed through ratings, this paper presents a system generating recommendations of *sets of items*. The objective of this system is to generate accurate recommendations in domains where users are looking for a more complete experience, in terms of joint item selections, and where the complexity of the related information cannot be evaluated and treated efficiently using the actual systems, like in the music playlist recommendations. In order to identify the underlying relations between songs, their styles and the artists that perform those, a hybrid Case-Based Reasoning approach combined with a graph model is used. This framework was designed with the scope to overcome the semantic gap present in multimedia recommendations, and at the same time, to be able to perform better in cold start situations. The experimentation results presented, show that the proposed approach is able to deliver recommendations of equal and higher accuracy than some of the widely used recommendation methods.**

Keywords—Case-Based Reasoning; Recommendation Systems; Recommendations of Sets of Items; Graph-based Similarity; Music Recommender Systems; Playlist Recommendations

I. Introduction

Recommender Systems (RSs) have recently become a fundamental part of various applications intending to support users when searching for products and services that could be interested in at a given moment, while helping them to overcome the existing information overload. Especially, in complicated application domains, like in multimedia recommendations, due to the semantic gap between the high level descriptions of user requirements and the low level specifications of the traded items, these systems seem as promising solutions to better capture users' needs and ensure their satisfaction, [12].

The majority of current recommender systems generate recommendations of *single items* that are presented as a ranked list of the top items that most probably will be liked by a user. These systems focus mainly on isolated item characteristics or user ratings and tend to recommend items similar to those liked in the past by the user (Content Based, CB), or by similar users (Collaborative Filtering, CF) based on the assumption that user preferences remain stable over time, [16].

However, in many situations, more than simply predicting whether a single item will or not be liked, the underlying structure of *joint item selections* should be evaluated in order to capture the presence or absence of an item within a concrete concept, overcome the existing semantic gap and recommend *sets of items* designed to be used together, being at the same time relevant, coherent and varied. The current techniques 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. Furthermore they show a limited performance in cases where the probability distribution of the selected items is not equal, like in cases of recommendations of sets or sequences of items. Such an example is the domain of music playlists generation, where the set of songs selected to be reproduced together affects the final result and may depend both on content and contextual data, [12].

In this paper, we present a Recommendation Framework for music playlists recommendations with the intention to better address the above issues. Given that the active user has already selected some songs, the purpose is to identify and recommend the set of songs that are more convenient to complete his/her playlist while enabling users' access to relevant novel items, in order to deliver them a more exciting user experience. Our approach tries to identify the specifications of the items selected together, enabling this way new or less popular items that have the desired specifications to be discovered. However, the presented methodology is not restricted to the playlist recommendations and can be easily extended to serve similar domains characterized by rich content where the importance is placed on the *items selected together* under certain circumstances.

In the next section, we present a brief overview of music and playlist recommendation techniques and their common limitations, followed in section 3 by the presentation of our system, the items modeling and the playlists generation, while in section 4 the evaluation of this framework can be found, as well as its comparison with some of the currently used techniques, showing that our methodology is able to improve recommendations' accuracy. Finally, we conclude and present some of the issues that form our ongoing and future work.

II. Related Work

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 sold and shared online, as well as information about them publicly available, [3]. However, this has also created numerous issues related to music access, discovery, navigation, sharing and information services that need to be facilitated, both for users searching for music and for artists that want to promote their music. This way, like in other domains that face information overload problems, music recommender systems have evolved with aim to handle the information overload related to music, enabling users in finding new music items, or accessing specific music they are looking for and may have difficulties in discovering.

