Deployment and Evaluation of a Usage Based Collaborative Filtering Recommendation System with Blacklists

Luís Lemos[1], Alípio M. Jorge[2], José Paulo Leal[3]

DCC – FCUP, Universidade do Porto, Portugal
[1] INESC Porto
[2] LIAAD – INESC Porto L.A.
[3] CRACS – INESC Porto L.A.
llemos@inescporto.pt , amjorge@fc.up.pt , zp@dcc.fc.up.pt

Abstract. Social web sites, like Palco Principal (PP), have considerably sized databases with the most varied types of information from user subscription information to user interactions. All that information is susceptible of being processed using data mining algorithms in order to extract knowledge relevant for the business. This document describes the implementation and evaluation of a system that uses data mining techniques, and that has been developed with the specific intent of generating recommendations, namely music recommendations, to users of the PP web site. The implemented system uses collaborative filtering techniques to recommend music additions to a user's playlist. The impact of the deployed system has been evaluated online.

Keywords: collaborative filtering, recommender systems, deployment, online evaluation.

1 Introduction

One of the currently most popular means of interaction used by people are social web sites. In these web sites, and in the particular case of Palco Principal (PP), the web site of the company with the same name, users can share opinions about musical tracks they like or dislike, ear music from listed bands, comment on tracks, read news from the music world, etc. As stated in [3], Palco Principal is a company with a technological basis, founded in 2006, that has developed the web site www.palcoprincipal.com .

PP web site provides a space where musicians, bands and other musical projects can promote themselves. Signing in the web site is free for all. Listeners can listen and download tracks that are usually outside the mainstream circuits. As any company, PP is continuously improving their product, in this case their web site, and they regarded as an important improvement to have a system that would recommend new tracks to their listener users (listeners).

Originally, the recommendation system was planned to be applied to musical tracks but was conceived and implemented as a generic recommendation system so that it can be applied to other items of the PP web site. For example, it can be used to recommend friends or related events. The deployed recommender system uses collaborative filtering techniques [5] to establish relations between musical tracks and creates a cosine Similarity Matrix. When a recommendation is requested, the tracks that are most related, according to the Similarity Matrix, with the ones on the user's playlist are returned. Each user creates his/her playlist or playlists.

This paper describes the generic recommender system, its implementation, deployment and evaluation in terms of performance and impact in the site.

2 Objectives

The main objective was to implement, deploy and evaluate a system that recommends additional music tracks to playlists. As in any project, the first task to be performed is the gathering of requirements as these will help define the development guidelines.

Palco Principal (PP) wanted a black box system that received IDs of tracks, for a given user, and returned the IDs and names of recommended tracks. The track IDs passed on to the system would mostly be of tracks from one of the user's playlist. However, because the system is more general, it works with virtual playlists (i.e. any set of musical tracks and not necessarily just one user's playlist).

Two other requirements of the company were that the system would be able to recommend in real-time, and that it would be easily integrated with their web site which is implemented in PHP [4] and MySQL [2]. For these reasons, it was decided to implement the recommendation system exclusively in SQL with Stored Procedures that are called directly from PP's web page. SQL also proved to be a very efficient approach.

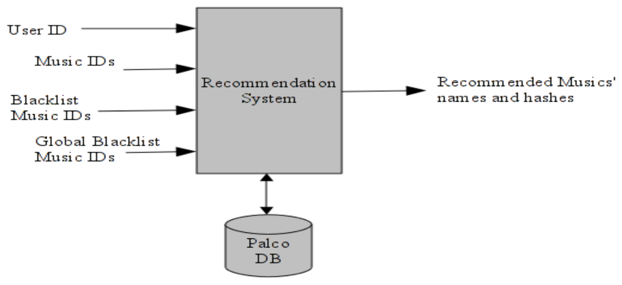


Figure 1: Final Recommendation System Interface

Although the first recommendations returned by the system were considered very good by the representatives of the company, we anticipated a potential problem. Real users could accept recommendations, but could not reject them. This motivated the use of blacklists, which allows users to declare they do not want a specific track to be recommended to them. Thus, unwanted recommendations would not appear again in a later recommendation. Moreover, if many users did not like one specific track then that track would probably not be a very good recommendation. To exploit this information, a global blacklist was conceived to shorten the chances of highly recommending unpopular tracks.

The historical playlist data required to build the recommender model was contained in PP's database. The final system with blacklists, and the obviously required access to PP's DB, is presented in Figure 1.

