EAI Endorsed Transactions
on Artificial Intelligence

Research Article EAI.EU

Considering Durations and Replays to Improve Music Recommender Systems

Pierre Hanna*

Computer Science Laboratory (LaBRI), University of Bordeaux, 351 Cours Liberation, 33405 TALENCE, France

Abstract

The consumption of music has its specificities in comparison with other media (movies, books), especially in relation to listening durations and replays. Music recommendation can take these properties into account in order to predict the behaviours of the users. Their impact is investigated in this paper. A large database was thus created using logs collected on a streaming platform, notably collecting the listening times. The proposed study shows that a high proportion of the listening events implies a *skip* action, which may indicate that the user did not appreciate the track listened. Implicit *like* and *dislike* can be deduced from this information of durations and replays and can be taken into account for music recommendation and for the evaluation of music recommendation engines. A quantitative study as usually found in the literature confirms that neighborhood-based systems considering binary data give good results in terms of MAP@k. However, a more qualitative evaluation of the recommended tracks shows that many tracks recommended, usually evaluated in a positive way, lead to *skips* or thus are actually not appreciated. We propose the consideration of implicit *like/dislike* as recommendation engine inputs. Our evaluations show that neighbourhood-based engines remain the most precise, but filtering inputs according to durations and/or replays have a significant positive impact on the objective of the recommendation engine. The recommendation process can thus be improved by taking account of listening durations and replays. We also study the possibility of post-filtering a list of recommended tracks so as to limit the number of tracks that will be unpleasantly listened (*skip* and implicit *dislike*) and to increase the proportion of tracks appreciated (implicit *like*). Several simple algorithms show that this post-filtering operation leads to an improvement of the quality of the music recommendations.

Received on 05 July 2018; accepted on 27 October 2018; published on 22 January 2019

Keywords: Music Recommendation, Recommender Systems, Usage Data

doi:10.4108/eai.5-2-2018.156379

1. Introduction

Recommender systems [1] are taking an increasingly important part in the digital applications currently being developed and proposed to the public: e-commerce, news articles, music, video, book, etc. They help users make their decisions by providing lists of items that should be most likely of interest to them. The design of recommendation engines has thus been the subject of numerous scientific researches, compiled in several books [2, 3].

Different types of recommender systems have been developed [4]. Systems based on collaborative filtering are historically the first recommender systems [5]. They take into account the interactions between users and items, in order to highlight associations between users and particular items that these users have not yet consumed. Thus, a recommended item is an item liked by a user with similar tastes. In a different way, content-based systems [6] consider both descriptors of the content of items and positive or negative ratings made by users for these items. By considering similarity based on the content, new items similar to those previously appreciated can be provided to each user. Finally, hybrid recommender engines [7] combine different types of engines to balance the advantages and drawbacks of each technique.

*Email: pierre.hanna@labri.fr

Collaborative filtering approaches can be grouped in the two general classes of neighbourhood and model-based methods. Neighbourhood-based (or memory-based) collaborative filtering methods consider user ratings stored in memory to predict ratings for new users or new items [8]. Two approaches can be applied, taking into account similarities between users (user-based recommendation [9]) or similarities between items (item-based recommendation [10, 11]). These approaches therefore process ratings to compute predictions, while model-based approaches use these ratings to learn a predictive model [12, 13]. This model captures the important elements of relationships between users and items, learning the parameters of the models from an existing dataset.

Whatever the approaches considered, they are therefore based essentially on the ratings of items provided by users. These ratings can be explicit, i.e. directly made by users, or implicit [14], i.e. deduced from the actions of users on items or on the browsing for items . Examples of explicit ratings include the Netflix video streaming platform, which asks for its users to rate the movie they had just watched by providing 1 to 5 stars. This explicit action is binding for some users. For example, Netflix has just suspended its explicit rating method used for several years to propose a less restrictive rating process [1]. The Deezer or Spotify music streaming platforms also allow their users to indicate their satisfaction by clicking on a *thumb-up* or a *thumb-down* on each track listened or to select a track for their *loved track* list. Many existing recommender systems deal with explicit ratings (e.g. Netflix). However, this explicit data requires additional user commitment which may appear to be effortful. As a result, the number of explicit ratings is often limited.

For each user, numerous implicit usage parameters, such as clicking, listening or browsing histories, are considered in order to precisely describe its usage and may thus be taken into account by the recommender systems in order to estimate its preferences. For example, a user who listens to several tracks from the same performer probably likes this performer and may appreciate other tracks from that same performer. Therefore, other existing recommender systems deal with implicit data, such as consumption history or browsing history [15]. It is then necessary to determine the most pertinent descriptors of implicit ratings. Implicit positive ratings are often rightly considered since the choice of an object by a user is a strong implicit rating. But the possibilities to induce negative ratings from user actions are still under-exploited, such as an interrupted film or an interrupted book, negative comments, etc. Indeed, these implicit ratings are inevitably noisy because they are deduced indirectly from user actions. Another difficulty inherent in recommender systems based on implicit rating is the evaluation of these systems. The estimation of explicit ratings is a clear goal and is discussed in the literature [3]. It can be evaluated directly. The evaluation of a system of estimation of implicit ratings is more debatable.

Among the applications of recommender systems, music recommendation is very important in the context of digital music streaming. Many systems of music recommendation have been listed in surveys [16]. The consumption of music exhibits specificities that must be taken into account by the recommendation engines. For example, facing the large amount of tracks available (several tens of millions on some platforms), recommender systems can consider music at the artist level [17], at the album level or at the track level.

The study presented in this article is devoted to music recommendation, choosing to place itself at the music track level. The item considered here is therefore a music track. No descriptors of music tracks are taken into account : no content-based approach is applied.

Generally, existing music recommendation engines take as input the data corresponding to the collection of the clicks of users on tracks. Sometimes, these clicks are filtered in order to take into account only track streams that last at least 30 seconds. However, a click or a 30-second play does not necessarily imply a complete listening of the track. It also does not systematically mean that the user likes the track. Indeed, a listening can be passive, such as listening to a radio or a playlist. Otherwise, it can be active, with the explicit choice of a track by the user. Thus, when listening is passive, listening does not even correspond to a choice of the user. Moreover, if the user launches a playlist or a radio, he can passively listen to a track or a short part of a track which he does not like and which he would not have liked to be suggested.

These peculiarities of the music consumption lead to consider other meaningful elements among the usage data to supplement the explicit data and to deduce from them implicit ratings. Early studies indicate that implicit ratings may be more accurate in the area of the music recommendation [18]. Browsing and listening histories seem particularly relevant to describe the rating of a track by users. Indeed, a user who particularly enjoys a track will have a strong tendency to listen to it again. On the other hand, a track that has not be appreciated will probably not be listened again. Similarly, a short time listening tends to indicate that the user has not wanted to listen to the track until the end, what can be translated by the fact that he does not appreciate it too much. Thus, taking into account these implicit rating data may improve the quality of

[1] https://media.netflix.com/en/company-blog/goodbye-stars-hello-thumb

the recommendations. Early work on play counts for the recommendation based on collaborative filtering [19] or on skipping behavior for playlist generation [20, 21] have led to promising results.

However, more recent results, such as the Million Song Dataset [22] (MSD) challenge, have shown that the most accurate systems have been obtained by neighbourhood-based recommender systems with binary data as input [23]. Taking into account the play counts did not have a positive impact on the accuracy of the systems evaluated.

The study proposed in this paper deals with two issues raised by the various observations relating to music recommendation:

1. Is it possible to improve the relevance of the recommended tracks by taking account of durations and replays?
2. From a list of music tracks, is it possible to obtain a precise estimation of the ratings of these tracks by a specific user? Is it possible to filter out the tracks that this user would probably not appreciate?

To address these two problems, the data collected for the proposed experiments are presented in Section 2.1. An analysis of the users' behaviour with respect to the durations and replays is presented in Section 2.2. Then, the experimental protocols and the results of evaluations of difference recommendation algorithms are presented for different recommendation algorithms in Section 3. In particular, different input data are compared in Section 3.3 and several filtering algorithms are proposed and tested in Section 3.4.

2. Analysis of the dataset

In this section, the dataset used for the experiments of this article is described. Durations and repetitions of music plays are studied in order to deduce criteria of implicit *like/dislike*.

In the following, the set denoted by U represents the set of users. The set denoted by I represents the set of tracks. The set denoted by E represents the set of listening events (assumed as a user has listened to a track). Each event is related to a user, a song and a time-stamp t. They are therefore denoted by $E(u, i, t)$ with $u \in U$ and $i \in I$.