Music recommender systems have their basis both in fields of recommender systems and classical multimedia information retrieval, therefore, although being used in a variety of applications, they still face the common limitations of both areas, [8]. One of the main issues in music information retrieval, as in multimedia information retrieval in general, is the semantic gap between the items' content description and the human perception of these items. Traditional music information retrieval techniques use content audio analysis and low level signal characteristics and lack of ability to capture other music aspects that are closer to the human perception. Therefore, using CB recommendation techniques apart from the general limitations, like the overspecialization and limited diversity, requires a deeper knowledge of the application domain and is more exposed to the semantic gap [3], [15]. On the other hand, using CF techniques to predict the ratings a user would assign to a previously unknown song, based on the ratings of similar users, like in other application domains, comes with cold-start, data-sparsity and long tail effect limitations, [7].

Currently more and more sites incorporate some music reproduction into their environment, or focus purely on presenting music sets, therefore as another interesting issue in the music recommendation domain has emerged the construction and recommendation of music playlists.

A music playlist is a collection of music items presented in a meaningful sequence. In these cases, apart from the characteristics of each item, the set of the recommended items must have some characteristics as a whole, like *variety* and *coherence*. 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 [1], [10].

However, as there are no solid methods combining user's 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 those should have, in order to compose a pleasant and satisfactory result for the listener, [15].

Bonnin and Jannach in [2] review and compare the mostly used methods for automatic playlist generations and recommendations. 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 to find tracks with similar musical features, 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, the authors propose two popularity-based recommendations approaches that also include some artist information.

The authors in [11] 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 sets together with higher frequency have some common characteristics, like a particular genre, pairwise suitability, relative popularity etc. They model the transmissions between musical items based on a graph representation, where adjacent songs are represented as nodes and each arc has weight equal to the number of times that this transition was observed. The resulting graph is transformed into a Markov random field where a playlist can be generated as a random walk starting from a given song (node) and using the Markov transition probabilities.

Baccigalupo and Plaza in [1], present an interesting Case-Based Reasoning approach to music playlists' recommendation with aim to generate playlists of a desired length, being both varied and coherent. Every playlist is treated as a case and their relevance is computed based on their songs co-occurrences and a recommendation is formed as a combination of the items in the most similar lists. The authors also analyze the properties that may bias the effectiveness of their approach, namely songs' popularity and sub-lists' length.

As the context within which a music item is consumed or a playlist is generated is of high importance, Domingues et al. [4] present an interesting approach of incorporating the contextual parameters within the recommendation model. More than performing a pre or post filtering based on the actual context, they represent it as "virtual items". Their results show that these additional dimensions are able of improving the recommendations' accuracy when combined with the usual recommendation algorithms. In addition this contextual modeling may also enable the access to less popular or novel but however relevant to the active user, items, [5].

Finally a slightly different approach to music recommendations is presented by Rosa et al. [13]. The authors associate music songs with users' sentiments as these can extracted from the users' posts in their social networks by using lexicon-based sentiment metrics. However, these last papers focus on song recommendations and would possibly

need to be reformulated or extended in order to be used for playlist recommendations.

III. System Description

The aim of the implemented system is the automatic generation and recommendation of music playlists, more specific, the *recommendation of sets of music items*, being able to complete an initial playlist that a user has selected, more than recommending single items, songs or artists based on a seed song or artist correspondingly.

To this direction, a hybrid Case-Based Reasoning (CBR) system was designed in order to capture the similarity among playlists based on songs', artists' and song styles' distributions in these lists. Having analyzed the existing lists and the songs', music styles' and artists' co-occurrences in those, when a new sub-list is introduced it is first matched over the existing ones, in order to find the most similar ones, and from those to reveal the most adequate way to complete the new list. In order to perform this matching, first of all, the music items have to be described through an adequate model that would permit their comparison without the need of domain specific knowledge. According to the scope of the system and the intention to overcome the semantic gap of multimedia recommenders, a graph-based model was selected as the more adequate way to model the songs and their connections through their common attributes and their common appearances in playlists. This modelling of songs and playlists as well as the recommendation process, are described in more detail in the following sections.