3 System overview

To provide recommendations for one particular user, the implemented collaborative filtering algorithm uses the playlists of the other users. Two users that have many tracks in common will probably like the same tracks and will be willing to add each other's tracks to their respective playlists. This means that, if users A and B have two identical playlists and A chooses music X to add to his playlist, then it is likely that B also likes music X and would like to have it in his own playlist.

Algorithmically, the recommender establishes a relation between the tracks from the user's playlist for which one wants recommendations, and the tracks in other playlists and creates a Similarity Matrix. The Similarity Matrix contains the degree of similarity between each pair of tracks [5]. Therefore, the only information required from the site's DB is the playlists and their tracks.

Having the Similarity Matrix, it is necessary to get the *N* tracks that have higher weight (higher similarity) for each track on the user's playlist, sort them by weight and return them to the web page to be displayed to the user. That weight is the measure of similarity between one track and the whole playlist.

The system is therefore divided into two major modules. One builds the Similarity Matrix (Model Generation), and the other (Recommender) determines the recommendations from the Similarity Matrix and the active user's musical preference information (playlist, blacklist and global blacklist).

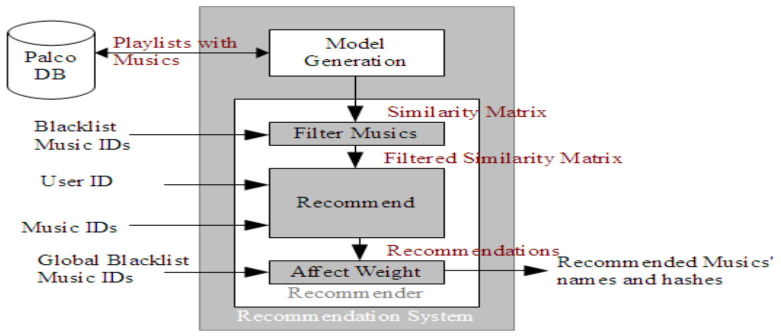


Figure 2: Detailed high level image of functions and interactions

The purpose of blacklists is to influence the results of the recommendations either by excluding tracks using the user's personal blacklist or by changing their weight to a lower value using the global blacklist (thus positioning the track in a lower position on the resulting recommendations set). The global blacklist is just the count of how many users have blacklisted each track. The entire system is presented in Figure 2.

4 System description

The Model Generation module builds a Similarity Matrix that has track IDs indexing lines and columns of the matrix. Each position of the matrix, indexed by row and column, is calculated using the cosine based similarity shown in Formula (1).

$$M_{(i,j)} = \frac{I_{(i,j)}}{\sqrt{D_i * D_j}} \ . \tag{1}$$

Where i indicates the row, j indicates the column, $I_{(i,\,j)}$ is the number of playlists in which both tracks i and j exist. D_i is the number of playlists in which the row track i exists (similarly for D_j). Note that the Model Generation is a computationally intensive algorithm since the Similarity Matrix size is the square of the number of tracks and, as such, will not be executed in real-time. Instead, it will be executed from time to time (Example: Once a day).

The Recommender will be run when a recommendation list is requested for a particular user. It starts by filtering out the personally blacklisted track IDs from the Similarity Matrix. After that, the collaborative filtering algorithm is applied. For each row track remaining in the filtered Similarity Matrix we obtain the top N column IDs with highest similarity. Those Top N are the nearest neighbors of each track. Then, tracks already in the playlist of the user are excluded so that they are not recommended again. After that, for each remaining row track, a set of recommendations is calculated using Formula (2).

$$R_i = \frac{\sum\left(I_{(N_i,m)}\right)}{\sum(N_i)} \ . \tag{2}$$

Where i is the row track, N_i are the neighbors of track i, m are the tracks in the user's playlist, $I(N_i,m)$ are the tracks in the intersection of N_i and m. The numerator is the sum of the weights of the tracks in $I(N_i,m)$ and the denominator is the sum of the weights of all neighbors of track i.

The weight of the tracks to recommend is multiplied by a *Rejection Index* that is calculated, from the global blacklist and all playlisted tracks, using Formula (3).

$$RI_i = 1 - \frac{B_i}{B_i + P_i + 1} \ . \tag{3}$$

Where i is the track ID for which the *Rejection Index* is being calculated, Bi is the number of times that track i has been blacklisted by all users and Pi is the number of times that the same track occurs in playlists. After having their weight affected, the Top K row tracks are recommended.