2.1. Dataset

One of the references concerning database for recommender system experiments is the dataset proposed by Netflix [24]. This dataset consists of approximately 1 million explicit ratings (1 to 5 stars) of over 17,000 movies by roughly 500,000 users. The algorithm competition proposed with this dataset consisted of comparing prediction algorithms. In the area of movie recommendation, other very high size datasets, such as MovieLens [25], also allow comparisons of recommender systems.

In the field of music recommendation, several evaluation datasets have been proposed. Based on the Netflix challenge, the KDD cup [26] was organized in 2011 and is based on data from the Yahoo Music service. The challenge was also to estimate user preferences from explicit ratings. The amount of data is very huge, with over 250 million ratings available. Another dataset, named *Million Song Dataset*, is one of the most widely used in the field of musical information retrieval. The tags and audio features of a million songs of over 40,000 artists are made available. The challenge *MSD* [22], organized in 2012 and mostly based on this dataset, consisted of comparing the prediction algorithms of the listening events. More recently, the dataset *LFM-1b* [27] delivers a collection of one billion songs by 120,000 users. The peculiarity of this dataset comes from the availability of the timestamps of each listening event. The dataset *30Music* [28] makes available roughly 31 million listening events by over 45,000 users, with the particularity of grouping listening sessions.

In this paper, a new dataset specific to music is considered. The originality concerns the availability of duration of the listening events, mostly not proposed by other datasets. Thus, the duration of the listening events enables to assume whether the tracks have been listened to in their entirety or not. By way of comparison, no *skip* events, e.g. very short listening events, are available in the *30Music* [28]. For the study proposed, the data were collected on the international music streaming platform Deezer[2] during the month of September 2016. This limited time over a month is deliberately short to avoid the effects of dynamic on the evolution of musical tastes [29]. The anonymised data is composed of over 180 million listening events of over 3,100,000 tracks by roughly 450,000 users. The dataset considered cannot be made publicly available. No explicit feedback from users about musical tracks is available. In the following, evaluations will thus focus on implicit feedbacks.

Users were randomly chosen, from a list of active users, without any selection process according user profiles, musical genres or styles. Minimal activity was required for user selection, minimizing the possibility that these users would use other music sources [30]. Figure 2 show this user activity by representing the number of tracks, and the corresponding total duration, listened by each user of the dataset. Table 1 shows other details about the dataset, such as the number of tracks, users or listening events. The dataset has not been filtered, which implies that some listening events result

[2] www.deezer.com

from passive sessions such as radios or playlists, while other events result from an explicit choice of users.

Following the model of the *MSD* challenge, the collected data are randomly separated into two groups A and B. 100,000 users were randomly selected . From this set, users without any listening activity during the month considered were removed, resulting in 87,785 users for the group B. All other users who listened at least one song compose the group A. This group was limited to 100 million listening events. All listening events are considered for the users from the group B, resulting in over 80 million listening events. In this group B, for each user, the listening events are randomly separated into two subgroups: *visible* events (subgroup denoted by B_V) and *hidden* events (subgroup denoted by B_H). The random separation is computed according to couple <user, track> to ensure that each couple <user, track> is only present in one subgroup.

Concerning the study presented in this paper, data consists of to quadruplets <user, track, listening duration, time>. The set of these quadruplets is denoted by E. The data of the groups A, B, B_H and B_V are denoted respectively by E_A, E_B, E_{B_V} and E_{B_H}. $E(u,i,t)$ represents the quadruplet of E whose user is u, whose track is i, and whose timestamp is t:

$$E(u,i,t) = (u,i,d,t) \in E, \forall d \tag{1}$$

Data are anonymised, so tracks and users are represented by integer identifiers. The duration time is expressed in seconds and is denoted by $d(u,i,t)$ for the duration of the listening event $E(u,i,t)$ of the track $i \in I$ by the user $u \in U$, at the time t.

2.2. Listening events statistics

The dataset used for this study allows to attribute to each interaction between a track and a user the duration of listening. On the other hand, it is also possible to deduce directly from these data the repetitions of listening, e.g. the replays, that is to say the number of times that a user listens to the same track. Before considering these two properties as input to the recommender systems, it is necessary to evaluate their distribution.

Listening events. The listening durations of the tracks vary widely, ranging from one second to the full duration of the track. Figure 1 presents the distribution of these listening durations. Table 2 gives some quantitative data, e.g. values of:

$$|\{d(u,i,t) \setminus d(u,i,t) < d_0\}| \tag{2}$$

for $d_0 \in \{5, 30, 60, 120\}$ seconds.

The first observation is the high proportion of very low durations: almost a quarter of the listening events lasts for less than 5 seconds, nearly two thirds of the

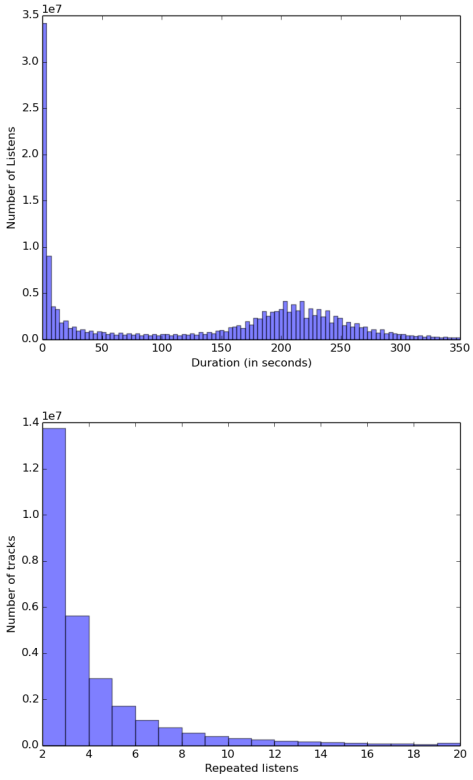

Figure 1. Distribution of the listening times (in seconds) in the dataset (left), and distribution of play counts in the dataset (right).

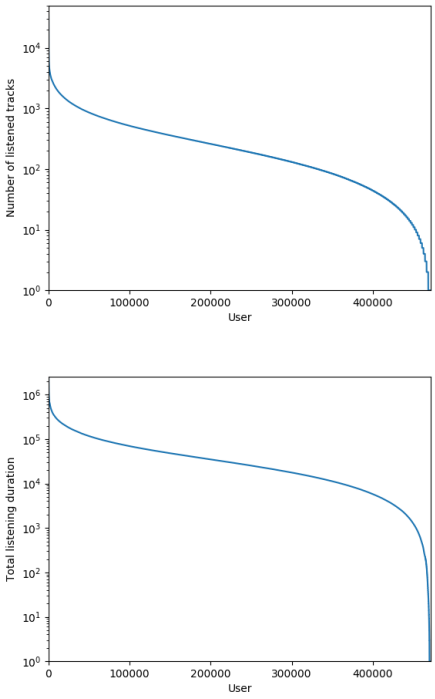

Figure 2. Number of tracks listened per user (left) and total duration of listening events per user (right).

Data	Tracks $\|I\|$	Users $\|U\|$	Events $\|E\|$	Unique Pairs (u, i)
Data A	2,442,392	383,678	100,000,000	54,390,720
Data B	2,004,193	87,785	80,255,408	30,776,280
Total	3,189,794	471,463	180,255,408	85,167,000

Table 1. Details about the two parts of the dataset used for recommender system experiments.

Data	Events $< 5s$	Events $< 30s$	Events $< 60s$	Events $< 120s$
Data A	24.4%	34.1%	37.8%	42.9%
Data B	24.2%	34.0%	37.8%	43.0%

Table 2. Distribution of the durations of listening events.

listening events last for more than 30 seconds. These short durations can be explained by several behaviours. For example, some users listen sets of tracks (for example, playlists) by clicking and listening only a few seconds each a song in order to find a particular track. Other users listen to songs passively from a themed radio, but perform regular skips on tracks they do not like. Moreover, some users choose one specific playlist and decide to listen a few songs from this playlist by rapidly skipping other songs. Such habits lead to get listening events that might last even less than one second. The distribution of the durations of listening events beyond the 30 first seconds indicates that after a minimum duration of listening, users reach more and more the end of the track listened to. Since the duration of a track is not always the same, it appears on Figure 1 a Gaussian-like type distribution with a greater proportion of listening durations around 210 seconds. This is justified by the greater proportion of popular existing tracks whose duration is around 3 minutes and 30 seconds, the favourite format for singles and radio broadcasts.

Play counts. Play counts of the track i by the user u are denoted by $p(u, i)$, for $(u, i) \in U \times I$, and defined according to the following equation 3 :

$$p(u, i) = |\{E(u, i, t), \forall t\}| \tag{3}$$

Analysis of the distribution of these play counts p also reveals very different behaviours. Figure 1 shows the distribution of these play counts. Table 3 gives some quantitative data by presenting the number of play counts greater than 2, 5 and 10, that is to say $p_{p_0} = |\{p(u, i) \setminus p(u, i) \geq p_0\}| \text{for} p_0 \in \{2, 5, 10\}$. The proportion of replays is significant since it represents almost one third of the interaction between a user and a track. Therefore, listening to the same track several times is rare (for example only a few percent for more than 10 replays) and may indicate that a track is particularly appreciated.