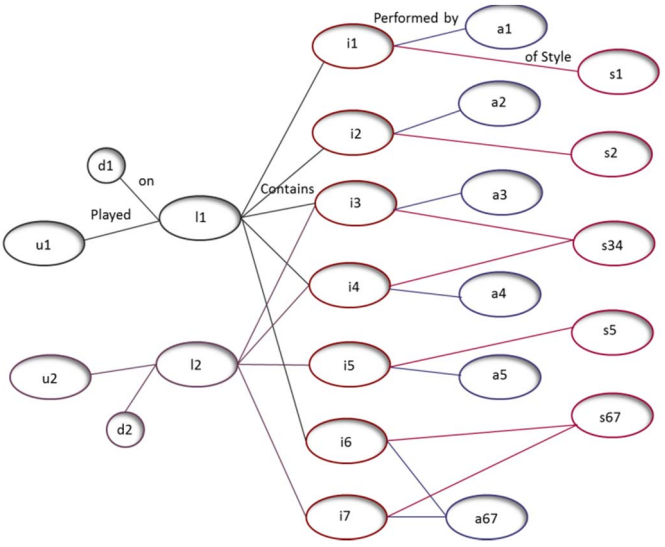
A. Recommendation Model

In contrast to most Case-Based item recommenders that in the music domain would model songs (or artists) as cases, with their attributes forming the case description, the proposed recommendation methodology models each playlist as a case. Given that a user has selected an initial sequence of songs we intend to recommend the set of songs that seem most appropriate for completing this new playlist. Our methodology uses Case-Based Reasoning in order to identify the most similar past sequences and based on those to generate the recommendations for the new list.

As most Case-Based Recommenders, our process is based on the core CBR concepts of similarity and retrieval. CBR works based on the assumption that *"Similar problems have similar solutions"*, therefore in order to find the optimal solution for a new problem first identifies and *retrieves* the most similar past cases in order to *re-use* the way that these were resolved and find possible solutions for the new case, [9]. Generally cases in CBR are modelled as ordered pairs $c = (p, q)$ with $p \in P$ being the problem description and $q \in Q$ the problem solution part, with $p \cap q = \emptyset$. The problem case base that contains all the previously examined cases can be modeled as $C = (P, Q)$.

In the music recommendation domain, let $I = \{i_1, i_2, \dots, i_l\}$ be the set of the l music items, songs, that are stored in a music database and each of them described with its name and unique id, associated with the artist who performs it, as well as with a set of metadata defining the song's style, like the category

Fig. 1. Graphic data representation



(ex: rock), subcategory (ex: alternative), the tempo (low/mid/up), the lyrics language, the vocal (male, female, duet) etc.

A playlist is a set of j songs reproduced one after the other during the same music session, it may be a Dj set, a podcast etc., and is modeled as $l = \{i_1, i_2, \dots, i_j\}$, [1], [2]. Therefore, using the common notion of cases' in CBR, we model a music playlist as a case $l \equiv c$, where for the recommendations generation the problem description part would be the set of the already selected songs $p = \{i_{i1}, \dots, i_{ij}\}$ and the solution $q = \{i_{il}, \dots, i_{i(l+n-1)}\}$ the set of the n songs that will be recommended to complete the initial list. Finally, the problem case base C contains all the previously generated playlists.

The problem data is associated like in Fig. 1, were we have a set a set of playlists $L = \{l_1, \dots, l_k\}$ each containing some songs $l = \{i_1, \dots, i_m\}$ each of which is associated with its style and the artist that it is performed by. By style we refer to the set of metadata associated with the song, other than the artist. Usually in playlists may appear songs of the same style but not many songs of the same artist. Furthermore each list is associated with the time moment and the user by whom was created. However, playlists are not restricted to specific users thus the developed system can generate accurate recommendations for both new and existing users with similar requirements.

The similarity between two playlists, being referred to as *global similarity* is computed as the aggregated pairwise similarity of the songs included in them, being referred to as *local similarity*. As we base our analysis on music item co-occurrences we regard all the items as attributes of a case having the same effect on the user's final satisfaction. Thus, equal importance is placed on the songs included in playlists.