Since the number of neighbors for each track (the top *N*) is set constant, we can speed up the calculation of the recommendation score by computing offline the neighbors for every track. This will lead to a significant decrease in the time taken to perform recommendations. Moreover, the Similarity Matrix takes much more space than the matrix that stores only the nearest neighbors for each track. The flowchart of the operations is presented in Figure 3.

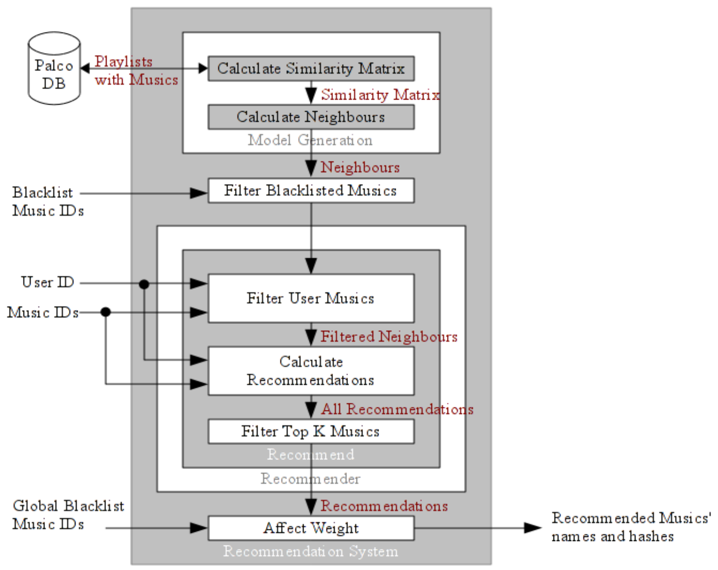
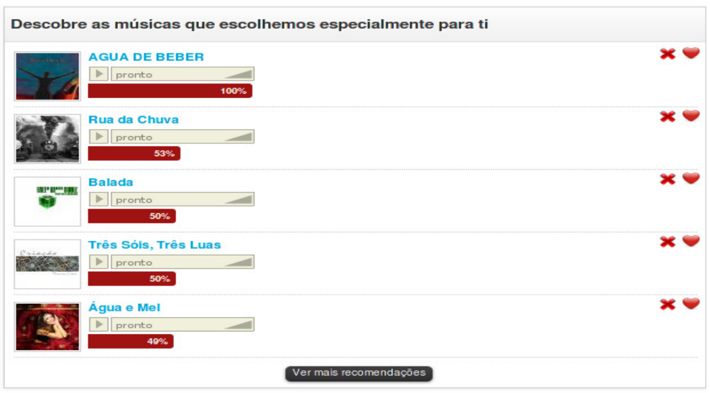


Figure 3: Final system flowchart

The deployed recommender prompts a recommendation listing when a user logs in (Fig. 4). The interface enables listening to the proposed track (button play), adding to the playlist (button heart) and adding to the blacklist (button cross).


Descobre as músicas que escolhemos especialmente para ti

AGUA DE BEBER
pronto
100%

Rua da Chuva
pronto
53%

Balada
pronto
50%

Três Sóis, Três Luas
pronto
50%

Água e Mel
pronto
49%

Ver mais recomendações

Figure 4: Recommendation list presentation.

5 Recommender System Evaluation

When the Recommender System was ready to go online, evaluation methods were devised to gather usage data for a posterior assessment of the impact on PP's web site. The first stage was to divide PP's authenticated user universe into two groups, as in an A/B test [1]. One group would be exposed to the recommendations (the test group) and the other group would keep seeing the web site without recommendations (the control group). This splitting has been done automatically and online through the use of cookies and on the http server. The test period lasted 9 days, from 2010-03-29 to 2010-04-06. After 9 days the company decided to expose every user to recommendations. The 9 days period was too short for definite conclusions on the A/B test. No significant differences on session length were observed. However, through Google Analytics, we could observe a large difference in new additions to the playlists: 310 (test group) against 36 (control group).

Table 1. - Playlist results: new additions.

	Number of added tracks per playlist	Number of playlisted tracks by user	Number of times a music was playlisted
Maximum	166	166	12
Minimum	1	1	1
Average	6,56	6,58	1,36
Median	3	3	1
Standard Deviation	13,72	13,74	0,85
Total	309 playlists	308 users	1491 tracks
Count below Average	235 playlists	234 users	1132 tracks
Count below 50% Maximum	307 playlists	306 users	1487 tracks
Total playlisted tracks	2026	2026	2026

Tables 1 and 2 provide statistics obtained by analyzing directly the playlist database. Here, numbers are much higher since they are for all users, test group and non test group. This is because in the playlist database we do not have the information about which users are testers or non-testers. In an extreme situation, the same user could even move to a different group if he deleted the cookies or logged in from a different computer.