Finally, analysis of the listening durations reveal a high proportion of durations with low values. However, either these short-length plays are not taken into account in the existing studies on the music recommendation, or the listening events are taken into account in the same way, whatever their durations. Similarly, replays represent a significant proportion of listening events. In order to consider these usage properties in a recommendation engine, it is necessary to translate them into implicit ratings.

2.3. Definitions

For applications related to recommendation, explicit ratings are often scores, such as in the Netlfix system, or sometimes just binary ratings to indicate whether the product is loved by the user (for example a heart or a thumb-up) or, on the contrary, is not loved by the user (eg a thumb-down or a crossed heart). Obviously, this way of indicating a rating implies that all the items are not rated, since a user only submits a rating for the tracks for which his opinion is definitive. In the following, the terms of *like* (the user likes the track) and *dislike* (the user does not like the track) correspond to this binary rating.

However, these explicit *like*/*dislike* ratings are not always available in applications. Moreover, even when this is the case, they are not always provided by the users because they are often felt as effortful. They can also be distorted by the fact that they are sometimes shared with other users. This is why it is experimented here to deduce from the interactions of the user implicit ratings under the *like*/*dislike* form. Several definitions are proposed:

Skip. Listening event is assumed as short, e.g. related to a *skip* action, when a track has been listened to, but the listen duration has been less than 30 seconds. This shortened listening of a track i by user u at time t is denoted by $K(u, i, t)$ and is defined by:

$$K(u, i, t) = \{E(u, i, t) \setminus d(u, i, t) < 30\} \tag{4}$$

$K(u, i)$ is the number of shortened plays:

$$K(u, i) = |\{K(u, i, t), \forall t\}| \tag{5}$$

Data	p_2	p_5	p_{10}
Data A	30.8%	5.8%	1.5%
Data B	39.6%	11.5%	4.2%

Table 3. Distribution of play counts p_{p_0} for $p_0 \in \{2, 5, 10\}$.

Stream. A listening event is named as a *stream* if it lasts at least 30 seconds. Although it is independent from the song duration, this threshold is usually considered by music industry or streaming services for analysis (charts for example) or royalty calculations. For the approach proposed, the same threshold is assumed. Therefore, a listening event whose duration is over 30 seconds of a track i by user u at time t is denoted by $S(u, i, t)$ and is defined by:

$$S(u, i, t) = \{E(u, i, t) \setminus d(u, i, t) \geq 30\} \tag{6}$$

Stream plays. A stream is assumed when a track is listened for more than 30 seconds by the same user. The number of listening events for track i by user u whose duration is greater than 30 seconds is denoted by $P(u, i)$ and is defined by:

$$P(u, i) = |\{S(u, i, t), \forall t\}| \tag{7}$$

2.4. Towards implicit *like*/*dislike*

On the basis of these definitions, as well as the previous observations of the duration and play count distributions, a definition for *like* and *dislike* implicit ratings is proposed for the following.

Implicit like. A user u has listened at least twice to the same track i more than 30 seconds, e.g. if $P(u, i) \geq 2$, without any skip event, e.g. $K(u, i) = 0$. This choice is justified by the intuition that replaying a track over a long time suggests that the user is satisfied with this track. In this case, implicit *like* rated by the user u for the track i, denoted $L(u, i)$, is 1. Otherwise, it is 0.

$$L(u, i) \longrightarrow \begin{cases} 1 & \text{if } P(u, i) \geq 2 \text{ and } K(u, i) = 0, \\ 0 & \text{else.} \end{cases} \tag{8}$$

Implicit dislike. A user u has skipped at least one time a track i, e.g. $K(u, i) > 0$, without having ever listened to it another time, e.g. $P(u, i) = 0$. This choice is justified by the intuition that shortening a play indicates that the related track is not appreciated. In this case, implicit *dislike* rated by the user u for the track i, denoted $D(u, i)$, is 1. Otherwise, it is 0.

$$D(u, i) \longrightarrow \begin{cases} 1 & \text{if } K(u, i) > 0 \text{ and } P(u, i) = 0, \\ 0 & \text{else.} \end{cases} \tag{9}$$

2.5. Discussion

It is important to note that a track i on which a user u interacts does not always imply an implicit *like* or *dislike*. In the definitions proposed, a decision is not always taken. Thus it is possible for a couple (u, i) that $L(u, i)$ and $D(u, i)$ are both 0. On the other hand, for a couple (u, i), it is impossible for $L(u, i)$ and $D(u, i)$ to be both 1.

These choices for implicit *like* and *dislike* definitions are debatable but appear justifiable by observations of average usage. Obviously, each user behaves differently, and the ideal would be to automatically adapt these definitions to each user. The experiments described below are not intended to validate these choices, but to study the impact of these very simple criteria on the quality of the recommendations. Table 4 gives the number of streams, skips and the number of couples (user, track) that can be recoded into implicit *likes* and *dislikes*. It is important to note that their number and ratio are of equal value in both data groups A and B and are significant, notably the quantity of skip events (34%) and the number of couples (user, track) with implicit *dislike* (around 30%). In the following, the objective is to try to improve the quality of the music recommendations by reducing the numbers of skipped tracks or implicit *dislike*, while maximizing the numbers of streams and implicit *like*.

3. Qualitative study of recommendations

From the dataset described above, a music recommender system can be developed and evaluated. Several types of models can be compared according to the usual criteria, such as the metrics used in the *MSD* Challenge [22], but also by taking into account the implicit rating criteria proposed.

3.1. Employed recommendation algorithms

Several types of recommendation engines are proposed in the literature [3]. However, content-based approaches are not possible for the available test database. Collaborative filtering approaches can be tested using dedicated implementations such as *Apache Spark* [31] or *Apache Mahout* [32]. Model-based approaches prove to be accurate for a goal of predictive estimation of ratings of tracks by users. However, for the purpose of finding a list of tracks that a user will appreciate to listen to, the results of the MSD challenge have

	Data A	Data B
Listening Events $\|E\|$	100,000,000	80,255,408
Streams $S(u,i)$	65,896,289 (65.9%)	53,006,478 (66%)
Skips $K(u,i)$	34,103,711 (34.1%)	27,248,930 (34.0%)
Unique pairs (u,i)	54,390,720	30,776,280
Implicit like $L(u,i)=1$	7,189,641 (13.2%)	4,645,605 (15.1%)
Implicit dislike $D(u,i)=1$	17,165,006 (31.6%)	9,246,306 (30.0%)

Table 4. Distribution of the different types of listening events in the experimental datasets.

shown the limitations of model-based approaches. *Apache Mahout*[3] also provides implementations of neighbourhood-based (user-oriented or item-based) methods. Since the ratio between tracks and users in the dataset makes the application of the algorithms based on the neighbourhoods between items too time consuming, only the algorithms based on the neighbourhoods between users are experimented. However, tests of item-based algorithms that we carried out on limited datasets yielded results similar to those obtained by applying user-based algorithms.

Model or neighbourhood-based approaches take as inputs explicit or implicit ratings of a track by a user. These ratings are number values (integers in general). From the dataset proposed here, it is possible to consider the number of listening events $P(u,i)$ for each pair $(u,i) \in U \times I$, or only one binary value describing the fact whether the user has listened (possibly considering a minimum listening time) the track or not. Another approach tested here in Section 3.3 is to reduce the usage data describing the interaction of a user on a track to get only one rating score. This score is implicit and can then be input of algorithms.

3.2. Evaluation of recommendation algorithms

Evaluation of recommender systems is a vast and difficult problem [33]. Here, the evaluation proposed is offline. It evaluates the ability of the recommender system to retrieve the tracks the user will listen to. A part of the tracks listened to by this user being known, the objective is to find the list of the other tracks that the user has actually listened to. The evaluation is therefore based on a hidden part of the dataset, similarly to evaluation in the MSD challenge. To do this, two groups of users of the dataset were randomly selected. From the first group, the data A relating to these users are grouped together and are entirely available for learning models or neighbourhoods. From the second group, the data B are constituted. This data B is split into two parts: for each user, a random half of the listened tracks constitutes the data B_V, visible for learning, while the other hidden half constitutes the data B_H, remaining only available for evaluation. It was chosen to consider only one fold, since the time necessary to compute recommendations is very high and since the number of users in the dataset is very high. The set of users of the dataset B is denoted by U_B.

