

The Benefit of Using Tag-Based Profiles

Claudiu S. Firan

L3S Research Center
Appelstrasse 9a
30167 Hanover, Germany
firan@L3S.de

Wolfgang Nejdl

L3S Research Center
Appelstrasse 9a
30167 Hanover, Germany
nejdl@L3S.de

Raluca Paiu

L3S Research Center
Appelstrasse 9a
30167 Hanover, Germany
paiu@L3S.de

Abstract

Collaborative tagging, i.e. the process of assigning metadata in the form of keywords to shared content by many users, has emerged as an important way to provide information about resources on the Web and elsewhere. Such keywords (tags) are used to enable the organization of information within personal information spaces, such as photo collections, but can also be shared, allowing browsing and searching with the help of tags attached by other users to information resources from the Web. Recent research has shown that such tag distributions stabilize over time and can be used to improve search on the Web. In this paper we are interested in another aspect, namely how they characterize the user and enable personalized recommendations. Using data from a frequently used music search portal, Last.fm, we analyze tag usage and statistics and investigate the use of tag-based user profiles in contrast to conventional user profiles based on song and track usage. We specify recommendation algorithms based on tag user profiles, and explore how collaborative filtering recommendations based on these tag profiles are different from recommendations based on song/track profiles. Finally, we describe a new search-based method, which uses tags to recommend songs interesting to a user, yielding substantially improved results. The paper finishes with a discussion of some future work to further improve tag-based search and recommendation in community web sites.

1. Introduction

More and more companies start offering personalized services toward their users and online music recommender systems are one prominent example. Pandora¹, Last.fm²,

Foafing the Music³ or Yahoo! Music⁴ are a few of the currently online available music recommendation systems. These systems employ different approaches for recommending music tracks to their users, ranging from content based and collaborative filtering techniques to hybrid methods. While clearly useful to their users, these more conventional recommendation techniques still suffer from a number of problems: in the case of collaborative filtering⁵, musical pieces with no ratings cannot be recommended because recommendations are based on actual user ratings. Besides, artist variety in recommended pieces can be poor, making these recommendations less satisfactory than they could be. Recommending tracks that are similar to users' favorites in terms of content induces unreliability in modeling users' preferences and besides, content similarity does not necessarily reflect preferences. Hybrid recommendation techniques combine the advantages of the two approaches and are thus better. However, to our knowledge, the only usable music recommender system using hybrid techniques is Foafing the Music, which relies heavily on FOAF profiles created by the users - not an easy task for non-expert users. Last, but not least, though many of these community sites allow tagging, these tags are not used for recommendation or any form of advanced search.

The rest of this paper is structured as follows: Section 2 presents a short motivating scenario, in Section 3 we discuss existing approaches to music recommendations. We continue with the description of the Last.fm dataset, which we used in our experiments (Section 4). In Section 5 we define track-based and respectively tag-based profiles, discuss how they differ from each other and present appropriate algorithms for creating these user profiles. Using the algorithms described in Section 5, we then propose 7 different music recommendation methods (Section 6) based on

¹<http://www.pandora.com/>

²<http://www.last.fm/>

³<http://foafing-the-music.iaa.upf.edu/>

⁴<http://music.yahoo.com/>

⁵Method of making automatic predictions about the interests of a user by collecting preference information from many users.

collaborative filtering and search. We evaluated our algorithms through a user study, the experimental setup and the results we obtained are summarized in Section 7. Section 8 presents our conclusions and several ideas to be further investigated.

2. A Motivating Scenario

As motivation for our work, let us look at the following simple example: Tom is a programmer in a big company and while writing code he likes to listen to music. His preferred music is “electronic” and related kinds of music. On his desktop⁶ Tom has stored quite a few MP3 files with his favorite music, but as he has been listening to them over and over again, he would really like a change. He therefore decides to use an online music recommender system. He opens the corresponding web page in his browser, but before being able to listen to some new music, he needs to provide personal information like age, location and gender. He also has to indicate one or more of his favorite artists. After finishing this long registration process he can finally start using the system, but to his bad surprise, the first track he gets recommended is a piece he already has on his desktop. Tom has the possibility to rate the music this system recommends to him, such that over time the recommendations (hopefully) will get better as the system learns his preferences. However, this intensive interaction with the online music recommender system in the initial phase is quite annoying for Tom, and certainly his group manager will not be too happy if he sees Tom switching back and forth between his working environment and the web browser for rating music pieces. What Tom needs is a system not suffering from this cold start problem, a system that is able to infer his profile without much interaction, but still with the ability to provide good music recommendations.

3. Related Work

Most available music recommender systems are based on collaborative filtering methods; i.e., they recommend music to a user by considering some other users’ ratings for the same music pieces. This technique is quite widely utilized, including music shopping services like Amazon⁷ or iTunes⁸, and has proven to be effective. However, this recommendation method suffers from the cold start problem, as described in Section 2.

A better recommendation technique is described in [2]. The paper gives an overview of the Foafing the Music system, which uses the Friend-of-a-Friend (FOAF) and Rich

Site Summary (RSS) vocabularies for recommending music to a user, depending on her musical tastes. Music information, such as new album releases, related news artists, or available audio pieces, is gathered from RSS feeds from the Web, whereas FOAF documents are used to define user preferences (i.e., for building the user profiles). Another hybrid music recommendation method is presented in [6], which simultaneously considers user ratings and content similarity and is based on a three-way aspect model, so that it can directly represent substantial (unobservable) user preferences as a set of latent variables introduced in a Bayesian network. Probabilistic relations over users, ratings, and contents are statistically estimated. Our approach differs from these previous ones by the fact that the user profile is inferred automatically from his desktop music data without any additional manual effort from the user’s side, and that we rely on tag information instead of track-based profiles.