Table 2 reveals that throughout the test period, 64 users added 242 unique tracks to blacklists with a total of 279 blacklisted tracks. From the other fields, it can be concluded that most users added few tracks to blacklists. This can either mean that most users enjoyed most of the proposed recommendations but did not add them to their playlists, or they just tried it and gave up. Also, the number of times each track was added to a blacklist was very small, with most tracks only being added once.

Table 1, for the same period, reveals that 308 users added 1491 unique tracks to their playlists with a total of 2026 tracks added. For a more precise analysis of the impact of the recommendations, and besides the data gathered during the test period, the web site usage data was collected for two additional periods. Before the test period and after the test period. Results are presented in Tables 3 and 4.

Table 2. - Blacklist results: blacklisted tracks.

	number of blacklisted tracks by user	number of times a music was blacklisted
Maximum	43	4
Minimum	1	1
Average	4,36	1.15
Median	2	1
Standard Deviation	7,98	0,43
Total	64 users	242 blacklisted tracks
Count below Average	50 users	211 blacklisted tracks
Count below 50% Maximum	61 users	237 blacklisted tracks
Total blacklisted	279 tracks	279 tracks

Table 3. - Playlisted tracks analysis

Number of playlisted tracks by date:	Prior to recommendation test period	During recommendation test period	After recommendation test period	All time (prior+during+after)
Max	275	297	366	366
median	132.5	243	264	178,5
Min	70	119	168	70
Average	147.5	225,11	270.92	197,57
Standard deviation	55.9	57,75	65.56	79,87
Total playlisted	3540	2026	3522	9088
Total days	24	9	13	46
Linear regression	0.01	-0,01	0.02	0,13

Around 31 tracks have been blacklisted per day and 225 were added to playlists during the test period. Those numbers kept increasing after the test period. The "Linear Regression" coefficient for blacklists reveals a decrease after the test period, which can be regarded as natural after the first impact. The overall growth in activity is, nevertheless, positive. Figure 5 and Figure 6 graphically present the evolution of the number of playlisted and blacklisted tracks over time.

Figure 6 shows that the number of blacklisted tracks increased with time, from the start to the end of the test period. This is somewhat the expected behaviour for a new feature. Figure 5 shows that the number of playlisted tracks was more or less stable but increasing. In conjunction with the information from blacklisted tracks, this might indicate that users did not use the recommendations much to add tracks to their playlists. One could estimate that, if nothing else happens, the number of blacklisted tracks will continue to slowly increase and that the number of playlisted tracks will continue to increase at the same rate it would if there were no recommendations.

Table 4. - Blacklisted music analysis

Number of blacklisted tracks by date:	During recommendation test period	After recommendation test period	All time (during + after)
Max	53	96	96
median	32	45	35
Min	3	13	3
Average	31	44.85	39,18
Standard deviation	15,41	22.68	20,81
Total blacklisted	279	583	862
Total days	9	13	22
Linear regression	0,08	-0.06	0,07

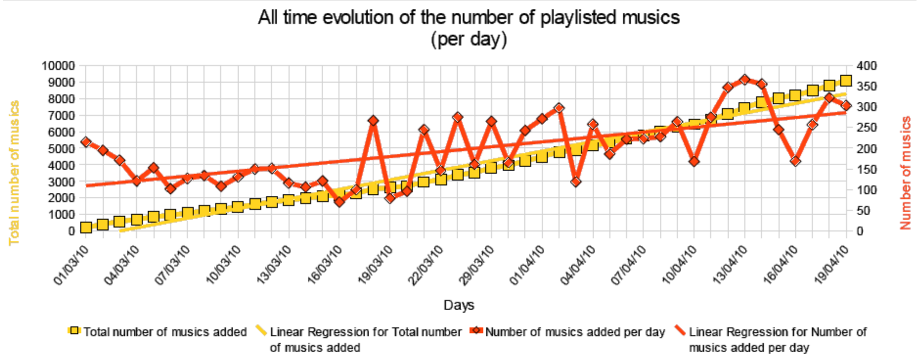


Figure 5: All time evolution of the number of playlisted tracks

A further analysis of the data collected into the DB indicates that of all those tracks added to playlists and blacklists, 330 users added tracks to either their playlists or their blacklists and only 42 users added tracks to both their playlists and blacklists. This means that 22 (64-42) users have added tracks only to their blacklist!