Many evaluations of recommender systems, such as the Netflix prize, rely on Root Mean Squared Error (RMSE) metrics, measuring the similarity between predicted and actual estimates given by users. Since the objective of the evaluation is to retrieve the tracks listened by each user, the evaluation metrics are those used in evaluations of information retrieval systems. Mean Average Precision at k (MAP@k) is measured following the evaluation computed during the MSD challenge [34]. Therefore, the objective here is to propose to users the tracks that they do not have listened in the visible part of the dataset, but that they listen in the visible part.

Table 5 presents the results obtained by running two different algorithms. An algorithm recommending the most popular tracks is proposed for comparison. It is simply based on the neighbourhoods between users (100 users), calculated on binary data with Tanimoto distance. The best results are obtained by this algorithm *Mahout User-Based Binary*. The difference with recommendation by popularity is significant. The resulting values are similar to the ones obtained during the challenge MSD [23].

Finally, the results show that given the list of tracks listened a user, a recommender system based on binary data find more precisely the tracks that the user will then listen to than a system based on popularity. This evaluation criterion cannot be considered exhaustive

[3] http://mahout.apache.org/

Algorithms	MAP_E@10	MAP_E@100	MAP_E@500
Popularity	0.153 (0.257)	0.051 (0.091)	0.039 (0.061)
Mahout User-Based binary	**0.482** (0.353)	**0.214** (0.194)	**0.163** (0.141)

Table 5. $MAP_E@k$ (k=10,100,500) for popularity-based and user-based recommendation engines. Here, user-based recommendation engine considers listening events as input. Standard Deviations are indicated in brackets.

since it concerns the tracks that the user will listen to, even without using a recommender system. The recommendation of these tracks is however of great importance. Indeed, proposing to a user the tracks that he would listen to reassures him and thus favours a better acceptance [35].

However, a more qualitative analysis of recommended tracks shows limitations to these good quantitative results. Tracks recommended and listened by users are not always complete listening event: 39% of listening events concerning recommended tracks last less than 30 seconds and are nevertheless evaluated positively. Similarly, listening event does not always mean a positive appreciation of the track : 21% of listening events match the implicit *dislike* criteria, for only 22% for the implicit *like* criteria. These limitations express the need to evaluate the quality of the tracks recommended, in order to obtain recommendations that will favour the tracks which will be appreciated by the users (*like*) and avoid the undesired or unappreciated tracks (*skip*, *dislike*). This is the purpose of the suggestions and experiments presented in the next two sections.

3.3. Improvement by considering durations and replays

The input data of the recommender systems are made up of all the tracks listened to by users, whatever the duration of the listening event. In order to answer the first question raised in Section 1, it is proposed here to filter the input listening data according to durations and replays. Three inputs are compared: all listening events (i.e. no filtering), durations over 30 seconds (i.e. streams) and implicit *likes*.

The three induced systems are evaluated according to MAP at ranks 10, 100 and 500. They can be compared with the reference recommender system based on the popularity of the tracks, whose evaluation results are presented in table 5. As the objective is to evaluate also the quality of the recommended tracks, 5 evaluations of MAP on 5 different criteria are proposed : listening events ($MAP_E@k$), streams ($MAP_S@k$), *like*($MAP_L@k$), skips ($MAP_K@k$) and *dislike* ($MAP_D@k$). The definitions of MAP@k are different for these 5 evaluations. For example, for the calculation of MAP@k considering *like*, only the recommended tracks that gave rise to an implicit *like* are considered positive, e.g. for user $u \in U_B$, the tracks i such that $L(u,i) = 1$.

The expected objectives are therefore different according to the criteria. For listening events, streams and *likes*, the highest MAP@k value is expected, as it would indicate that the tracks listened to, appreciated or liked, are highly ranked in the list of the recommended tracks. On the contrary, for skips and *dislikes*, the lowest value of MAP@k is expected because it would indicate that short-time listened or disliked tracks are not highly ranked, or even absent, in the list of the recommended tracks. Table 6 shows the different MAP@k values obtained, using the recommendation engine based on the neighbourhoods between users, denoted UB (User-Based), from the binary listening data.

In general, MAP@k values are relatively low. But it is important to remember that no filtering (for example considering popularity) of tracks is processed during our experiments. Nevertheless, MAP@k values remain much higher than those obtained with a popularity-based approach. For example, the $MAP_L@100$ for *like* tracks is 0.067 for the system that inputs all listening events, whereas it is only 0.014 for a system based on popularity.

Concerning $MAP_E@k$ relative to the tracks listened to, the best results are obtained when all the tracks listened by users are the inputs: at rank 100, $MAP_E@100$ is 0.214 whereas $MAP_S@100$ is only 0.157 when the inputs are only listening events whose duration is more than 30 seconds (e.g. streams). This result appears logical, because the information is more complete than the other inputs, which are filtered and therefore reduced. Therefore, the identification of neighbours implies more recommendations. More recommendations corresponding to the actual listening events are thus possible. On the other hand, as observed previously, $MAP_L@k$ for tracks *like* is quite low compared to $MAP_E@k$ for tracks played : 0.086 instead of 0.482, for example at rank $k = 10$. This significant difference corroborates the observation previously done that only 22% of the recommended tracks are implicit *likes*. Regarding recommendations that may be considered as wrong picks, MAP@k for skips or tracks *dislike* are quite high. Indeed, nearly 40% of recommended tracks are skipped, and 21% are not appreciated by users.

By modifying and filtering the input data of the recommender system, an increase of the MAP@k can be observed. For example, with only the tracks *like* as

Algorithms	MAP@10	MAP@100	MAP@500
$MAP_E@k$ (events)			
Popularity	0.153 (0.257)	0.051 (0.091)	0.039 (0.061)
UB / events	**0.482** (0.353)	**0.214** (0.194)	**0.163** (0.141)
UB / streams	0.423 (0.339)	0.181 (0.173)	0.140 (0.131)
UB / likes	0.207 (0.250)	0.080 (0.113)	0.057 (0.080)
$MAP_S@k$ (streams)			
Popularity	0.108 (0.205)	0.035 (0.067)	0.033 (0.056)
UB / events	0.358 (0.330)	0.157 (0.170)	**0.139** (0.139)
UB / streams	**0.371** (0.335)	**0.158** (0.167)	0.139 (0.137)
UB / likes	0.190 (0.247)	0.074 (0.111)	0.060 (0.084)
$MAP_L@k$ (likes)			
Popularity	0.020 (0.070)	0.014 (0.043)	0.016 (0.044)
UB / events	0.086 (0.178)	0.067 (0.130)	0.071 (0.127)
UB / streams	0.093 (0.184)	**0.071** (0.132)	**0.076** (0.130)
UB / likes	**0.099** (0.196)	0.067 (0.129)	0.070 (0.127)
$MAP_K@k$ (skips)			
Popularity	0.067 (0.162)	0.027 (0.062)	0.027 (0.056)
UB / events	0.156 (0.239)	0.085 (0.129)	0.085 (0.116)
UB / streams	0.113 (0.189)	0.060 (0.098)	0.062 (0.094)
UB / likes	**0.035** (0.075)	**0.020** (0.044)	**0.021** (0.045)
$MAP_D@k$ (dislikes)			
Popularity	0.024 (0.074)	0.012 (0.033)	0.014 (0.033)
UB / events	0.069 (0.148)	0.046 (0.089)	0.050 (0.086)
UB / streams	0.024 (0.060)	0.020 (0.046)	0.024 (0.047)
UB / likes	**0.009** (0.032)	**0.006** (0.021)	**0.007** (0.022)

Table 6. MAP@k (rangs k=10,100,500) on different criteria (listening events, streams, skips, implicit likes and dislikes, for the same recommendation engine based on neighbourhoods between users, for different inputs (listening events, streams and likes). Standard Deviations are indicated in brackets. These differences with the reference algorithm based on popularity are statistically significant (Wilcoxon $p < 0.05$).

input, the $MAP_L@10$ relative to the tracks *like* is 0.099 instead of 0.086 for the $MAP_E@10$. The number of appreciated tracks within the recommended tracks is increased here by 13% and these tracks are best ranked in the list of recommendations. This difference appears numerically limited but may be of great importance for some applications.

On the other hand, the differences are more important for skips or tracks *dislike*. Still considering the tracks *like* as inputs, $MAP_D@10$ relative to the *dislike* tracks is lower, 0.009 instead of 0.069 for the $MAP_E@10$. The number of unappreciated tracks here is almost divided by 5. Filtering input data significantly reduces the possibilities that the user is recommended a track that he will not appreciate.

Several conclusions can be drawn from this comparisons: by filtering the input data of the recommender systems, the recommendation of tracks *like* is slightly improved, and the recommendation of skipped tracks or tracks *dislike* is quite sharply reduced. Some users may be very sensitive to the recommendation of tracks they do not like. Filtering data over duration and replays may be a good approach to limit the possibility of recommending such tracks.