A totally different approach for producing music recommendations is presented in [4]. Their method is applied to an interactive music system that generates playlists fitting the preferences indicated by the user. For automatically generating music playlists, their approach uses a local search procedure in the solution space, based on simulated annealing: the algorithm iteratively searches the solution space moving from one solution to a neighboring solution, compares their quality and stops when an optimal solution is found.

[5] describes a system that queries web search engines for pages related to artists, downloads the pages, extracts text and natural language features, and analyzes these features in order to produce textual summary descriptions of each artist. These descriptions are used to compute similarity between artists and can be further used for producing recommendations. However, the paper does not present any evaluation experiments regarding the quality of the recommendations received by using this technique. Also, our approach differs from the one presented in [5] by the fact that they are searching the web for finding similar artists, whereas we rely on user tags (in particular from the Last.fm web site) for grouping tracks and artists and for building up user profiles.

The most similar approach to ours is [1], which uses collaboratively created data from the Web for making recommendations. However, their goal is generating personalized tag recommendations for users of social bookmarking sites such as Delicious⁹. Techniques for recommending tags do already exist and are based on the popularity of tags among all users, on time usage information, or on simple heuristics to extract keywords from the URL being tagged. Their approach complements these techniques and is based on recommending tags from URLs that are similar to the one in

⁶Throughout the paper with “desktop” we mean any personal computer.

⁷<http://www.amazon.com/>

⁸<http://www.apple.com/itunes/>

⁹<http://del.icio.us/>

question according to two variants of cosine similarity metrics. We use tags as well, but use them for recommending interesting songs based on tag-based profiles.

4. Last.fm Functionalities and Usage Data

As basis for our experiments we have used a crawl of the Last.fm website. Claiming to be the “social music revolution”, Last.fm is a great place to congregate and share musical tastes, giving to its users access to thousands of tracks from all musical genres. Before using the system, users need to create an account and specify their preferred music genre. The interaction with the service can be done through the web interface, via the embedded Flash player, or through the Last.fm player which the users need to download and install locally on the desktop. By listening to tracks and rating them (there are three options: *Love this track*, *Skip this track* or *Ban this track*), user profiles are created. Based on these profiles, the service produces a number of personalized features, such as finding out about artists that the user likes, other people with similar tastes, appropriate charts, or events in the neighborhood. Based on the music the user likes, Last.fm connects her to other users that have similar tastes (her neighbors). However, for using this feature, users need to listen to at least 5 artists and wait for about one week before their neighbors appear. Every week, the list of neighbors is updated, based on what the user has been listening to during that week.

Other social connections can be created on Last.fm through friendship relations: the user can add friends through an appropriate button, and accept friendship requests to make the friendship “mutual”. As with neighbors, they can listen to their friends’ radio stations. Users with similar interests can get together also by joining groups. Music statistics are generated for groups as well as for individual users. Anyone can create these groups and encourage people to join. It is also possible for a group founder to restrict access to the group and its associated forum.

Another interesting functionality offered by Last.fm which we will focus on in this paper is the possibility of tagging: users can assign as many tags as they want to any track, album or artist in their profile. With tagging, they can find their items better, can view everyone else’s tags, or choose to play a tag-featured radio (music that has been tagged with some specific tag).

For the purpose of our investigations, we crawled an extensive subset of the Last.fm website, namely pages corresponding to tags, music tracks and user profiles. All information extracted was indexed and stored into a Lucene¹⁰ index.

Tags. We could identify a number of 21,177 different tags, which are used on Last.fm for tagging tracks, artists or

¹⁰<http://lucene.apache.org/>

Nr.	Tag	Freq.	Nr.	Tag	Freq.
1	rock	866,040	26	hard rock	114,871
2	indie	488,944	27	industrial	110,207
3	seen live	478,098	28	punk rock	108,388
4	alternative	453,081	29	heavy metal	107,851
5	electronic	346,832	30	japanese	106,091
6	metal	312,221	31	experimental	104,122
7	pop	299,499	32	favorites	102,619
8	punk	247,665	33	chillout	98,663
9	indie rock	206,876	34	rap	97,106
10	classic rock	206,654	35	soundtrack	90,875
11	female vocalists	184,478	36	instrumental	87,022
12	alternative rock	180,320	37	hip hop	81,780
13	jazz	174,782	38	british	76,549
14	electronica	167,989	39	trance	76,460
15	hip-hop	158,571	40	indie pop	75,997
16	singer-songwriter	139,041	41	soul	75,692
17	folk	137,598	42	metalcore	74,989
18	hardcore	133,682	43	thrash metal	74,786
19	emo	130,831	44	power metal	73,921
20	death metal	129,766	45	progressive metal	73,800
21	dance	127,282	46	90s	72,279
22	80s	123,183	47	blues	68,931
23	ambient	119,680	48	acoustic	65,627
24	black metal	119,468	49	reggae	65,355
25	progressive rock	116,036	50	post-rock	64,095

Table 1. Number of times the most popular 50 tags were used on Last.fm.

