# Could You Play That Song Again? – Reminding Users of Their Favorite Tracks Through Recommendations

Our Approach to the 2018 WSDM Cup Music Recommendation Challenge

Malte Ludewig
TU Dortmund
Germany
malte.ludewig@tu-dortmund.de

Dietmar Jannach
AAU Klagenfurt
Austria
dietmar.jannach@aau.at

## ABSTRACT

With the increasing popularity of unlimited music streaming services, traditional radio stations are more and more replaced by virtual stations that play seemingly endless personalized playlists created by a recommender system. While in many other application domains of recommenders the main task is to help the user discover new things, recommending and playing already familiar tracks is not uncommon in the music domain. A main challenge in that context is therefore to predict which tune to play again at which point in time.

In this paper, we present a hybrid approach to this reminding problem in music recommendation. The proposed method combines alternating least squares matrix factorization, mini-batch k-means clustering and gradient boosting decision trees. It was evaluated using data provided by the Taiwanese music service KKBOX and led to the 8th place in the 2018 WSDM Cup Music Recommendations Challenge.

## CCS CONCEPTS

**•Computing methodologies →Classification and regression trees; •Information systems →Recommender systems; Collaborative filtering; Clustering; Music retrieval;**

## KEYWORDS

Reminders; Music Recommendation; Cold Start

**ACM Reference format:**
Malte Ludewig and Dietmar Jannach. 2018. Could You Play That Song Again? – Reminding Users of Their Favorite Tracks Through Recommendations. In *Proceedings of WSDM'18 Cup, Los Angeles, California, USA, February 5-9, 2018,* 5 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

## 1 INTRODUCTION

Personalized item suggestions that are automatically provided by recommender systems are a common and popular feature of many of today's web shops and mobile applications. With an enormous quantity and diversity of items and the increasing popularity of unlimited streaming services, especially businesses in the music domain can profit from these type of systems. Their recommendations support users in different ways, e.g., by helping them discover new items, and at the same time also create business value, e.g., through increased customer satisfaction [3, 5, 6, 8, 9, 13].

To determine a set of suitable items for a user, recommendation approaches from the literature often employ collaborative filtering (CF) techniques, which are based on knowledge about the historical, collective behavior of a larger user group. The main computational task in most existing works is to help users find new items that – according to the historical data – they are not yet aware of. However, in the music domain, playing only tracks that are unknown to the user might not lead to the best user experience.

An analysis of listening logs of such online services in [10] in fact revealed that many users listen to their favorite tracks over and over. It can therefore be important for a successful service to remind users of already known tracks from time to time or to repeatedly recommend certain tracks in a given context like commuting or workout. The main tasks in such situations are therefore (a) to identify suitable tracks to play again and (b) to determine the right point in time for a potential repetition. If done properly, such recommendations can then lead to positive impacts on the business, as was shown in [11] for the e-commerce domain.

Common challenges of collaborative filtering techniques are the user and item cold start, which are situations when little information exists about individual new users or items. In our particular setting, where we are also interested to learn when to repeatedly recommend a track, we in addition face a "reminding cold start", in which only one single or a few log entries for an individual user-track combination exist.

Various techniques were proposed in the literature to deal with cold-start situations in collaborative filtering. In many cases, researchers rely on additional item features and combine the main CF technique with a content-based one in a hybrid approach. Furthermore, clustering users and items is another common technique applied for cold-start situations. In such approaches, new users are assigned to their nearest cluster and their recommendations are then determined based on the other users in that cluster.

In this paper, we adopt these existing strategies from the literature and propose a new hybrid method for "re-recommending" items to users: we capture collaborative signals with the help of matrix factorization techniques, address the user and item cold-start problems through clustering, and finally use decision trees to determine the probability that a user would like to listen to a certain, already known track. Technically, the proposed method is therefore a hybrid combination of alternating least squares (ALS)


*WSDM'18 Cup, Los Angeles, California, USA*

DOI: 10.1145/nnnnnnn.nnnnnnn

matrix factorization, mini-batch k-means clustering, and gradient boosting decision trees (GBDT). We applied our method to the music recommendation problem of the WSDM'18 Cup[1], where the specific task was to predict repeated track play events given log data from the Taiwanese music service KKBOX. The proposed method led to the eight place (among over 1,000 registered participants) on the final leaderboard.

## 2 PROBLEM DESCRIPTION

### 2.1 Computational Tasks

The computational problem in the 2018 WSDM Cup Music Recommendation Challenge, which was the guiding scenario of our research, is as follows. Given a user and a track, determine the probability that the user will listen to that same track again within the time span of one month. The main inputs to the task are

a) a set of unique user-track listening events, with
b) an indicator for tracks that were actually repeated, and
c) meta data for the user, the track, and also the context of the listening event.