3.4. Filtering lists of tracks

The positive results of the previous section invite to consider other applications related to the recommendation, such as the possibility of filtering or reranking the tracks of a list (playlist, radio), taking into account the previous interactions of the user, in particular durations and replays. This problem is related to question 2 in Section 1.

General method. Figure 3 proposes an illustration of the approach proposed. From the listening events (durations and replays) of a user, implicit ratings can be estimated for the list of tracks considered. These ratings can then be used to select, filter or rerank the initial list of tracks to better match the tastes of the user.

Research in the music recommendation area provides many approaches to estimate the ratings of tracks that a user does not know [3]. Evaluations of these estimations have been the objective of the Netflix prize [24] and are mainly based on the measurement

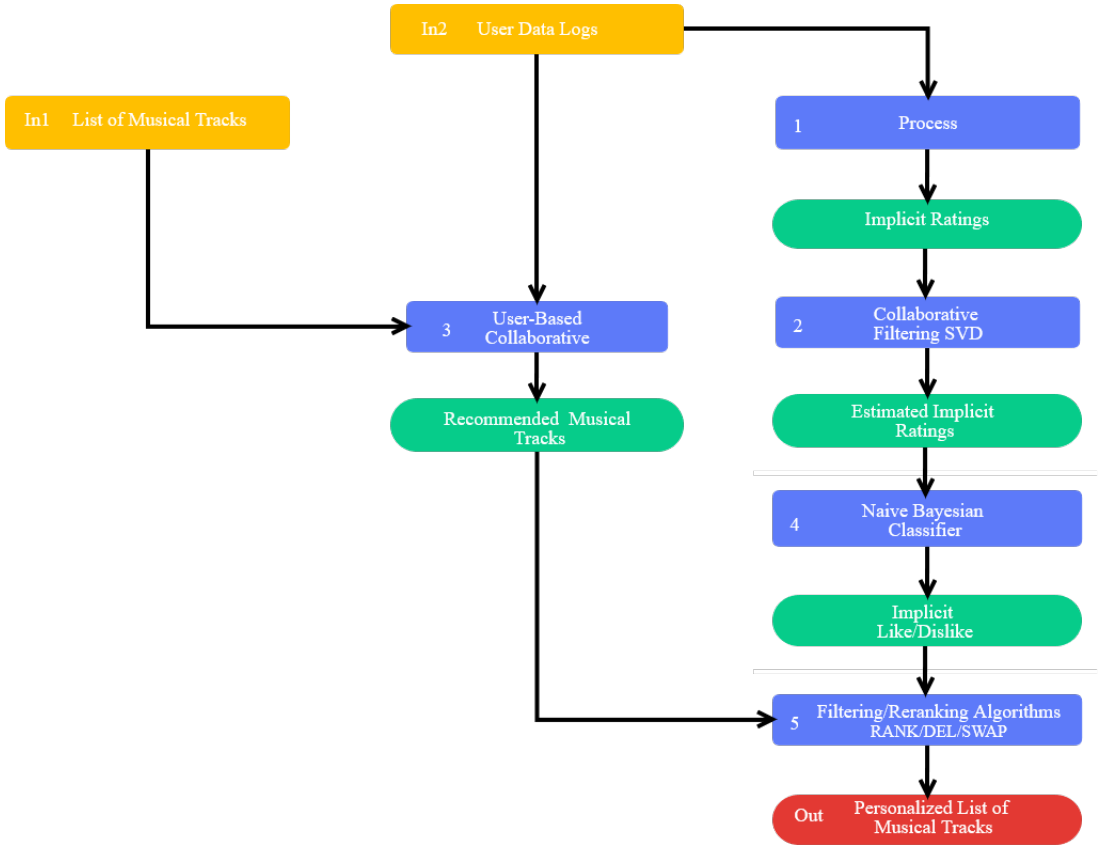

Figure 3. Illustration of the filtering process proposed: a list of tracks is filtered regarding the user usage data in order to obtain a list of personalized tracks.

of the error between the real value and the estimated one, for example by considering Root Mean Square Error (RMSE) calculation. The most accurate methods are based on latent factor models [36]. These methods prove to be less efficient on retrieval metrics such as MAP@k. But the problem raised here is to treat a posteriori a list of tracks, possibly computed by another recommendation engine or by editors. Methods based on latent factor appear thus to be a good choice. Among these methods, the SVD method available in the Apache Mahout toolkit[4] is chosen for the following experiments, with matrix factorization computed by applying Stochastic Gradient Descent algorithm[5]. Settings are fixed empirically: 500 latent factors, 20 iterations and 0.05 as regularization parameter. Inputs are composed of all listening events.

Score function. The difficulty in applying such model-based systems comes from the fact that there is no explicit rating available of music tracks, on the contrary of the Netflix prize for example. It is therefore necessary to propose a mapping function, denoted by f, which transforms the listening data, including durations and replays, into integers or real numbers corresponding to ratings:

$$f : E(u,i) \longrightarrow r(u,i) \tag{10}$$

where $E(u,i) = \{E(u,i,t), \forall t\}$ is the set of listening events of the track i by the user u and $r(u,i)$ is the implicit rating of the track i by the user u, corresponding to an integer ($r(u,i) \in \mathbb{N}$).

Obviously, there is a multitude of possibilities for defining this function f, and thus a multitude of possible choices to define what *a user loves a track* means for duration or replay data. The analysis of listening data presented in Section 2.2 indicates that the behaviour of each user can differ greatly from this point of view. Indeed, some users will skip several times the same track, while continuing to love it and expecting to listen to it later. On the contrary, other users, after skipping a track, will never listen to it again. Behaviours are multiple and their study and modelling would require specific work beyond the scope of this study.

In the following, three simple functions were chosen empirically to calculate ratings from listening durations $d(u,i,t)$ and play counts $P(u,i)$. Implicit ratings computed are reduced to known explicit rating schemes

[4]class SVDRecommender
[5]class ParallelSGDFactorizer

and correspond to integers between 1 and 5, using the principle of star rating systems such as Netflix. The objective here is not to find the optimal function, but to verify the contribution of the filtering approach.

Function f_1 considers only the stream plays $P(u,i)$ by setting a threshold at 2 streams. If there is more than 2 streams, it is considered that the track is appreciated and a score between 3 and 5 is assigned, depending on the number of plays. If there is exactly 2 streams, the implicit rating is assumed as more ambiguous and the score is only 3. Finally, if there is only 1 stream or less (only listening events whose duration are less than 30 seconds for example), the score assigned takes a low value or the minimum value of 1 if there is no stream at all. Thus, function f_1 is described in equation 11 below:

$$f_1 : E(u,i) \longrightarrow \begin{cases} r(u,i) = 5 & \text{if } P(u,i) \geq 4, \\ r(u,i) = 4 & \text{if } P(u,i) = 3, \\ r(u,i) = 3 & \text{if } P(u,i) = 2, \\ r(u,i) = 2 & \text{if } P(u,i) = 1, \\ r(u,i) = 1 & \text{if } P(u,i) = 0. \end{cases} \tag{11}$$

Functions f_2 and f_3 consider both durations and replays. Function f_2 is based on the implicit *like* and *dislike* definitions introduced in Section 2.2. Thus, if we implicitly consider that a user u likes a track i, that is $L(u,i) = 1$, then the rating assigned is the maximum value of 5. If, on the contrary, we implicitly consider that the user u does not appreciate the track i, that is, $D(u,i) = 1$, then the rating assigned is the minimum value of 1. Apart from these two cases, the rating assigned is a neutral score of 3. This function f_2 is described in Equation 12 below:

$$f_2 : E(u,i) \longrightarrow \begin{cases} r(u,i) = 5 & \text{if } L(u,i) = 1, \\ r(u,i) = 3 & \text{if } D(u,i) = 0 \\ & \text{and } L(u,i) = 0, \\ r(u,i) = 1 & \text{if } D(u,i) = 1. \end{cases} \tag{12}$$

Finally, function f_3 considers the number of skips $K(u,i)$ with respect to the number of plays $P(u,i)$. If the number of plays is greater, the rating is globally positive by being greater than or equal to 3, depending on the number of plays. On the other hand, if the number of skips exceeds the number of plays, the rating is less positive, and then changes according to the number of plays. This function f_3 is described in Equation 13 below:

$$f_3 : E(u,i) \longrightarrow \begin{cases} r(u,i) = 5 & \text{if } P(u,i) \geq 4 \\ & \text{and } K(u,i) < P(u,i), \\ r(u,i) = 4 & \text{if } P(u,i) \geq 2 \\ & \text{and } K(u,i) < P(u,i), \\ r(u,i) = 3 & \text{if } K(u,i) < P(u,i), \\ r(u,i) = 2 & \text{if } K(u,i) \geq P(u,i) \\ & \text{and } P(u,i) > 0, \\ r(u,i) = 1 & \text{if } P(u,i) = 0. \end{cases} \tag{13}$$

	Data A				
	rating 1	rating 2	rating 3	rating 4	rating 5
f_1	31.4%	49.2%	10.2%	3.8%	5.5%
f_2	31.4%		55.4%		13.3%
f_3	31.4%	7.8%	42.8%	12.8%	5.2%
	Data B_V (non-hidden part of data B)				
	rating 1	rating 2	rating 3	rating 4	rating 5
f_1	29.7%	43.7%	11.5%	5.0%	10.1%
f_2	29.7%		55.0%		15.4%
f_3	29.7%	10.5%	36.2%	14.4%	9.3%

Table 7. Distribution of ratings for data A and B_V.