albums. For each of these tags we have extracted information regarding the number of times each tag has been used, number of users which have used the tag, as well as lists of similar tags together with their similarity scores. Based on the data we have crawled we can observe that the usage of the tags follows largely a power law distribution, except for the sharp drop at the end (Figure 1): after the top first 76 most used tags (representing 0.36% of the tags) the usage number falls down to 5% of the usage number corresponding to the most used tag (“rock”). At the time of our crawl

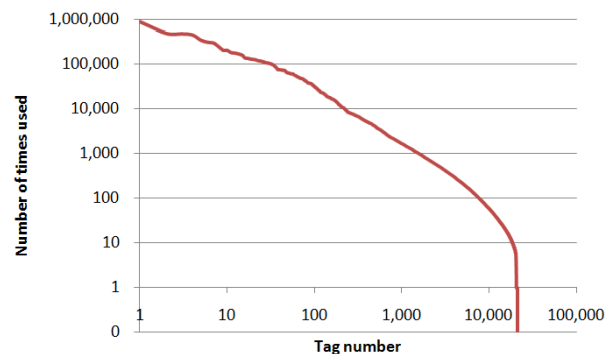


Figure 1. Tag Distribution

tags have been used by the users of the Last.fm system for a total number of 18,735,549 times. The top 50 most used tags are summarized in Table 1.

An analysis of the top 100 most used tags showed that approximately 60% of the tags represent genre descriptions, while the rest of 40% is shared among tags describing per-

sonal impressions (e.g., “seen live”), artists (e.g., “female vocalists”), time period (e.g., “80s”), country of provenience, soundtrack, tempo, or instruments.

Tracks. From the Last.fm music database we crawled a total number of 317,058 tracks and their associated attributes, including track and artist name, number of times the track has been played on Last.fm, albums featuring the track, users’ comments and tags which have been used for tagging the tracks and the corresponding tags’ scores (based on the number of times each tag has been used for tagging each track). Figure 2 presents the variation of the track popularity over the collection of tracks, showing a similar distribution as in Figure 1, except for the power law curve decreasing slower.

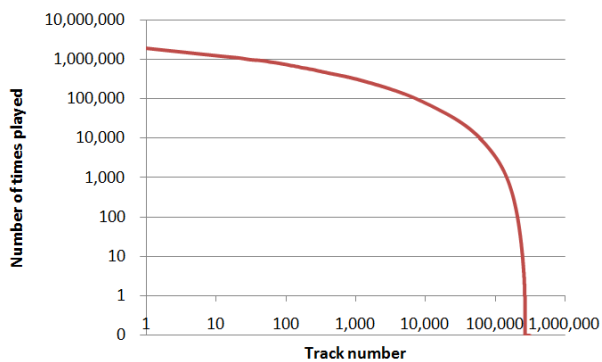


Figure 2. Track Distribution

Users. We extracted details about 289,654 registered users on Last.fm. For these users the extracted information includes user ID, gender, age, location, registration date, listened number of tracks, list of friends and neighbors, overall top listened tracks and list of used tags. From this number of users, we filtered out all users who have not yet listened to at least 50 tracks and who have not used at least 10 different tags, and were thus left with 12,193 unique users for our experiments. For these 12,193 users the distribution of tags is plotted in Figure 3, showing the correlation between the number of users and the total number of times each user made use of tags, by the time of our crawling.

Similarly, the relation user to total number of listened tracks is shown in Figure 4. Note that both figures 3 and 4 are plotted for the set of users having listened to at least 50 tracks and having used at least 10 different tags – therefore the sudden cutoff.

5. Tag-Based Profiles vs. Track-Based Profiles

Tags are more and more widely used, which makes it easier to extract them for all types of information objects, including music tracks. We chose to extract available human annotations (tags) from Last.fm tracks or directly use

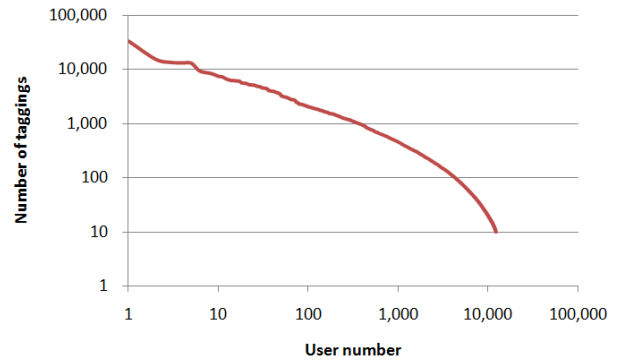


Figure 3. User - Tag Distribution

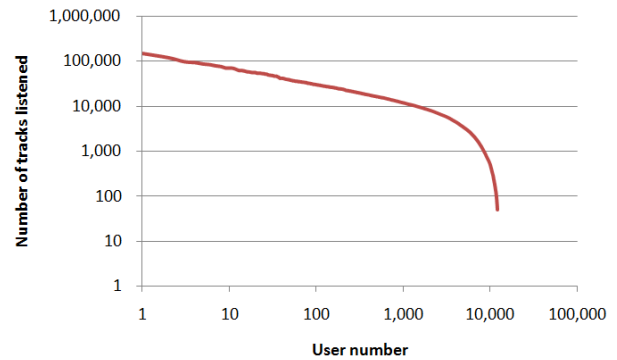


Figure 4. User - Track Distribution

those tags that the users applied themselves to other tracks or artists. Although most music recommendation methods use track-based algorithms to present the user with new interesting tracks, given the increasing tendency toward tagging all types of multimedia files on the Internet we wanted to investigate how these tags could be used for recommendations. We also wanted to avoid extensive manual ranking, so opted to construct user profiles from locally stored MP3 files. This usually works very well, as most users have quite a few music files in MP3 format on their desktop, usually much more than needed for a music recommender system to provide satisfactory results.