Overall, the problem setting is quite different from common music recommendation scenarios in the literature, where the problem is often to find continuations to an ongoing listening session, with no particular emphasis on already known tracks [1]. The specific and seldom-covered challenges can be summarized as follows.

- *Reminding.* Determining whether or not and in which situations a user wants to listen to a track again.
- *Cold-start.* Ensuring good recommendation quality also for new users and new tracks.

The specific cold-start problem has similarities but is not exactly the same as the typical cold-start situation in the literature, where there is no information about the user or item at all. In our scenario, as the probability has to be predicted for a user-track listening event, with this event there always exists at least one implicit feedback data point. However, in such a "reminding cold start" situation there is, e.g., no exact information about how often and when a user has repeatedly listened to a track, which still makes it impossible to infer user-specific patterns from the data.

### 2.2 Data

The dataset that was provided in the WSDM Cup Challenge consists of a log of play events and a multitude of meta data about the users, the tracks, and the context of the play events. Each play log entry has a unique user and track identifier and a flag that indicates if the same user-track tuple has re-occurred in the data within the next month. The data therefore does not contain multiple listening events for the same user and track, but only this single indicator flag for each unique user-track pair. Furthermore, the data was sampled by the provider in a way that repeated and non-repeated tracks are better balanced.

Table 1 and Table 2 show an overview of additionally provided content and meta-data features. All features, except for the two dates that indicate the registration and expiration of a membership account, can be seen as categorical features and were mostly provided as strings with a special separation character when there were multiple values. Generally, the data was rather sparse in terms of the interaction matrix and many content feature entries were missing.

**Table 1: Content and meta data features for users and tracks.**

| Track | | User | |
|---|---|---|---|
| Track Id | Lyricist | User Id | Registration Date |
| Artist Name | Language Id | City | Expiration Date |
| Track Name | ISRC[2] | Age | |
| Genre | Track Length | Gender | |
| Composer | | Registration Method[3] | |

**Table 2: Meta data provided for each play event.**

| Event Feature | Brief Description |
|---|---|
| Source Tab | The KKBOX mobile application user interface is organized in tabs. The *source tab* feature indicates from which tab, e.g., *Search* or *My Library*, a listening event was triggered. |
| Source Screen Name | Similar to the *source tab*, but instead of the tab name the internal name of the application layout is used as the indicator. |
| Source Type | In a third refinement, the *source type* feature indicates in which kind of playback the event occurs, e.g., a single track, an album, a playlist, or a radio station. |

### 2.3 Evaluation Procedure

All performance evaluations were based on an offline experimental design. The given interaction dataset is split into training and test sets, where the first is used for model fitting and the second for model evaluation in terms of, e.g., the prediction accuracy.

*2.3.1 Competitive Evaluation.* Besides the unique interactions triples (training set) and the additional meta data, the participants were provided with about 2.5 million user-item-tuples without repetition indicators. These tuples form the *competitive* test set (25% of the training data). For each of these pairs, the task was to predict the repetition probability. Then, a corresponding "solution" file with those predictions had to be uploaded to a certain website. Specifically, the WSDM 2018 Cup was hosted on the data science community and competition platform Kaggle. The uploaded files were automatically evaluated in terms of the classification performance metric AUC[4]. The ranking of the participants was then posted on a public leaderboard.

*2.3.2 Local Evaluation.* Uploads to the Kaggle site were restricted to five daily submissions. This made it important to create an additional local evaluation environment, e.g., for parameter optimization. As the data was chronologically ordered, we applied a time-based splitting procedure. Similar to the train-test-ratio of

[1] https://wsdm-cup-2018.kkbox.events/
[2] International Standard Recording Code
[3] A categorical feature with five values; no further explanation was provided.
[4] https://www.kaggle.com/c/kkbox-music-recommendation-challenge

the *competitive* dataset, we chose the last 25% of the *competitive* training data as a *local* test set while the rest was used for training purposes. Additionally, to be able to quickly test new features or parameter configurations, we created an additional *sample* by cutting the *local* training set in half and using only the latter part of it. When creating these samples, considering time aspects was essential to ensure that our sampling procedure resulted in datasets and corresponding evaluation results that are representative of the *competitive* evaluation. The key characteristics of all datasets are shown in Table 3.

