# On Weighted Hybrid Track Recommendations

Simon Franz[1], Thomas Hornung[1], Cai-Nicolas Ziegler[2],
Martin Przyjaciel-Zablocki[1], Alexander Schätzle[1], and Georg Lausen[1]

[1] Institute of Computer Science, Albert-Ludwigs-Universität Freiburg, Germany
{franzs,hornungt,zablocki,schaetzl,lausen}@informatik.uni-freiburg.de
[2] American Express, PAYBACK GmbH, München, Germany
cai-nicolas.ziegler@payback.net

**Abstract.** Music is a highly subjective domain, which makes it a challenging research area for recommender systems. In this paper, we present our TRecS (Track Recommender System) prototype, a hybrid recommender that blends three different recommender techniques into one score. Since traceability is an important issue for the acceptance of recommender systems by users, we have implemented a detailed explanation feature that supports transparency about the contribution of each sub-recommender for the overall result. To avoid overspecialization, TRecS peppers the result list with recommendations that are based on a serendipity metric. This way, users can benefit from both recommendations aligned with their current taste while gaining some diversification.

## 1 Introduction

While e-commerce has embraced the benefits of using recommender systems early on, the music domain has long been influenced by offline radio stations, where static playlists based on track popularity and expert preselections are broadcast to every listener. With the advent of music streaming platforms, such as Last.fm[1] or Spotify[2], the balance has shifted and users can now create their own private radio stations. As a downside, users now have to curate their own playlists and are less likely to discover new music. For this, a music recommender system is an elegant supplement, which can make use of both the wisdom of the crowds and the user's past listening history.

In this paper, we present our TRecS prototype that combines multiple metrics into one comprehensive prediction score: the similarity of tracks is computed based on the listening history of Last.fm users (*track similarity*), the tags that are associated with tracks (*tag similarity*), and the temporal listening profile of individual tracks (*time similarity*). While these metrics assure recommendations that share characteristics with music a user has liked so far, we have additionally implemented a serendipity measure [1] that includes complementary music to the list of recommended tracks. TRecS is a weighted hybrid recommender [2], where the weights for each metric are adjustable, and the system supports an

[1] http://www.last.fm
[2] http://www.spotify.com

F. Daniel, P. Dolog, and Q. Li (Eds.): ICWE 2013, LNCS 7977, pp. 486–489, 2013.

explanation for each recommended track with respect to the contribution of each sub-recommender to the overall prediction score. A study with over 140 participants has shown that the perceived quality of recommendations gradually improves with the number of rated recommendation lists.

## 2 TRecS Architecture and Design

TRecS relies on one collaborative metric and two content-based metrics (cf. Section 2.1) [3]. The overall architecture of the system is shown in Figure 1. Our crawled data set from Last.fm is first preprocessed (e.g. data cleansing and disambiguation) and reduced to 50,000 tracks while maintaining key characteristics of the original data set. Afterwards, for each metric the similarity between all songs is precomputed and stored in the knowledge base. At runtime, when a user requests a new recommendation list, the user's context, i.e. the tracks rated so far, is used to compute the next recommendations based on a weighted score of the three sub-recommenders. Before the result list is returned, it is peppered with additional serendipitous tracks (cf. Section 2.4).

The TRecS prototype is available online with an introductory tutorial at:

`http://trecs.informatik.uni-freiburg.de`

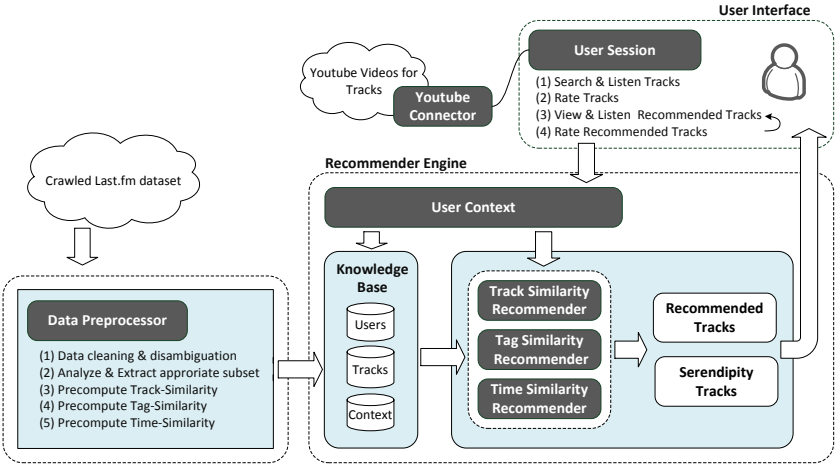


**Fig. 1.** TRecS system architecture

For computing similarities, each track is represented as a vector of *features*. The similarity between each track can now be determined with some distance metric between these vectors. Thus, it is sufficient to describe for each recommender *how* features are represented for each track and *which* distance metric is used.