For the rest of the paper we will distinguish between profiles created for Last.fm users and for Non-Last.fm users. They differ in terms of the source of information which is used for building up the profiles: for Last.fm users the starting point is represented by their web pages on Last.fm, whereas in the case of the Non-Last.fm users we start from the information available on their desktops. For this type of users we extract metadata about each track existing on the desktop, and match artist and track name (extracted from filename or ID3 tag) against the Last.fm music database. This, as we will show in section 5.2 provides all data necessary to create comprehensive user profiles that accurately

reflect users' music preferences. We will use the following notations:

$ITF(TG) = \text{Inverse Tag Frequency for Tag } TG$

$p(TR, U) = \text{Preference of User } U \text{ for Track } TR$

$p(TG, U) = \text{Preference of User } U \text{ for Tag } TG$

$TR.U_Listened = \text{Number of times User } U \text{ has listened to Track } TR$

$TR.Overall_Listened = \text{Number of times Track } TR \text{ was overall listened on Last.fm}$

$TG.U_sedFor_TR = \text{Number of times Track } TR \text{ was tagged with Tag } TG \text{ by all users}$

$TG.U_sed.Overall = \text{Number of times Tag } TG \text{ was used overall}$

$TG.U_sedBy_U = \text{Number of times User } U \text{ has used Tag } TG$

$Tracks.Containing_TG = \text{Number of tracks on the User's Desktop that were tagged with Tag } TG$

We distinguish between Last.fm and Non-Last.fm users using either *Last.fm* or *Non-Last.fm* indices in the corresponding formulas.

5.1. Track-based Profiles

Track-based profiles are defined as collections of music tracks with associated preference scores, describing users' musical tastes, as follows:

$Profile.Tracks(U) = \{ \langle TR_i, P_i \rangle \mid TR_i = \text{user's track}, P_i = p(TR_i, U) \}$

5.1.1 Track-based profiles for Last.fm users

In the case of Last.fm users the profiles are inferred from the users' web pages available on the Last.fm site. Their collection of tracks includes all tracks the users have been listening to inside the system. Their associated scores are a function of the number of times users have listened to these music tracks. The algorithm for creating the track-based profiles for this type of users is described below:

Alg. 5.1.1.1: Track-based profile for Last.fm user

- 1: For each track TR in user's tracks list UTR
 - 2: **Compute track's score** P
 - 3: **Add** pair $\langle TR, P \rangle$ to user profile
 - 4: **Return** user's track-based profile
-

with $P = p(TR, U_{Last.fm}) = \log(TR.U_{Last.fm} \text{Listened})$

5.1.2 Track-based profiles for Non-Last.fm users

For Non-Last.fm users the only available source of personal information is represented by their desktops. We first extract explicit metadata such as artist and track name either from the filename or from the ID3 tags (if any) of the music files existing on the desktop. This information is then

matched against the Last.fm database and only tracks with a TFxIDF¹¹ score above 0.9 are kept for further processing. This pre-processing step is described below:

Alg. 5.1.2.1: Get list of tracks

- 1: For each track (MP3) on user desktop
 - 2: **Extract** artist name and track name from filename as $S1$
 - 3: **Extract** artist name and track name from ID3 tag as $S2$
 - 4: **Combine** $S1$ and $S2$ into S
 - 5: **Search** with S on Last.fm index
 - 6: **Retrieve** tracks LT matching query with Lucene TFxIDF score > 0.9
 - 7: **Add** tracks LT to the user's list of tracks UTR
 - 8: **Return** UTR
-

Once the list of tracks for a Non-Last.fm user is created, the track-profile is realized in a similar manner as for a Last.fm user: algorithm 5.1.1.1 is applied on the list of tracks with the only difference that the preference scores for the tracks are now a function of the overall number of times tracks have been listened on Last.fm.

Alg. 5.1.2.2: Track-based profile for Non-Last.fm user

- 1: **Create list of tracks** UTR applying Alg. 5.1.2.1
 - 2: **Apply** Alg. 5.1.1.1 on list of tracks UTR
-

with $P = p(TR, U_{Non-Last.fm}) = \log(TR.Overall_Listened)$

5.2. Tag-based Profiles

Tag-based user profiles are defined as collections of tags together with corresponding scores representing the user's interest in each of these tags. The formal definition is given below:

$Profile.Tags(U) = \{ \langle TG_i, P_i \rangle \mid TG_i = \text{user's tag}, P_i = p(TG_i, U) \}$

Again, we distinguish between profiles created for Last.fm and non-Last.fm users. For this type of profiles, the list of tags can be extracted either from the users' list of tracks (tags which have been used to tag the tracks) or directly from the tags the users have used themselves. The different variants of tag-based profiles for Last.fm and Non-Last.fm users are described in detail in Sections 5.2.1 and 5.2.2.