Given two playlists, a new $l_N = \{i_{N1}, \dots, i_{Nn}\}$ with length n_N and a retrieved $l_R = \{i_{R1}, \dots, i_{Rm}\}$ with the *i-th* attribute of each of them, the *i-th* song, being i_{Ni} and i_{Ri} correspondingly, that have local similarity equal to $sim(i_{Ni}, i_{Rm})$, the global similarity $Sim(l_N, l_R)$, of these two cases can be calculated as:

$$Sim(l_N, l_R) = \sum_{i=1}^{l} maxsim(i_{Ni}, i_{Rm}) / n_N \quad (1)$$

Where, $maxsim(i_{Ni}, i_{Rm})$ is the maximum local similarity value found among the song i_{Ni} in the new list and the songs included in the retrieved list. As it can be seen from equation (1) above, where the sum of local similarity values is averaged by the length of the new list, the used similarity metric is not symmetric, meaning that for two lists l_a and l_b we may have $Sim(l_a, l_b) \neq Sim(l_b, l_a)$.

The global similarity value heavily relies on the local similarities of the items included in the different cases, therefore the item model used for the songs and their similarity calculation will be explained in more detail below.

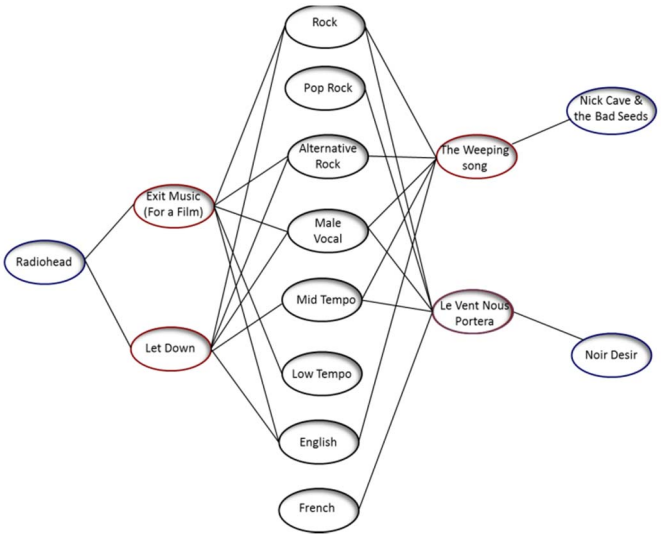
B. Item Model

The performance of the actual recommender systems may be heavily affected by the data sparsity and the long-tail distribution of the items in this domain, which is the result of the huge amount of the existing items, among which very few are finally used and even less evaluated by the users. As in the current problem we don't have user evaluations of songs, we infer the placement of a song within a playlist as a notion of preference of a user towards this song, this style and this artist. However we cannot assume that not listening to song is always a result of not liking it, especially when referring to a song closely similar to the song that the user has liked it is very probable that the user may not be aware of that song. This is the reason why our intention is to abstract from the exact songs that were placed into a given playlist and to place our emphasis on the characteristics and the style of songs that were placed within a specific playlist, on a given moment (*Semantic generalization*) in order to better capture user preferences and playlist preferred structures. This approach highlights an important difference of our framework with the usually applied techniques in domains where item co-occurrences are analyzed, like the association rules (ARs) mining, which is that these techniques evaluate the presence or absence of an item within a set, while our method evaluates the similarity among the items in sets.

To this direction arises the need of a proper *categorization* of the available music items so that we could approximate their similarity without using low level descriptions (like bits per minute, tonality and other sound related metrics) and computationally expensive, domain specific metrics. Therefore, songs first have to be transformed into a convenient representation that would enable their categorization into smaller groups of almost identical items, based on the notion of some relationship based model.