**Table 3: Dataset characteristics**

| Characteristic | Competitive | Local | Sample |
|---|---|---|---|
| Users | 34.4k | 30.8k | 26.5k |
| Tracks | 413.0k | 353.4k | 260.3k |
| Artists | 46.4k | 40.6k | 31.8k |
| Interactions | 9.93M | 7.38M | 3.69M |
| Class Ratio | 1.014 | 1.153 | 0.841 |
| Train/Test Ratio | 0.346 | 0.333 | 0.333 |
| Cold-start User | 7.19% | 7.38% | 6.09% |
| Cold-start Items | 12.20% | 10.37% | 8.67% |

## 3 TECHNICAL APPROACH

We determined the desired repetition probabilities by designing a supervised classification problem using a multitude of features that we engineered based on the given data. Furthermore, a number of features was created that originated from the application of clustering and collaborative filtering techniques on the data. As a learning method, we used gradient boosting decision trees (GBDT) [4], a method that often led to good results in classification competitions on Kaggle in the past.

In terms of the GBDT implementation, we tested different libraries, *LightGBM* by Microsoft[5] [12], *CatBoost* by Yandex[6], and *XGBoost*[7] [2]. Since all packages led to comparable accuracy results, we chose LightGBM because of its smaller memory footprint and higher computational efficiency.

### 3.1 Feature Engineering

We designed the following groups of features.

*3.1.1 Basic Features.* All the features listed in Table 1 and Table 2 can be directly fed into the GBDT. We encoded the categorical features as values from 0 to $n_c-1$ for $n_c$ different values (using $-1$ for unknown values). For multi-category strings, e.g., "pop|rock", we encoded the entire string as a category because selecting individual categories (e.g., the first or the most popular ones) did not lead to performance improvements. An exception is the genre, where we retained only the most popular one.

*3.1.2 Features based on Play Statistics.* In addition to the basic features, we created simple statistical features based on the user and track identifiers and given categorical features. These statistics were calculated for single identifiers or categories (e.g., play counts for each user or track) as well as for pairs of identifier and category (e.g., play count for a user for a given genre). Table 4 shows a generic overview of all features. Here, entries marked with a star utilize the class information in the training data and can thus only be generated for none cold start identifiers or categories, i.e., all values from the test set have to be present in the training set.

**Table 4: Features based on play statistics (overview).**

| Individual Features | Pair Features (Identifier and Category) |
|---|---|
| Count (All) | Count (All) |
| Count (Repeated)* | Count (Repeated)* |
| Ratio Repeated/All* | Ratio Repeated/All* |
| Norm. Count (Min-Max) | Norm. Count (Min-Max) |
| Share of All Actions | Share of All Actions |
| | Share of All by Identifier |
| | Share of All in Category |

*3.1.3 Latent Features.* Matrix factorization (MF) is a popular collaborative filtering recommendation technique. In the given problem setting, we considered the play events in the log data as implicit feedback signals to create the required user-item "rating" matrix and to determine latent vector representations for users, tracks, and artists, which were then directly added as features. We tested different MF techniques for implicit feedback matrices, in particular Bayesian Personalized Ranking (BPR) and an optimized ALS-based method[8] [7, 16, 17]. The best results were achieved with the ALS-based method using 32 as the number of latent factors.[9]

*3.1.4 Cluster-based Features.* To address the user and track cold-start problems, we employed a clustering technique to group the users and the tracks in the training data. New users and items in the test set were assigned to the "closest" existing cluster. The cluster identifier was then used as a new categorical feature, from which we could also derive additional statistical features, including those marked with a star in Table 4. We applied mini-batch k-Means with various values for $k$ and separately tested various representations for users and tracks, e.g., vectors from the implicit user-item "rating" matrix, only the provided content-based features, and finally the previously computed latent representations. The best results were achieved with exactly those latent factors and a cluster size of 25.

*3.1.5 Time-based Features.* The dataset includes some dates and was found to be chronologically ordered. We therefore also considered the time dimension in the feature engineering process.

We for example used the duration between the registration and expiration date to capture how attached a user is to the platform. Furthermore, we designed features to represent that individual songs might be trending only for a period of time or that users are more active in certain phases. Technically, we split the dataset in even parts and created additional statistical features for these parts as shown on the left side of Table 4, e.g., the number of play events for a user or song in the last fifth of the whole time period.