5.2.1 Tag-based profiles for Last.fm users

For Last.fm users, the first type of tag-based profiles can be created starting from the list of tracks the users have been

¹¹The TFxIDF score (term frequency-inverse document frequency) is a weight often used in information retrieval and text mining for evaluating how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus.

listening to on Last.fm. For each of these tracks, we extract the list of all tags which have been used inside the system for annotating them. In this case, the preference associated to a tag is proportional to the number of times these tracks tagged as TG were listened by the user and to the number times this tag has been used by all users on Last.fm to tag those tracks. The description of the algorithm is given below:

Alg. 5.2.1.1: Track-Tag-based Profile for Last.fm User

- 1: For each track TR in user's track list UTR
 - 2: **Extract** list of used tags TTG for TR
 - 3: **Add** TTG to user's tag list UTG
 - 4: For each tag TG in UTG
 - 5: **Compute** tag's score P
 - 6: **Add** pair $\langle TG, P \rangle$ to user's profile
 - 7: **Return** user's tag-based profile
-

with $P = p(TG, U_{Last.fm}) = [ITF(TG)] \cdot \log \sum_i (\log(TR_i \cdot U_{Last.fm} \cdot Listened) \cdot \log(TG \cdot U_{UsedFor} \cdot TR_i))$

$$ITF(TG) = \log \frac{\sum_i TG_i \cdot U_{UsedOverall}}{TG \cdot U_{UsedOverall}}$$

Similar to the IDF value in Information Retrieval, in order to reduce the influence of tags which are very popular among users, but might not accurately reflect user's personal musical taste, we introduce an optional parameter in the preference formula, ITF . The formula penalizes tags which appear very often and boosts the preference for tags appearing more rarely.

The second possibility for creating the tag-based profiles for Last.fm users is to directly take the tags which the users have already used (information which can be found on their web pages) together with their frequency. The algorithm in this case looks as follows:

Alg. 5.2.1.2: Tag-based Profile for Last.fm User

- 1: For each tag TG in user's tag list UTG
 - 2: **Compute** tag's score P
 - 3: **Add** pair $\langle TG, P \rangle$ to user's profile
 - 4: **Return** user's tag-based profile
-

with $P = p(TG, U_{Last.fm}) = \log(TG \cdot U_{UsedBy} \cdot U_{Last.fm})$

For this case, we do not need to introduce the ITF parameter in the preference formula, since now the profile is already very personalized – the user has directly used these tags.

5.2.2 Tag-based profiles for non-Last.fm users

For non-Last.fm users, the collection of tags building up their profiles is inferred based on the list of tracks the users have on their desktops. This list of tracks is compiled as presented in Alg. 5.1.2.1 and then transformed using Alg.

5.2.1.1. Preference scores for the tags are now computed as follows: the score depends on the number of times these tracks which are part of the users' profile and are tagged as TG have been listened by all Last.fm users and to the number times this tag has been used by all users on Last.fm. Again in the formula we have the optional parameter ITF used to decrease the bias toward very popular tags. Moreover, for Non-Last.fm users we keep in the profile only the top 100 preferred tags, after evaluating recommendation results with several such values for this algorithm ranging from 10 to 500.

Alg. 5.2.2.1: Track-Tag-based Profile for Non-Last.fm User

- 1: **Create** list of tracks UTR applying Alg. 5.1.2.1
 - 2: **Apply** Alg. 5.2.1.1 on list of tracks UTR
 - 3: **Retain** in the profile **top 100** preferred tags
 - 4: **Return** user's tag-based profile
-

with $P = p(TG, U_{Non-Last.fm}) = [ITF(TG)] \cdot \log \sum_i (\log(TR_i \cdot Overall \cdot Listened) \cdot \log(TG \cdot U_{UsedFor} \cdot TR_i))$

$$ITF(TG) = \log \frac{\sum_i TG_i \cdot U_{UsedOverall}}{TG \cdot U_{UsedOverall}}$$

The second variant of the tag-based profile corresponding to a Non-Last.fm user looks similar to the previous one. In this case the preference depends on the number of tracks on the user's desktop that are tagged with tag TG .

Alg. 5.2.2.2: Tag-based Profile for Non-Last.fm User

- 1: **Create** list of tracks UTR applying Alg. 5.1.2.1
 - 2: **Apply** Alg. 5.2.1.1 on list of tracks UTR
 - 3: **Retain** in the profile **top 29** preferred tags
 - 4: **Return** user's tag-based profile
-

with $P = p(TG, U_{Non-Last.fm}) = \log(Tracks \cdot Containing \cdot TG)$

Since Non-Last.fm users did not use the tags building up their profile by themselves — they are just inferred based on the music tracks they have on their desktop — we chose to simulate the Last.fm profiles by maintaining only the top 29 preferred tags. From the data we have crawled we could see that the average number of used tags among the users is 29, therefore we keep in the tag-profiles we create with algorithm 5.2.2.2 only top-29 most preferred tags.

6. Music Recommendations

Using track-based and tag-based profiles we implemented and evaluated different algorithms for producing music recommendations. This paper describes 7 algorithms which, based on the type of profile and the technique we used for getting the recommendations, can be grouped into three categories: Collaborative Filtering based on Tracks (Section 6.1), Collaborative Filtering based on Tags (Section 6.2) and Search based on Tags (Section 6.3).

6.1. Track-based Recommendations

CF based on TRacks (CFTR). Traditional music recommender systems use User-Item Collaborative Filtering methods with music tracks as items. This method is successfully used in Last.fm and other systems, though we still have the cold start problem, i.e. users have to listen (and possibly rank) a minimum number of tracks and tracks have to be ranked by some listeners, before recommendations are possible. We will use such an algorithm as baseline to compare our other algorithms against.