From the user listening data, the functions proposed are applied to compute the implicit ratings for each track listened to. Thus, a list of triplets $(u,i,r(u,i))$ is obtained, with $u \in U$ and $i \in I$. Table 7 shows the distributions of the implicit ratings obtained, according to the different mapping functions, for the data A and B_V (non-hidden part).

In this table, it is important to note that the number of ratings $r(u,i)$ with the minimum value of 1 is the same regardless of the chosen score function (31.4% of scores for data A). This observation is justified by the choices of the three functions tested, each of which considers a track on which the user has performed a short play (less than 30 seconds) and has never listened to. This criterion also corresponds to the implicit *dislike* definition. For the other scores, the distributions are different with a higher density on the rating 2 for the function f_1 (49.2% for the data A), and on the rating 3 for the function f_3 (55.4% for data A). More generally, the distributions for the data A are very close to those of the data B_V.

The obtained triplets $(u,i,r(u,i))$ constitute the input of the recommender system allowing to estimate the ratings of all the couples (u,i), in particular the couples for which there was no interaction of the user u on the track i. The estimated ratings are denoted by $\tilde{r}(u,i)$.

Classification of recommended tracks. Estimated ratings should discriminate tracks that the user may like from those that the user may not like. This discrimination

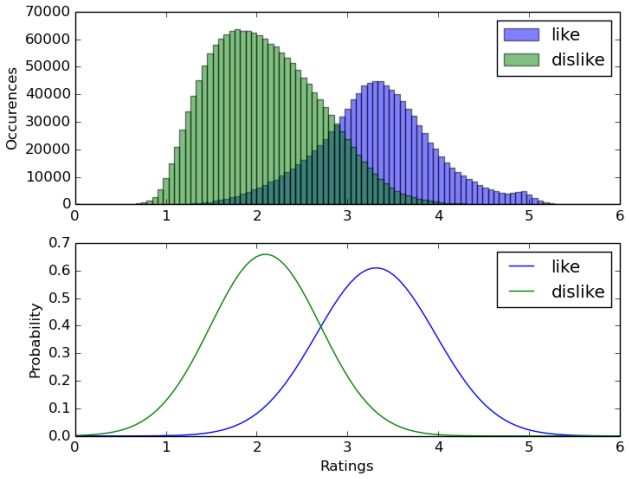

Figure 4. Distribution of the two classes like/dislike (implicit) for the ratings estimated by the SVD recommendation engine on the data B, according to the mapping function f_3 defined by Equation 13.

corresponds to a binary classification based on estimated ratings. To perform this classification, a Bayesian naive method is applied. The evaluation proposed here is based on the filtering of tracks recommended by the UB recommender system based on user neighbourhoods, which considers listening events as inputs, since this algorithm gives the best results in terms of $\text{MAP}_E@k$ (see table 6). This system produces a list of recommended tracks for each user. This list thus constitutes the input of the filtering system experimented. The goal of the filtering process is to increase the proportion of recommended tracks the user will appreciate (*like*) and reduce the proportion of recommended tracks the user will not appreciate (*dislike*). The classification induced is expected to label the tracks by one of the classes *like* or *dislike*.

Figure 4 shows the distributions of the ratings obtained by the function f_3 (for example) and estimated for tracks from data B, labelled by the classes *like* or *dislike*, as well as the associated Gaussian modelling. For example, in this configuration, distributions allow for possible discrimination with a precision of 80% for a 83% recall. The applied naive Bayes classification allows us to assign a label *like* or *dislike* to each recommended track. For a user u and a track i, this is equivalent to obtain estimations of the two functions $D(u,i)$ and $L(u,i)$ which are denoted by $\widetilde{D}(u,i)$ and $\widetilde{L}(u,i)$ in the following.

Filtering algorithms. The next step consists in processing the list of recommended tracks in order to improve it, by considering estimated ratings and/or labels. Before any filtering process, this list of tracks recommended for user u is denoted by $R(u)$.

Several algorithms are possible, but only three possibilities are compared here:

- The first algorithm consists in reranking the tracks i of the recommendation list $R(u)$ with respect to the estimated evaluation ratings $\tilde{r}(u,i)$, in descending order. This algorithm, denoted RANK, is described below.

Algorithm 1 Algorithm RANK

Ranking tracks i of the list $R(u)$ according to $\tilde{r}(u,i)$ (in descending order)

- The second algorithm proposed is based on the deletion of tracks that are assumed to be unappreciated by the user. A track assumed to be an implicit *dislike* must therefore be removed from the list of recommended tracks. The algorithm DEL described below consists therefore in browsing the list of tracks recommended for the user u, and in removing from this list the tracks i for which $\widetilde{D}(u,i) = 1$.

Algorithm 2 Algorithm DEL

for track i in list $R(u)$ **do**
  if $\widetilde{D}(u,i) = 1$ **then**
    delete i from list $R(u)$
  end if
end for

- The third algorithm proposed relies on the principle of the previous DEL algorithm by removing from the list the recommended estimated as implicit *dislike*, e.g. tracks for which $\widetilde{D}(u,i) = 1$. The difference with DEL algorithm is the replacement of the track deleted by the first track of the rest of the list for which $\widetilde{L}(u,i) = 1$. The proposed algorithm named SWAP is described below in the algorithm 3.

Evaluation of filtering process. Table 8 shows the distribution of the recommended tracks according to the different criteria used in Section 2.2, for different filtering algorithms and different mapping functions. These distributions are compared to the list of recommended tracks without any filtering process.

The results obtained correspond to the fixed objective, since the application of the filtering algorithms generally allow to increase the proportion of streams and *like*, while significantly decreasing the proportion of skips and *dislike*. For example, the algorithm DEL

Filtering	Events	% Streams (> 30s)	% Like	% Skips	% Dislike
No filtering	**492,898**	79.0	22.1	39.1	21.0
SWAP	373, 233	87.6	28.1	33.2	12.4
DEL	375, 911	91.7	29.4	**30.8**	8.3
RANK	224, 885	**93.0**	**36.1**	36.7	**7.0**

Table 8. Influence of different filtering algorithms on the quality of recommendation at rank 10, with rating f_1.

Algorithm 3 Algorithm SWAP

Input: R(u), functions $\tilde{r}$, $\widetilde{L}$ and $\widetilde{D}$
for track i in list $R(u)$ **do**
  if $\widetilde{D}(u,i) = 1$ **then**
    delete i from list $R(u)$
    for track j in list $R(u)$ from the rank of the track i **do**
      if $\widetilde{L}(u,i) = 1$ **then**
        insert j instead of i in list $\widetilde{R}(u)$
      end if
    end for
  else
    insert i in list $\widetilde{R}(u)$
  end if
end for
return $\widetilde{R}(u)$

applied with mapping function f_3 results in more than 29% of tracks *like* instead of 22% without any filtering process. At the same time, the proportion of tracks *dislike* is 8.3% instead of 21% without filter. The comparison between the filtering algorithms indicates that the SWAP algorithm is less accurate than the DEL or RANK algorithms. The explanation for this lower contribution probably comes from the cumulative classification errors, replacing tracks estimated to be *dislike* by tracks estimated to be *like*. This aggregation of errors is not carried out either with the RANK algorithm or with the DEL algorithm, which only take into account the *dislike* tracks. The RANK algorithm has the disadvantage of filtering many tracks, which has an impact on the total number of *like* tracks recommended, although the percentage is much improved (36.1% instead of 22.1% without any filtering process).