Overall, we engineered a rich variety of features to be potentially included in our model. Due to memory limitations and due to the fact that some features introduced noise (leading to lower accuracy

[5] https://github.com/Microsoft/LightGBM
[6] https://github.com/catboost/catboost
[7] https://github.com/dmlc/xgboost

[8] We used the implementations provided at https://github.com/bbc/theano-bpr and https://github.com/benfred/implicit with the default parameters.
[9] Much wider vectors could not be tested due to the given memory restrictions (32GB).

results), we did not include all of them in our final model. The full list of the 283 features can be found online.[10] Additionally, the full source code is provided on GitHub.[11]

### 3.2 Training, Stabilization, and Ensembles

In order to train, stabilize, and combine our model in an ensemble, we proceeded as follows. First, we manually fine-tuned the algorithms parameters for the entire training set, using the last 20% as a validation part. The best parameters were 0.1 as the learning rate, 1000 optimization epochs, a maximum tree depth of 15, and 256 leafs at most. Obviously, as the optimization objective we used the AUC. To stabilize the model and prevent it from over-fitting, we randomly left out some examples while keeping the full last 20% as the validation set. This approach was finally repeated in a randomized $n$-fold training phase to create an ensemble of $n$ models. The obtained probabilities were averaged and taken for the final submission. When increasing $n$, we could observe that the AUC results gradually improved following a logarithmic curve. Due to limitations regarding our computing power, we had to restrict $n$ to 10, so that our final submission was the result of 10 models.

## 4 RESULTS AND OBSERVATIONS

At the end of the competition, our team placed eighth among the over 1,000 registered teams. Our final AUC score was about 3.5% lower than that of the winners (0.721 vs. 0.748). In the following, we provide additional details about the relative importance of the used features, their impact on the AUC measure, and other observations.

**Table 5: Effectiveness of the different tested feature groups in the evaluation for all datasets.**

| Feature Group | Competitive | Local | Sample |
|---|---|---|---|
| Basic | 0.67284 | 0.67174 | 0.67997 |
| Basic + Time Features | 0.67477 | 0.67228 | 0.68151 |
| Basic + Cluster Features | 0.68939 | 0.69249 | 0.69462 |
| Basic + Play Statistics | 0.70885 | 0.70920 | 0.70709 |
| Basic + Latent Features | 0.71018 | 0.71187 | 0.71060 |
| Final | 0.71685 | 0.71673 | 0.71432 |
| 10-Fold Ensemble | 0.72095 | 0.72088 | 0.72021 |

Table 5 shows the impact of the feature groups from Section 3.1 and the $n$-fold random training approach from Section 3.2 in terms of the AUC score. We report the results both for the *competitive*, the *local* evaluation setup, and the *sample* based measurement. In general, all additional feature groups considerably increase the accuracy over the basic features that were directly given in the raw data. With 5.08% and 5.25%, the highest gains were achieved by incorporating the play statistics and, most importantly, the latent features. The cluster-based and time-based features, in contrast, led to only smaller improvements. The final model, which includes all feature groups, led to the overall highest single-model accuracy, which was again significantly improved by the 10-fold ensemble.

Table 6 shows the twenty most important features of our final model . Over 50% of these top 20 and nearly 80% of the top 50 features are latent ones all related to the user representations. Thus, given the overall improvement that can be achieved with latent features (see Table 5), this type of integration for content-based and CF techniques seems most promising for future extensions.

Due to limitations of the hardware that was available to us, we could only test latent representations of a limited size. The usage of larger, additional, or multiple representations, e.g., based on Word2Vec or DeepWalk [14, 15], therefore represents an area where further accuracy improvements could be obtained.

**Table 6: Importance of the top 20 features in the final model. N: Normalized by the identifier. N2: Normalized by the category. T5: Only for the current fifth of the dataset.**

| Feature | Importance | Feature | Importance |
|---|---|---|---|
| T5-User-Cnt (N) | 1.0000 | Latent-User (10) | 0.4097 |
| T5-User-Cnt | 0.7066 | Latent-User (19) | 0.4067 |
| User-SourceType-Cnt (N) | 0.5493 | Latent-User (30) | 0.4029 |
| User-ScreenName-Cnt (N) | 0.4855 | Latent-User (25) | 0.4004 |
| User-ScreenName-Cnt (N2) | 0.4584 | Latent-User (12) | 0.3990 |
| User-Artist-Cnt | 0.4390 | Latent-User (2) | 0.3990 |
| User-SourceTab-Cnt (N) | 0.4286 | Latent-User (38) | 0.3973 |
| User-SourceType-Cnt (N2) | 0.4270 | Latent-User (11) | 0.3971 |
| User-Artist-Cnt (N) | 0.4137 | Latent-User (1) | 0.3964 |
| Latent-User (6) | 0.4130 | Latent-User (5) | 0.3943 |

However, as in many other scenarios and also reflected by the top two features, time aspects also seems to play a very important role. Unfortunately, we discovered the chronological order of the data late in the process of the competition, which left little time for feature engineering in this regard. Thus, the extension of the time-based features could lead to a higher accuracy.