Alg. 6.1.1: Collaborative Filtering based on Tracks (CFTR)
Track Profile \leftrightarrow Track Recommendation \leftrightarrow Tracks

1: Create users **track-based profile** (Alg. 5.1.1.1/ Alg. 5.1.2.1)
2: **Get track recommendations** based on the Taste-Recommender Java library:
3: Compute **top 10 most similar users** SU with current user U
 based on cosine similarity between track profiles
4: **For each** similar user SU_i
5: Get tracks TR_j with preference $p(TR_j, SU_i)$
6: **Combine** lists TR_j of tracks into $\Rightarrow RTR$
7: **Recommend** music tracks RTR

6.2. Tag-Based Recommendations

For the three algorithms proposed in this paper as tag-based recommendation algorithms, the matrix on which we apply collaborative filtering is a User-Tag matrix. In this matrix, line i corresponds to the tag-profile of user i and contains corresponding preference scores for tags which have been used by the user (and 0 for the other tags). In these algorithms what we obtain as result of applying CF on the User-Tag matrix is of course a list of recommended tags, based on what tags other similar users have used. What we want to achieve are music recommendations and not tag recommendations. Therefore with this list of tags we search which tracks have been tagged with most of these tags, taking into account their associated preference scores. We return the top 10 matching tracks, scored by cosine similarity, as recommended songs.

CF based on Track-Tags with ITF (CFTTI). The first algorithm we propose in this category uses tag-based profiles which have been extracted from the list of tracks users have been listening to (Alg. 5.2.1.1/ Alg. 5.2.2.1). For not biasing profiles toward highly used tags, when computing preference scores associated to the tags we also include the *ITF* parameter. The recommended list of tags obtained after applying CF on the User-Tag matrix is then used for getting the music recommendations:

Alg. 6.2.1: CF based on Track-Tags with ITF (CFTTI)
Tracks \leftrightarrow Tag Profile \leftrightarrow Tag Recommendation \leftrightarrow Search w/ Tags \leftrightarrow Tracks

1: Create **tag-based profiles** (Alg. 5.2.1.1 / Alg. 5.2.2.1 both with ITF)

2: **Get tag recommendations** based on the Taste-Recommender Java library:
3: Compute **top 10 most similar users** SU with current user U
 based on cosine similarity between tag profiles
4: **For each** similar user SU_i
5: Get top 50 tags TG_j by preference $p(TG_j, SU_i)$
6: **Combine** lists TG_j of tags into $\Rightarrow RTG$
7: **Create** Query Q
8: **For each** tag TG_i in RTG
9: **Add** pair $\langle TG_i, p(TG_i, U) \rangle$ to Q
10: **Search** with Q tracks being tagged with tags in $Q \Rightarrow RTR$
11: **Compute** cosine similarity between tracks in the Lucene index and Q
12: **Rank** resulted tracks RTR based on cosine similarity
13: **Recommend** music tracks RTR

CF based on Track-Tags No-ITF (CFTTN). This second algorithm differs from CFTTI by computing the tag-based profiles without the *ITF* parameter in the formula corresponding to tags' preference. Otherwise the steps in the algorithm are the same as in Alg. 6.2.1.

CF based on Tags (CFTG). For the third algorithm the user profiles on which the tag recommendation step is based, are more personal – users have already used those tags. In this case, line 1: in Alg 6.2.1 is modified and the algorithm looks as follows:

Alg. 6.2.3: CF based on Tags (CFTG)
Tags \leftrightarrow Tag Profile \leftrightarrow Tag Recommendation \leftrightarrow Search with Tags \leftrightarrow Tracks

1: Create **tag-based profiles** (Alg. 5.2.1.2 / Alg. 5.2.2.2)
2: **Get tag recommendations** based on the Taste-Recommender Java library:
3: Compute **top 10 most similar users** SU with current user U
 based on cosine similarity between tag profiles
4: **For each** similar user SU_i
5: Get top 50 tags TG_j by preference $p(TG_j, SU_i)$
6: **Combine** lists TG_j of tags into $\Rightarrow RTG$
7: **Create** Query Q
8: **For each** tag TG_i in RTG
9: **Add** pair $\langle TG_i, p(TG_i, U) \rangle$ to Q
10: **Search** with Q tracks being tagged with tags in $Q \Rightarrow RTR$
11: **Compute** cosine similarity between tracks in the Lucene index and Q
12: **Rank** resulted tracks RTR based on cosine similarity
13: **Recommend** music tracks RTR

6.3. Tag-Based Search

In our last set of algorithms, we use the tags extracted through the previously presented methods for direct matching with other tracks. This is done by creating a disjunctive query of clauses where each clause consists of a tag and its preference. The results are tracks ordered by cosine similarity between the vector of tags they have been tagged with and the vector of tags given in the query. Direct search using tags has the big advantage of being much faster than any collaborative filtering algorithm, the results being produced instantly. It also offers the user the possibility to enter keyword queries based on tags and get new tracks from different domains if wanted.