It is important to note that the number of recommended tracks actually listened to is falling sharply as expected, since this number includes all tracks, whether liked or not, in the first 10 recommendations. So, the number of tracks *like* is quite close: for example approximately 109, 000 without any filtering process, while 110, 500 tracks will be *like*d after filtering in the top 10. The difference is important on tracks which are not appreciated : 31, 200 tracks *dislike* are present in the top 10 after filtering, while they are over 103, 500 before. In this case, filtering process results in the removal of two-thirds of the tracks *dislike* in the top 10. Despite the inevitable errors on classification of tracks *like* and *dislike*, filtering process overall improves the selection of tracks from a pre-established list. The choice between the different settings of the different algorithms can be made according to the desired objective, which can be, for example, to minimize the tracks *dislike* for a specific user, or to reduce the tracks *dislike* while preserving the maximum number of tracks *like* for a less demanding user who usually listens to a lot of recommended tracks.

Evaluation of the recommender system after filtering. Another way of evaluating these filtering operations is to compare the MAP at different ranks, on the same principle proposed in Section 3.3. It is expected that the filtering process will have a positive influence on the results given in Table 6. Table 9 shows MAP at ranks 10, 100, and 500 obtained by filtering the tracks recommended by user-based recommender system, for different types of inputs according to the DEL, RANK, and SWAP algorithms.

The results show that MAP for listening events and streams remain higher without any filtering process. This observation was expected since the filtering process consists in deleting tracks. The number of tracks in the list that are then listened to decreases accordingly. Even if deleted tracks are unappreciated tracks in the list, such as tracks *dislike*, the $\text{MAP}_E@k$ and $\text{MAP}_S@k$ evaluations essentially quantify the importance of the filtering process carried out. At first glance, this decrease appears to be a negative result, which is not the case, since the objective here is mainly to evaluate performance considering *likes* and *dislikes*.

The filtering process applied has a very low impact on $\text{MAP}_L@k$ relative to tracks *like*: it sometimes results in slightly better $\text{MAP}_L@10$ (0.087 for DEL instead of 0.086 without any filtering process), but sometimes worse at higher ranks (0.065 for DEL instead of 0.067 without any filtering process, at rank 100). The algorithm SWAP is no better than the algorithm DEL, which is probably due to a lack of precision on the classification of tracks into *like* / *dislike* groups. The algorithm RANK allows to obtain a lower $\text{MAP}_L@k$ score than for the algorithms DEL or SWAP : tracks with the highest estimated ratings are not always the tracks that have been listened to and appreciated by users.

Algorithms	Rating functions	MAP@10	MAP@100	MAP@500
$MAP_E@k$ (events)				
No		**0.482** (0.353)	**0.214** (0.194)	**0.163** (0.141)
SWAP	f_1	0.282 (0.324)	0.125 (0.162)	0.068 (0.0.089)
SWAP	f_2	0.362 (0.350)	0.161 (0.181)	0.091 (0.107)
SWAP	f_3	0.341 (0.341)	0.149 (0.173)	0.084 (0.102)
DEL	f_1	0.319 (0.350)	0.106 (0.157)	0.069 (0.104)
DEL	f_2	0.317 (0.356)	0.121 (0.170)	0.089 (0.125)
DEL	f_3	0.363 (0.355)	0.131 (0.168)	0.093 (0.120)
RANK	f_1	0.182 (0.275)	0.062 (0.103)	0.050 (0.071)
RANK	f_2	0.059 (0.128)	0.035 (0.063)	0.053 (0.057)
RANK	f_3	0.182 (0.260)	0.070 (0.107)	0.071 (0.080)
$MAP_S@k$ (streams)				
No		**0.358** (0.330)	**0.157** (0.170)	**0.139** (0.139)
SWAP	f_1	0.235 (0.300)	0.099 (0.144)	0.067 (0.093)
SWAP	f_2	0.285 (0.323)	0.122 (0.159)	0.083 (0.109)
SWAP	f_3	0.292 (0.326)	0.124 (0.159)	0.084 (0.108)
DEL	f_1	0.276 (0.325)	0.096 (0.148)	0.073 (0.109)
DEL	f_2	0.278 (0.336)	0.109 (0.162)	0.089 (0.127)
DEL	f_3	0.324 (0.340)	0.123 (0.162)	0.099 (0.126)
RANK	f_1	0.169 (0.266)	0.058 (0.100)	0.055 (0.079)
RANK	f_2	0.049 (0.119)	0.028 (0.056)	0.045 (0.055)
RANK	f_3	0.169 (0.252)	0.065 (0.104)	0.070 (0.085)
$MAP_L@k$ (likes)				
No		0.086 (0.178)	**0.067** (0.130)	**0.071** (0.127)
SWAP	f_1	0.072 (0.166)	0.054 (0.115)	0.050 (0.104)
SWAP	f_2	0.079 (0.174)	0.060 (0.124)	0.056 (0.115)
SWAP	f_3	0.082 (0.175)	0.062 (0.125)	0.059 (0.116)
DEL	f_1	0.082 (0.176)	0.057 (0.121)	0.058 (0.118)
DEL	f_2	0.080 (0.175)	0.057 (0.125)	0.060 (0.123)
DEL	f_3	**0.087** (0.178)	0.065 (0.128)	0.067 (0.126)
RANK	f_1	0.054 (0.144)	0.037 (0.092)	0.042 (0.090)
RANK	f_2	0.019 (0.082)	0.016 (0.055)	0.024 (0.058)
RANK	f_3	0.059 (0.147)	0.043 (0.098)	0.050 (0.098)
$MAP_K@k$ (skips)				
No		0.156 (0.239)	0.085 (0.129)	0.085 (0.116)
SWAP	f_1	0.095 (0.193)	0.050 (0.092)	0.041 (0.075)
SWAP	f_2	0.105 (0.196)	0.059 (0.105)	0.047 (0.083)
SWAP	f_3	0.088 (0.172)	0.049 (0.093)	0.040 (0.074)
DEL	f_1	0.102 (0.195)	0.042 (0.084)	0.039 (0.079)
DEL	f_2	0.067 (0.149)	0.034 (0.077)	0.035 (0.077)
DEL	f_3	0.088 (0.171)	0.037 (0.076)	0.037 (0.075)
RANK	f_1	0.069 (0.154)	0.029 (0.065)	0.032 (0.064)
RANK	f_2	**0.016** (0.047)	**0.012** (0.026)	**0.024** (0.034)
RANK	f_3	0.055 (0.124)	0.025 (0.056)	0.035 (0.059)
$MAP_D@k$ (dislikes)				
No		0.069 (0.148)	0.046 (0.089)	0.050 (0.086)
SWAP	f_1	0.024 (0.077)	0.018 (0.047)	0.014 (0.037)
SWAP	f_2	0.042 (0.115)	0.029 (0.065)	0.024 (0.053)
SWAP	f_3	0.025 (0.083)	0.019 (0.050)	0.015 (0.038)
DEL	f_1	0.017 (0.050)	0.008 (0.027)	0.009 (0.027)
DEL	f_2	0.017 (0.056)	0.012 (0.038)	0.014 (0.039)
DEL	f_3	0.016 (0.045)	0.009 (0.030)	0.010 (0.030)
RANK	f_1	**0.006** (0.023)	**0.003** (0.013)	**0.006** (0.014)
RANK	f_2	0.006 (0.024)	0.005 (0.012)	0.013 (0.019)
RANK	f_3	0.006 (0.024)	0.004 (0.013)	0.012 (0.019)

Table 9. MAP@k (k=10,100,500) on different criteria (listening events, streams, skips, implicit like and dislike, for the different filtering algorithms DEL, SWAP and RANK. Reference algorithm with no filtering is the user-based collaborative filtering algorithm based on listening events. Standard Deviations are indicated in brackets. These differences with the reference algorithm with no filtering are statistically significant (Wilcoxon $p < 0.05$).

EAI Endorsed Transactions on Artificial Intelligence and Robotics
Online First

On the other hand, the difference is greater on the $MAP_K@k$ and $MAP_L@k$, relative to skips and tracks *like*. For example, the $MAP_K@k$ relative to the skips drops significantly for all the algorithms tested (for example 0.037 for DEL instead of 0.085 at rank 100 without any filtering process). Similarly, the $MAP_D@k$ relative to the tracks *dislike* drops very sharply as well: 0.009 for DEL instead of 0.046 at rank 100 without any filtering process. This decrease in MAP confirms the benefit of filtering to remove from the list the tracks that the user would not appreciate. The algorithm RANK obtains the lowest $MAP_K@k$ and $MAP_L@k$, in particular with the function f_2. This algorithm, by placing the tracks with the lowest ratings estimated at the end of the list, proves to be quite effective in this context. The balance between $MAP_L@k$ and $MAP_D@k$, relative to the tracks *like* and *dislike*, has of course to be adjusted according to the applications concerned, as discussed above.

In general, the results are better for the rating function f_3, which takes into account both replays and the listening event durations. Thus, it is certainly necessary to consider other parameters of description of the usage of the users, which could further improve the efficiency of the filtering process.