With respect to time taken by the SQL implemented recommender system, response times (recommendation times) are unnoticeable to the users and well under 1 second. Model building time is below 30 minutes for a current database of 38 000 tracks. This enables a frequent model update. The system is running on a single PC. SQL and stored procedures are not very easy to maintain from the point of view of the programmer, but provide very efficient executions and integrate smoothly with the database.

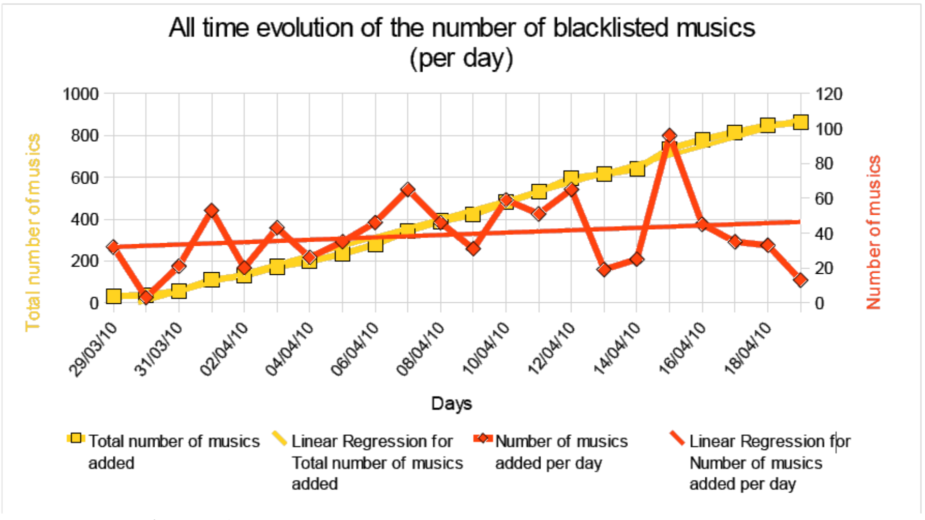


Figure 6: All time evolution of the number of blacklisted tracks

6 Conclusions

We have described the effort for deploying and evaluating a music recommender system on a real web site. The system is currently working online for registered users (free of cost) and responding in real time (www.palcoprincipal.pt). The recommender algorithm is a classical item-based collaborative filtering, extended with blacklists. Users are prompted with recommendation lists upon login and have the options of adding tracks to their playlists, adding tracks to their blacklists or ignoring recommendations. The impact of the activity on playlist addition has been measured using an A/B test, and a before/after analysis. Results strongly indicate an increase in playlist activity in the site generated by the recommender. The blacklist facility is also frequently used which shows the need for blacklists. The recommender system is implemented in SQL and is running since April 9th 2010. Response times are unnoticeable to the users. Model building time enables daily model refresh.

7 Future work

Affecting the tracks' weight with the Rejection Index after the recommendations are calculated is not really the best choice since it does not influence the calculation of the neighbors. It has been implemented like this mainly for computational reasons. Ideally the Rejection Index should affect the weights right after the Similarity Matrix is calculated.

The number of neighbors used during this project was fixed to a number thought to be acceptable (4). This parameter should be fine tuned with experiments. However, a higher number of neighbors will slow down the recommendation procedure.

More care should also be taken in future online experiments. Negotiating live experiments with a company eager to expose all users to a new feature is not easy but is worth trying.

This usage based recommender is currently being combined with content based recommenders to reduce the cold start problem (new tracks are not recommended because they are not in playlists) and to increase the width of the recommendation spectrum. Usage based recommendations tend to be more conservative.

Acknowledgements

We are mostly grateful to the Palco Principal company for their enthusiastic collaboration and enriching feedback, in particular to João Carvalho, Pedro Trindade and Daniel Botelho. This work is being supported by QREN AdI Palco3.0/3121 PONORTE and Multi-Year Funding Program for R&D Units (FEDER).

References

1. Ron Kohavi, Roger Longbotham, Dan Sommerfield, Randal M. Henne: Controlled experiments on the web: survey and practical guide. Data Min. Knowl. Discov. 18(1): 140-181 (2009)
2. MySQL main web page, http://mysql.com/
3. Palco Principal web site, http://www.palcoprincipal.com
4. PHP main web page, http://php.net/index.php
5. Badrul M. Sarwar, George Karypis, Joseph A. Konstan, John Riedl: Item-based collaborative filtering recommendation algorithms. WWW 2001: pp. 285-295