Initially, in our previous work on the market basket analysis, a hierarchical, a tree based, model that starts from the general item concept and at each hierarchical level groups items having common ancestors, was used, [6]. However this representation requires some domain knowledge or an adequately defined item database from which we could deduct the relative importance of the various item characteristics. Such a classification is not applicable in domains like music, and multimedia in general, where items are associated with metadata, usually expressed through tags, where their relative importance cannot be decided a priori without having the necessary domain knowledge. In addition this importance may depend on subjective factors related to each user's perception of music and the scope of each system. Therefore after analyzing the possible alternatives, a graph-based model that associates songs through their common attributes without evaluating their exact values, was selected in order to model the songs and to calculate song and style similarities, [14].

Fig. 2. Example of the items' tag graph



We model songs through their tags that represent the metadata related to the songs music style (category, subcategory, tempo, language, vocal etc.) and the artist that a song is performed by as follows,

$$i = \{id, style, artist\} = \{id, \{t_1, \dots, t_l\}, artist\} = \{t_1, \dots, t_n\}$$

We use the term *level of abstraction* to specify the quantity of details that are included in the item description when modeling them. Depending on the application scope we may wish to specify the song styles or the artists that best fit within a given playlist, therefore we would treat as "identical" all the items having the same style etc.

The similarity of two songs can be thus calculated based on the density of their common terms. Given two songs $i_a = \{t_{1a}, \dots, t_{na}\}$ and $i_b = \{t_{1b}, \dots, t_{nb}\}$ that have $n(a \cap b)$ number of tags in common, while $n(a/b)$ is the number of tags associated only with song a and $n(b/a)$ is the number of those associated only with song b, we can calculate their similarity using the following function:

$$sim(i_a, i_b) = 1 - \log_2(1 + \frac{n(a/b)+n(b/a)}{n(a \cap b)+n(a/b)+n(b/a)}) \quad (2)$$

In Fig. 2 we present an example, a subgroup, of the songs graph, where we have "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, and their connections through the tags they have in common. Based on this figure and the tags that each song is associated, these songs would be described as:

$$Let\ Down = \{Let\ Down, Rock, Alternative\ Rock, English, Male, Mid\ Tempo, Radiohead\}$$

$$Exit\ Music(For\ a\ Film) = \{Exit\ Music, Rock, Alternative\ Rock, English, Male, Low\ Tempo, Radiohead\}$$

$$The\ Weeping\ Song = \{The\ Weeping\ Song, Rock, Alternative\ Rock, English, Male, Mid\ Tempo, Nick\ Cave\&\ the\ Bad\ Seeds\}$$

$$Le\ Vent\ Nous\ Portera = \{Le\ Vent\ Nous\ Portera, Rock, Pop\ Rock, Male, French, Mid\ Tempo, Noir\ Desir\}$$

As we can see "Let Down" and "The Weeping Song" have the same tags, therefore they would be categorized as being of the same music style but are performed by different artists, while on the other hand we can see that "Let Down" and "Exit Music (For a Film)" both performed by Radiohead have very similar characteristics although one is characterized as mid and the other as low tempo. Finally, the lyrics of "Le Vent Nous Portera" are in French while the style, although being similar to the previous ones, is a little more pop. However, the degree of similarity of these items also depends on the level of abstraction that will be used.

IV. Testing and Evaluation

An initial evaluation of the proposed framework has taken place using a music database containing approximately 3000 playlists of maximum length of one hour, made using approximately 3500 rock songs. In specific, we used 3452 songs performed by a total of 833 artists. These songs were associated with the style attributes that permitted them to be modeled through 114 styles or when using the combination of music style and artist we have 1654 labeled styles.