These experiments confirm the benefit of the filtering algorithms proposed, based on durations and replays. The list of tracks passed as input (e.g. the list of recommended tracks here) can be filtered and thus greatly reduce skips and tracks *dislike* while preserving tracks *like*.

4. Conclusions and Perspectives

The questions raised in this article concern the consideration of replay and listening duration information. The dataset collected on a streaming platform allows to show that replays and listening durations can be effectively considered to improve the qualities of music recommender systems. There are two main contributions in the studies proposed. The first one concern the possibility of filtering the input data of recommender systems with respect to the listening event durations and replays, in order to improve the quality of the recommendations. In particular, it is thus possible to greatly attenuate the number of recommended tracks which will not be appreciated by users. The second contribution is the advantage of estimating implicit user ratings based on replays and listening durations, in order to estimate implicit ratings for all tracks and thus post-filter lists of tracks to adapt them to each user. The experiments show in particular that the number of tracks that the user will not appreciate can thus be greatly diminished. There are many applications for the improvement of radios or playlists by adapting them to each user and thus preventing users from changing radio or playlist they listen to. These two contributions mainly rely on the implicit negative ratings, which is specific to the musical context.

However, the proposed evaluations have limitations, particularly as they are offline. They are therefore limited to the a posteriori analysis of user usages. Implementation of online evaluations would allow to verify the results presented here, in particular the advantage of filtering recommendation lists based on implicit ratings.

There are many perspectives. It would be important to consider the relevance of other types of implicit ratings based on user actions such as collecting (favourites, playlists) for example. Other more explicit elements should also be tested such as the *thumb-up/thumb-down* often offered while listening to a track.

As mentioned earlier, the limitations of the approach proposed concern the estimation of implicit ratings based on listening events. Each user has its own behaviours, so the main challenge is to be able to propose implicit rating processes specific to each user and to evaluate the overall influence on the quality of the recommendation.

Acknowledgements

The author wishes to thank Simbals team, Deezer R&D and Recommendation teams for making this work possible, in particular Manuel Moussalam, Thomas Bouabca and Aurélien Hérault.

References

[1] Resnick, P. and Varian, H.R. (1997) Recommender Systems. *Communications of the ACM* **40**(3): 56–58, URL http://doi.acm.org/10.1145/245108.245121.

[2] Park, D.H., Kim, H.K., Choi, I.Y. and Kim, J.K. (2012) A Literature Review and Classification of Recommender Systems Research. *Expert Systems with Applications* **39**(11): 10059–10072.

[3] Ricci, F., Rokach, L. and Shapira, B. (2015) *Recommender Systems Handbook*, **54**.

[4] Burke, R. (2002) Hybrid Recommender Systems : Survey and Experiments. *User Modeling and User Adapted Interaction* **12**(4): 331–370, URL http://www.springerlink.com/index/N881136032U8K111.pdf.

[5] Goldberg, D., Nichols, D., Oki, B.M. and Terry, D. (1992) Using Collaborative Filtering to Weave an Information Tapestry. *Communications of the ACM* **35**(12): 61–70.

[6] Balabanović, M. and Shoham, Y. (1997) Content-based, collaborative recommendation. *Communications of the ACM* **40**(3): 66–72.

[7] Burke, R. (2007) Hybrid Web Recommender Systems. In *The Adaptive Web* (Berlin, Heidelberg: Springer Berlin Heidelberg), 377–408.

[8] Delgado, J. and Ishii, N. (1999) Memory-Based Weighted-Majority prediction. In *Proceedings of the ACM SIGIR Workshop on Recommender Systems*.

[9] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P. and Riedl, J. (1994) GroupLens: an Open Architecture for Collaborative Filtering of netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work (CSCW '94)* (New York, New York, USA: ACM Press): 175–186.

[10] Linden, G., Smith, B. and York, J. (2003) Amazon.com Recommendations: Item-to-item Collaborative Filtering. *IEEE Internet Computing* **7**(1): 76–80.

[11] Sarwar, B., Karypis, G., Konstan, J. and Reidl, J. (2001) Item-Based Collaborative Filtering Recommendation Algorithms. In *Proceedings of the tenth international conference on World Wide Web (WWW '01)*: 285–295.

[12] Paterek, A. (2007) Improving Regularized Singular Value Decomposition for Collaborative Filtering. In *ACM KDD Cup*: 39–42.

[13] Hofmann, T. (2003) Collaborative Filtering via Gaussian Probabilistic Latent Semantic Analysis. In *Proceedings of the 26th annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '03)* (New York, New York, USA: ACM Press): 259.

[14] Oard, D.W. and Kim, J. (1998) Implicit Feedback for Recommender Systems. In *AAAI Workshop on Recommender Systems*: 81–83.

[15] Hu, Y., Koren, Y. and Volinsky, C. (2008) Collaborative Filtering for Implicit Feedback Datasets. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08 (Washington, DC, USA: IEEE Computer Society): 263–272.

[16] Celma, Ã. (2010) *Music recommendation and discovery: The long tail, long fail, and long play in the digital music space*.

[17] Schedl, M. and Schnitzer, D. (2014) Location-Aware Music Artist Recommendation. In *Proceedings of the 20th International Conference on Multi Media Modeling (MMM 2014)* (Dublin, Ireland), **8326 LNCS**.

[18] Schafer, J.B., Frankowski, D., Herlocker, J. and Sen, S. (2007) Collaborative Filtering Recommender Systems. *The Adaptive Web* **4**(2): 81–173.

[19] Pacula, M. (2009) *A Matrix Factorization Algorithm for Music Recommendation using Implicit User Feedback*. Tech. rep., URL http://www.mpacula.com/publications/lastfm.pdf.

[20] Hu, Y. and Ogihara, M. (2011) Nextone Player: a Music Recommendation System Based on User Behavior. *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR'11)* : 103–108.

[21] Pampalk, E., Pohle, T. and Widmer, G. (2005) Dynamic Playlist Generation Based On Skipping Behavior. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR'05)*: 634–637.

[22] Bertin-Mahieux, T., Ellis, D.P., Whitman, B. and Lamere, P. (2011) The Million Song Dataset. *Proceedings of the International Conferences on Music Information Retrieval (ISMIR'11)* : 591–596.

[23] Aiolli, F. (2013) Efficient Top-N Recommendation for Very Large Scale Binary Rated Datasets. *Proceedings of the 7th ACM Conference on Recommender Systems (RecSys '13)* : 273–280.

[24] Bennett, J. and Lanning, S. (2007) The Netflix Prize. *Proceedings of KDD Cup and Workshop* : 3–6.

[25] Harper, F.M. and Konstan, J.A. (2015) The MovieLens Datasets. *ACM Transactions on Interactive Intelligent Systems* **5**(4): 1–19.

[26] Dror, G., Koenigstein, N., Koren, Y. and Weimer, M. (2012) The Yahoo! Music Dataset and KDD-Cup'11. In *JMLR Workshop and Conference Proceedings*, **18**: 3–18.

[27] Schedl, M. (2016) The LFM-1B Dataset for Music Retrieval and Recommendation. *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval* : 103–110.

[28] Turrin, R., Quadrana, M., Condorelli, A., Pagano, R. and Cremonesi, P. (2015) 30Music Listening and Playlists Dataset. *Proceedings of International Conference on Recommender Systems (RecSys'15)* .

[29] Yang, D., Chen, T., Zhang, W., Lu, Q. and Yu, Y. (2012) Local Implicit Feedback Mining for Music Recommendation. *Proceedings of the sixth ACM Conference on Recommender Systems (RecSys'12)* : 91.

[30] Louail, T. and Barthelemy, M. (2017), Headphones on the Wire: Statistical Patterns of Music Listening Practices, URL https://arxiv.org/pdf/1704.05815.pdf.

[31] Zaharia, B.Y.M., Xin, R.S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X. *et al.* (2016) Apache Spark: A Unified Engine for Big Data Processing. *Communications of the ACM* **59**(11): 56–65.

[32] Owen, S., Anil, R., Dunning, T. and Friedman, E. (2011) *Mahout in Action*.

[33] Shani, G. and Gunawardana, A. (2011) Evaluating Recommendation Systems. In *Recommender Systems Handbook*, 257–297.

[34] McFee, B., Bertin-Mahieux, T., Ellis, D.P. and Lanckriet, G.R. (2012) The Million Song Dataset Challenge. In *Proceedings of the 21st International Conference Companion on World Wide Web (WWW '12)*: 909.

[35] Jannach, D. and Adomavicius, G. (2016) Recommendations with a Purpose. *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys'16)* : 7–10.

[36] Sarwar, B.M., Karypis, G., Konstan, J.A. and Riedl, J.T. (2000) Application of Dimensionality Reduction in Recommender System - A Case Study. In *In Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Workshop on Web Mining for E–Commerce – Challenges and Opportunities (WEBKDD'00)*.