Search based on Track-Tags with ITF (STTI). Similar to CFTTI this algorithm is based on profiles created using the algorithms 5.2.1.1 and 5.2.2.1 and includes the *ITF* factor in the preference formula:

Alg. 6.3.1: Search based on Track-Tags with ITF (STTI)
 Tags \leftrightarrow Tag Profile \leftrightarrow Search with Tags \leftrightarrow Tracks

1: Create **tag-based profiles** (Alg. 5.2.1.1/ Alg. 5.2.2.1 both with ITF)
 2: **Create** Query Q
 3: **For** each tag TG_i in the profile of current user U
 4: **Add** pair $\langle TG_i, p(TG_i, U) \rangle$ to Q
 5: **Search** with Q tracks being tagged with tags in $Q \Rightarrow RTR$
 6: **Compute** cosine similarity between tracks in the Lucene index and Q
 7: **Rank** resulted tracks RTR based on cosine similarity
 8: **Recommend** music tracks RTR

Search based on Track-Tags No-ITF (STTN). The second search-based algorithm is based on Alg. 6.3.2, just that we remove the *ITF* parameter in the preference formula.

Search based on Tags (STG). Like the CFTG algorithm, STG uses profiles created by alg. 5.2.1.2 and 5.2.2.2. Tags contained in the profiles are then directly used for searching for tracks which have been tagged with these tags:

Alg. 6.3.3: Search based on Tags (STG)
 Tags \leftrightarrow Tag Profile \leftrightarrow Search with Tags \leftrightarrow Tracks

1: Create **tag-based profiles** (Alg. 5.2.1.2/ Alg. 5.2.2.2)
 2: **Create** Query Q
 3: **For** each tag TG_i in the profile of current user U
 4: **Add** pair $\langle TG_i, p(TG_i, U) \rangle$ to Q
 5: **Search** with Q tracks being tagged with tags in $Q \Rightarrow RTR$
 6: **Compute** cosine similarity between tracks in the Lucene index and Q
 7: **Rank** resulted tracks RTR based on cosine similarity
 8: **Recommend** music tracks RTR

7. Evaluation

7.1. Experimental Setup

We evaluated our algorithms with 18 subjects (B.Sc., Ph.D., and Post- Doc students in different areas of computer science and education). They installed our desktop application to extract their user profiles as described in Alg. 5.1.2.2, 5.2.2.1, and 5.2.2.2. Then we ran all 7 variants of our algorithms (Track Collaborative Filtering *CFTR* - baseline, Track-Tag CF *CFTTI*, *CFTTN* - with or without Inverse Tag Frequency *ITF*, Tag CF *CFTG*, Track-Tag Search *STTI*, *STTN* - with or without *ITF*, Tag Search *STG*) over their profiles. The average number of tracks in a user profile was 658, ranging from 17 up to 2,848, not being statistically significant in influencing algorithm outcome. For each of the algorithms we collected the top-10 recommended items (i.e., tracks), such that each

user had to rate a maximum number of 70 recommended tracks (it is possible that the same track gets recommended by more of the proposed algorithms, in which case the track was listed only once). Results were presented to the subjects in shuffled order, so that they were not aware of the algorithm which produced the result nor the score of the recommended item. For each of the recommended tracks, the users had to provide two different scores: one measuring how well the recommended track matches their music preferences ([0] - I don't like this track, [1] - I don't mind listening to this track, [2] - I like the track) and one reflecting the novelty of the track ([0] - I already know this track, [1] - I know something about this track, e.g. I know the artist OR I heard the track on the radio, but I do not remember the name, etc., and [2] - this track is really new for me).

The quality of the recommended results was measured using the normalized version of Discounted Cumulated Gain (DCG) [3], a rich measure which gives more weight to highly ranked documents, while also incorporating different relevance levels by giving them different gain values:

$$DCG(i) = \begin{cases} G(1) & , \text{if } i = 1 \\ DCG(i-1) + G(i)/\log(i) & , \text{otherwise.} \end{cases}$$

For novelty, only the average of the marks given was used, resulting in a value ranging from 0 (known) to 2 (really new).

7.2. Results

Table 2 shows the NDCG value, its statistical significance over the *CFTR* baseline computed using T-tests, and the average popularity (number of times a track was listened to on Last.fm) of recommended tracks for each algorithm. All Collaborative Filtering algorithms based on tags (*CFTG*, *CFTTI*, *CFTTN*) performed worse than the baseline, as standard User-Item CF techniques already show high precision. All our search algorithms, though, show quite substantial improvements over track based CF (*STG* 12%, *STTI* 37%, *STTN* 44% as shown in Figure 5; *STTI* and *STTN* both highly statistically significant). This outcome is certainly positively influenced by the consistent usage of tags on Last.fm: Most frequently used tags denote the track's genre, so our search gets biased toward specific user preferred music genres. It was also interesting to note that the better people knew the tracks (i.e., a lower novelty value), the higher they rated the recommendations. We observed an almost perfect inverse correlation between these two scores, with a Pearson's correlation coefficient between average NDCG and Novelty values per algorithm of $c = -0.987$, and still a high inverse correlation of all preference and novelty marks with $c = -0.513$.