To other matters, the task that was given in this challenge is very interesting but also very specific and AUC here is just capturing the classification accuracy for this specific situation. Though the metric is suitable for the competition, it might not reveal the real value of the "recommendations" in a list. Furthermore, the questions arise, how and to what extent these reminders or predicted probabilities could or should be incorporated into full recommendation lists.

## 5 CONCLUSIONS

We presented a hybrid method to the problem of finding the right tracks to remind a user of in certain situations, which is a highly relevant problem in practice. For the given task, which was framed as a classification problem, we engineered a multitude of predictor variables, and used Gradient Boosting Decision Trees as a learning method that finally led to good results in the WSDM'18 Cup for music recommendations. Our next steps include both the design of additional features for the classification task as well as the exploration of algorithms that ultimately create recommendation lists that consist of an optimal mix of novel and familiar tracks.

## ACKNOWLEDGEMENTS

We thank the organizers of the WSDM'18 Cup and also the people at KKBOX for the opportunity to participate in this interesting and challenging competition.

[10] http://ls13-www.cs.tu-dortmund.de/homepage/wsdm-cup-2018-music/
[11] https://github.com/rn5l/wsdm-cup-2018-music

## REFERENCES


[1] Geoffray Bonnin and Dietmar Jannach. 2014. Automated Generation of Music Playlists: Survey and Experiments. *Comput. Surveys* 47, 2 (2014), 26:1–26:35.

[2] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. 785–794.

[3] Paolo Cremonesi, Franca Garzotto, and Roberto Turrin. 2012. Investigating the Persuasion Potential of Recommender Systems from a Quality Perspective: An Empirical Study. *Transactions on Interactive Intelligent Systems* 2, 2 (2012), 11.

[4] Jerome H. Friedman. 2000. Greedy Function Approximation: A Gradient Boosting Machine. *Annals of Statistics* 29 (2000), 1189–1232.

[5] Florent Garcin, Boi Faltings, Olivier Donatsch, Ayar Alazzawi, Christophe Bruttin, and Amr Huber. 2014. Offline and Online Evaluation of News Recommender Systems at Swissinfo.Ch. In *Proceedings of the 8th ACM Conference on Recommender Systems (RecSys '14)*. 169–176.

[6] Carlos A. Gomez-Uribe and Neil Hunt. 2015. The Netflix Recommender System: Algorithms, Business Value, and Innovation. *Transactions on Management Information Systems* 6, 4 (2015), 13:1–13:19.

[7] Y. Hu, Y. Koren, and C. Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. In *2008 Eighth IEEE International Conference on Data Mining (ICDM '08)*. 263–272.

[8] Dietmar Jannach and Gediminas Adomavicius. 2016. Recommendations with a Purpose. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16)*. 7–10.

[9] Dietmar Jannach and Kolja Hegelich. 2009. A Case Study on the Effectiveness of Recommendations in the Mobile Internet. In *Proceedings of the Third ACM Conference on Recommender Systems (RecSys '09)*. 205–208.

[10] Dietmar Jannach, Iman Kamehkhosh, and Lukas Lerche. 2017. Leveraging Multi-dimensional User Models for Personalized Next-track Music Recommendation. In *Proceedings of the Symposium on Applied Computing (SAC '17)*. ACM, New York, NY, USA, 1635–1642. DOI: http://dx.doi.org/10.1145/3019612.3019756

[11] Dietmar Jannach, Malte Ludewig, and Lukas Lerche. 2017. Session-based item recommendation in e-commerce: on short-term intents, reminders, trends and discounts. *User Modeling and User-Adapted Interaction* 27, 3 (2017), 351–392.

[12] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Advances in Neural Information Processing Systems 30 (NIPS '17)*. 3149–3157.

[13] Evan Kirshenbaum, George Forman, and Michael Dugan. 2012. A Live Comparison of Methods for Personalized Article Recommendation at Forbes.Com. In *Proceedings of the 2012th European Conference on Machine Learning and Knowledge Discovery in Databases (ECMLPKDD'12)*. 51–66.

[14] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26 (NIPS '13)*. 3111–3119.

[15] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: Online Learning of Social Representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '14)*. 701–710.

[16] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence (UAI '09)*. 452–461.

[17] Gábor Takács, István Pilászy, and Domonkos Tikk. 2011. Applications of the Conjugate Gradient Method for Implicit Feedback Collaborative Filtering. In *Proceedings of the Fifth ACM Conference on Recommender Systems (RecSys '11)*. 297–300.