We performed various recommendations, splitting each time the database into a training and testing part (80% and 20% correspondingly), where the training part is used as the case base of the problem, the past playlists, whereas the testing part forms the set of new playlists that have to be completed. From the playlists in the testing part we extracted sublists of different length (of 3, 5, 7 and 9 music items long) in order to generate playlist recommendations of the same length and evaluate the capability of the methods to identify the missing items. The systems were also tested on providing recommendations of different levels of abstraction, specifically for recommending the exact items, or their abstractions through their music style, or the artist that performs them or their labeled style that combines their style and artist description. For example, when modeling items through their music styles we generate recommendations of the styles that best fit into a given music list that a specific user has started building.

When both the style and the artist name are used in the song modeling we have a more accurate description of the music items but the complexity of the problem increases as in each playlist many songs of the same or similar styles may appear but not many songs performed by the same artist. When recommending song styles independently of the artist that performs them the accuracy of all methods increases highly. Identifying the preferred song styles that fit within a playlist is important in order to identify the general playlist profile and user general preferences but in some cases it may be not sufficient.

Finally, instead of songs we may recommend artists whose songs the active user would like to place into the current playlist. Generally each artist has a personal style that can be defined as the distribution of styles of the songs in his/her discography. Some artists may have really diverse styles but the majority maintains some characteristics throughout their discographies. Therefore identifying the artist that a user would select is also a way of identifying the style of the songs he/she wishes to have in his/her playlist.

We evaluate our method, the so called MusCBR, in terms of recommendations' accuracy and compare it with two baseline recommendation methods, a Collaborative Filtering (CF) and an Association Rules-based (ARs). In the case of the CF Recommender the similarity among the users that created the playlists is used. As no ratings are present we used the hypothesis that the placement of a music item in a playlist, may it be a song, a music style or an artist, can be inferred as a preference towards this music item. On the other hand, for the AR recommender, the Apriori algorithm was used for the extraction of the rules based on which the recommendations were generated.

In tables I, II and III the recommendation results' accuracy, in terms of average precision, is presented. Recommendations' 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. Other commonly used measures are recall (the ratio of the number of correct items that are recommended over the total number of correct items) and F1 (the harmonic mean of precision and recall).

TABLE I. AVERAGE PRECISION VALUES WHEN USING SONGS' LABELED STYLE

List Length	Average Precision		
	MusCBR	***CF***	***ARs***
3	**0,084**	0,037	0,026
5	**0,080**	0,030	0,027
7	**0,075**	0,043	0,018
9	**0,079**	0,044	0,014

TABLE II. AVERAGE PRECISION VALUES WHEN USING SONGS' MUSIC STYLE

List Length	Average Precision		
	MusCBR	***CF***	***ARs***
3	0,33	**0,34**	0,21
5	**0,39**	0,34	0,31
7	**0,41**	0,37	0,40
9	0,40	0,34	**0,42**

TABLE III. AVERAGE PRECISION VALUES WHEN USING ARTISTS

List Length	Average Precision		
	MusCBR	*CF*	*ARs*
3	**0,090**	0,044	0,036
5	**0,096**	0,067	0,064
7	**0,113**	0,094	0,10
9	**0,123**	0,117	0,121

As it can be seen from the previous tables our methodology is able to perform equally and even better than the compared recommenders, even when using more specific item descriptions. When recommending unique items or when using their labeled styles we face an increased data sparsity, given that in each playlist there is a small number of songs of the same artist, therefore the problem complexity is almost the same. In these cases the AR methodology is 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. (We do not present the results for the recommendations on songs' unique ids as all the algorithms show a weak performance, with MusCBR reaching the highest, around 5%).

v. Conclusions and Future Work

In this paper we have described a hybrid Case-Based Reasoning approach for music playlists generation and recommendations. The implemented system recommends sets of songs, artists or item styles that are more probable to fit into a given playlist that a user has started.

In order the overcome the semantic gap of music information retrieval systems and to perform better in cold start situations, the system does not use user ratings or sound related characteristics of songs to generate recommendations. The presented methodology combines Case-Based Reasoning with a graph-based model for describing the songs that are connected through the tags that define their style and the playlists that they form part of, in order to identify their similarity and capture the co-occurring patterns and the user preferences in terms of sets of music items enjoyed together.