Another interesting result is that *STG* recommends much less popular tracks than our *CFTR* baseline, but still

Nr.	Algorithm	NDCG	Signif. vs. CFTR	Popularity	Novelty
1	CFTR	0.54	-	15,177	1.39
2	CFTG	0.25	High, $p \ll 0.01$	4,065	1.83
3	CFTTI	0.36	High, $p \ll 0.01$	6,632	1.72
4	CFTTN	0.37	High, $p \ll 0.01$	13,671	1.74
5	STG	0.60	No, $p = 0.22$	7,587	1.07
6	STTI	0.73	High, $p \ll 0.01$	10,380	0.82
7	STTN	0.77	High, $p \ll 0.01$	16,309	0.78

Table 2. Normalized Discounted Cumulative Gain over the first 10 recommended tracks, along with the average track popularity and average novelty

Nr.	Algorithm	NDCG	Signif. vs. CFTR	Popularity	Novelty
1	CFTR	0.60	-	19,717	1.33
2	CFTG	0.29	Yes, $p = 0.02$	7,787	1.84
3	CFTTI	0.33	Yes, $p = 0.02$	9,970	1.79
4	CFTTN	0.32	High, $p \ll 0.01$	25,576	1.77
5	STG	0.55	No, $p = 0.29$	7,799	1.11
6	STTI	0.76	Minimal, $p = 0.10$	10,709	0.81
7	STTN	0.80	Minimal, $p = 0.07$	15,664	0.61

Table 3. Normalized Discounted Cumulative Gain, average track popularity, and average novelty over the first 10 recommended tracks, only for users with less than 50 tracks on their Desktop

of higher quality, so that it is suited for people demanding a higher diversity of music, not listening to the same tracks over and over again. We can thus suggest different algorithms, depending on the user’s preference concerning popularity and novelty of tracks. Because of the high use of the “rock” tag (used twice as much as any other tag), many *hard rock* or *heavy metal* songs were recommended, mostly by tag-based CF algorithms. Further research has to be done in order to disambiguate tag meanings, and to reduce unwanted tag weights.

When looking only at people with less than 50 personal music tracks on their desktop (Table 3) - this was the case for 7 of our test subjects, the number of tracks ranging from 17 to 48 and averaging at 31 - we still find a gain of 26% and 33% over the baseline for *STTI* and *STTN*, respectively. This indicates that our user tag profiles also work with less rich music repositories. Results presented in Table 4 only for recommended tracks with high novelty (i.e., novelty mark = 2), show a decrease in NDCG for all algorithms, mostly not statistically significant since users had different music knowledge. Still *STTN* performs 13% better than *CFTR*, mainly because it recommends tracks with higher popularity.

Nr.	Algorithm	NDCG	Signif. vs. CFTR	Popularity
1	CFTR	0.31	-	22,766
2	CFTG	0.19	No, $p = 0.21$	4,852
3	CFTTI	0.24	No, $p = 0.42$	5,758
4	CFTTN	0.29	Minimal, $p = 0.13$	17,419
5	STG	0.27	No, $p = 0.25$	21,111
6	STTI	0.26	No, $p = 0.36$	29,196
7	STTN	0.35	No, $p = 0.25$	44,490

Table 4. Normalized Discounted Cumulative Gain and average track popularity over the first 10 recommended tracks, only for items with high novelty

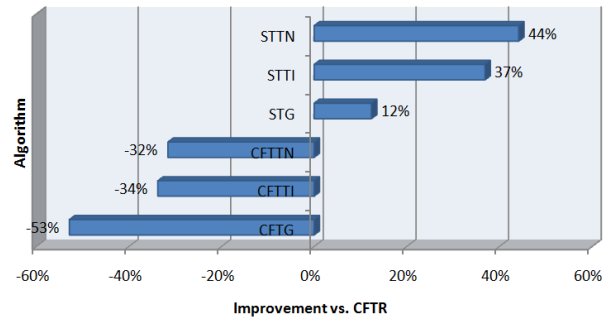


Figure 5. Relative NDCG gain (in %) over the CFTR baseline for each algorithm.

8. Conclusions and Future Work

Tag usage is increasing in many community web sites, yet only simple search algorithms are available to use these tags. In this paper we analyze tag usage and statistics for one of the most popular music community sites, Last.fm, and compare user profiles based on these tags with conventional ones based on tracks. Using these tag-based profiles, we define several new recommender and search algorithms, and investigate their behavior, comparing it to classical collaborative filtering based on track-based profiles as a baseline. A first set of algorithms, using collaborative filtering on tag profiles that were extracted from tracks, proved to be less successful than the baseline. A second set of tag-based search algorithms however improved results’ quality significantly. In addition to a 44% increase in quality for the best algorithm, search-based methods are also much faster than collaborative filtering and do not suffer from the cold start problem. An interesting additional feature we want to investigate in the future is to incorporate relevance feedback into search-based recommendations, such that the user is able to select negative tracks or tags, genres he does not like, live performances, or even instrumental setups.

References

- [1] A. Byde, H. Wan, and S. Cayzer. Personalized tag recommendations via tagging and content-based similarity metrics. *Proceedings of the International Conference on Weblogs and Social Media (ICWSM)*, 2007.
- [2] O. Celma, M. Ramirez, and P. Herrera. Foafing the music: A music recommendation system based on rss feeds and user preferences. *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR)*, 2005.
- [3] K. Järvelin and J. Keklinen. Ir evaluation methods for retrieving highly relevant documents. *Proc. of the 23th Intl. ACM SIGIR Conf. on Research and development in information retrieval*, 2000.
- [4] S. Pauws, W. Verhaegh, and M. Vossen. Fast generation of optimal music playlists using local search. *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR)*, 2006.
- [5] B. Whitman and S. Lawrence. Inferring descriptions and similarity for music from community metadata. *Proceedings of the International Computer Music Conference (ICMC)*, 2002.
- [6] K. Yoshii, M. Goto, K. Komatani, T. Ogata, and H. G. Okuno. Hybrid collaborative and content-based music recommendation using probabilistic model with latent user preferences. *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR)*, 2006.