### 2.1 Similarity Metrics

TRecS puts three different recommender systems to use, each based on its own similarity metric:

1. **Track similarity.** Each track is represented as a vector $\alpha_i = (c_1, \ldots, c_n)$, where $c_j$ represents the number of times the user $j \in U$ has listened to this track, and $|U| = n$ is the number of all users. The similarity is determined by the *Pearson product-moment correlation* (see, e.g., [4]).
2. **Tag similarity.** With tag similarity, each track is represented as a vector $\beta_i = (l_1, \ldots, l_m)$, where $l_j \in [0, 100]$ represents the score to what extent the tag $l_j \in L$ "describes" this track, where $|L| = m$ is the number of all used tags. Since for tags we do not need to care for user-specific scales, the similarity is determined by the *cosine similarity measure* [4].
3. **Time similarity.** Every season has its music, e.g. there are typical songs for Valentine's day or Christmas. To capture this behavior, each track is represented as a vector $\gamma_i = (w_1, \ldots, w_{52})$, where $w_j$ is the number of times the track has been listened to in the $j$th week of the year, accumulated over all users. Similar to tag similarity, the *cosine similarity* is computed.

Every recommender generates a track-track similarity matrix $A^k, k \in \{1, 2, 3\}$. These are used for generating predictions for the active user.

### 2.2 Comparing Recommenders

In order to see how close the similarity estimates of all three recommenders are to each other, we implemented the following approach: First we iterated through every matrix $A^k, k \in \{1, 2, 3\}$, and created a new matrix $B^k$, where the entries of $B^k$, i.e., $b^k_{i,j}$, are defined as $z$-scores:

$$b^k_{i,j} = \frac{a^k_{i,j} - \overline{a}^k}{\sigma^k}, \tag{1}$$

where $\sigma^k$ is the standard deviation over all entries of matrix $A^k$, and $\overline{a}^k$ is the mean over all its entries. Now we build all three pairs of matrices $\{B^x, B^y\}$, where $x, y \in \{1, 2, 3\}$ and $x \neq y$, and calculate the matrix $C^{xy}$ for each, where $c^{xy}_{i,j} = |b^x_{i,j} - b^y_{i,j}|$. For each matrix $C^{xy}$ we now calculate one scalar value, which is the mean over all its entries. We denote this scalar by $\overline{c}^{xy}$ and it gives us an indication how close the similarity matrices produced by recommender $x$ versus $y$ are. The lower the value, the closer they are to each other.

The results show that the track and tag recommender are *closest* to each other ($\overline{c}^{12} = 0.93$), while the time recommender produces more deviating results from both the track ($\overline{c}^{13} = 1.33$) and the tag recommender ($\overline{c}^{23} = 1.67$). This well aligns with our conjectures before conducting this test.

### 2.3 Prediction Generation

The prediction score $p(u, t_{new})$ of a track $t_{new}$, which user $u$ has so far not rated yet, is computed based on a linear combination of the similarity scores of the three recommenders, denoted as $sim(t_{new}, t)$:

$$p(u, t_{new}) = \frac{\sum_{t \in Tracks} sim(t_{new}, t) \cdot r(u, t)}{\sum_{t \in Tracks} sim(t_{new}, t)} \quad (2)$$

The so far rated songs of a user are considered by $r(u, t)$, reflecting a rating of user $u$ for track $t$. If a user has only rated tracks of a few different artists so far, the majority of the recommendations might be from only one artist. To alleviate this undesirable behavior, for each artist at most two tracks are recommended to the user, working as a simple diversification means [5].

The weights of the three recommenders can be adjusted in the prototype and are set to equal weighting by default.

### 2.4 Adding Serendipity

For recommending serendipitous tracks, the last five positively rated tracks of the user are investigated. For *each* of these tracks, the last five users having listened to this track are selected, yielding (at most) 25 candidate users $U_{cand}$. The intersection of tracks the users in $U_{cand}$ have listened to is computed and the tracks with the highest overlap are chosen.

If there are tracks with the same overlap, the number of times the track has been listened to by *all* users gives the final rank. The tracks are inserted in the result list, with the constraint that the serendipity ranking's order is preserved.

The ratio of serendipitous to similarity-based tracks is set to 30% vs. 70% by default.

## References

1. Zhang, Y.C., Séaghdha, D.Ó., Quercia, D., Jambor, T.: Auralist: Introducing Serendipity Into Music Recommendation. In: Adar, E., Teevan, J., Agichtein, E., Maarek, Y. (eds.) WSDM, pp. 13–22. ACM (2012)
2. Burke, R.: Hybrid Web Recommender Systems. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) Adaptive Web 2007. LNCS, vol. 4321, pp. 377–408. Springer, Heidelberg (2007)
3. Adomavicius, G., Tuzhilin, A.: Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. IEEE Trans. Knowl. Data Eng. 17(6), 734–749 (2005)
4. Baeza-Yates, R.A., Ribeiro-Neto, B.A.: Modern Information Retrieval - The Concepts and Technology Behind Search, 2nd edn. Pearson Education Ltd., Harlow (2011)
5. Ziegler, C.N., McNee, S., Konstan, J., Lausen, G.: Improving Recommendation Lists Through Topic Diversification. In: Proceedings of the 14th International World Wide Web Conference, Chiba, Japan. ACM Press (May 2005)