The initial experimentations have shown that the proposed methodology is able to perform better than two of the state of the art recommendation techniques, namely a Collaborative Filtering and an Association Rule-based one, even for more abstract data and seems to be able to provide a solid basis for the recommendations of relevant music playlists.

Our intention is to extend this framework in order to provide additional support to novel recommendations, as items' novelty is considered of high importance for the user experience. In addition, from our ongoing research it has been found that the parameter that mostly characterizes similar playlists is the time moment at which those were performed. Therefore, the incorporation of contextual data into the system forms among the principal issues of our future work. Finally more complexed similarity metrics will be tested, in order to incorporate and evaluate correctly the new parameters.

References

[1] C. Baccigalupo and E. Plaza, "Case-Based Sequential Ordering of Songs for Playlist Recommendation," Advances in Case-Based Reasoning 4106, p. 286-300, 2006.

[2] G. Bonnin, and D. Jannach, "A comparison of playlist generation strategies for music recommendation and a new baseline scheme," Workshops at the twenty-seventh AAAI conference on artificial intelligence, 2013.

[3] M. A. Casey, R. Veltkamp, M. Goto, M. Leman, C. Rhodes and M. Slaney, "Content-Based Music Information Retrieval: Current Directions and Future Challenges," Proceedings of the IEEE, 96. p. 668 – 696, 2008.

[4] M. A. Domingues, A. M. Jorge, and C. Soares, "Using contextual information as virtual items on top-n recommender systems," arXiv preprint arXiv: 1111.2948, 2011.

[5] M. A. Domingues, and S. O. Rezende, "The impact of context-aware recommender systems on music in the long tail," Intelligent Systems (BRACIS), 2013 Brazilian Conference on, IEEE, p. 119-124, 2013.

[6] A. Gatzioura. and M. Sànchez-Marrè, "A Case-Based Recommendation Approach for Market Basket Data," Intelligent Systems, IEEE, vol. 30, no. 1, p. 20-27, Jan-Feb 2015.

[7] M. Kaminskas and F. Ricci, "Contextual Music Information Retrieval and Recommendation: State of the Art and Challenges," Computer Science Review 6, p. 89-119, 2012.

[8] E. Y. Kim, M. E. Schmidt, R. Migneco, G. B. Morton, P. Richardson, J. Scott, A. J. Speck and D. Turnbull, "Music Emotion Recognition: A State of the Art Review," Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR 2010), p. 255-266, 2010.

[9] R. López de Mántaras, "Case-Based Reasoning," Machine Learning and its Application, vol. 2049, pp.127-145, 2001.

[10] F. Maillet, D. Eck, G. Desjardins, P. Lamere, "Steerable Playlist Generation by Learning Song Similarity from Radio Station Playlists," Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR 2009), p. 345-350, 2009.

[11] 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, p. 73-80, 2005.

[12] F. Ricci, L. Rokach, ans B. Shapira, "Introduction to recommender systems handbook," p. 1-35, springer US, 2011.

[13] R. L Rosa, D. Z. Rodriguez, and G. Bressan, "Music recommendation system based on user's sentiments extracted from social networks," IEEE Transactions on Consumer Electronics, 61(3), p. 359-367, 2015.

[14] D. Sánchez, M. Batet, D. Isern and A. Valls, "Ontology-based semantic similarity: A new feature-based approach," Expert Systems with Applications, vol. 39, Issue 9, p. 7718-7728, July 2012.

[15] B. Shao, D. Wang, T. Li and M. Ogihara, "Music Recommendation Based on Acoustic Features and User Access Patterns" IEEE Transactions on Audio, Speech, and Language Processing 17, p. 1602 – 1611, 2009.

[16] X. Su, and T. M. Khoshgoftaar, "A Survey of Collaborative Filtering Techniques," Advances in Artificial Intelligence, p. 1–19, 2009.