Effective Nearest-Neighbor Music Recommendations

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

Iman Kamehkhosh
TU Dortmund, Germany
iman.kamehkhosh@tu-dortmund.de

Nick Landia
Dressipi, UK
nick.landia@dressipi.com

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

ABSTRACT

Automated recommendations for next tracks to listen to or to include in a playlist are a common feature on modern music platforms. Correspondingly, a variety of algorithmic approaches for determining tracks to recommend have been proposed in academic research. The most sophisticated among them are often based on conceptually complex learning techniques which can also require substantial computational resources or special-purpose hardware like GPUs. Recent research, however, showed that conceptually more simple techniques, e.g., based on nearest-neighbor schemes, can represent a viable alternative to such techniques in practice.

In this paper, we describe a hybrid technique for next-track recommendation, which was evaluated in the context of the ACM RecSys 2018 Challenge. A combination of nearest-neighbor techniques, a standard matrix factorization algorithm, and a small set of heuristics led our team KAENEN to the 3rd place in the "creative" track and the 7th one in the "main" track, with accuracy results only a few percent below the winning teams. Given that offline prediction accuracy is only one of several possible quality factors in music recommendation, practitioners have to validate if slight accuracy improvements truly justify the use of highly complex algorithms in real-world applications.

CCS CONCEPTS

• **Information systems → Nearest-neighbor search**; **Music retrieval**; **Collaborative filtering**; **Recommender systems**;

KEYWORDS

Music Recommendation; Automatic Playlist Continuation; Cold Start; Session-based Recommendation

ACM Reference Format:
Malte Ludewig, Iman Kamehkhosh, Nick Landia, and Dietmar Jannach. 2018. Effective Nearest-Neighbor Music Recommendations. In *Proceedings of Proceedings of the ACM Recommender Systems Challenge 2018 (RecSys Challenge '18).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3267471.3267474

RecSys Challenge '18, October 2, 2018, Vancouver, BC, Canada

ACM ISBN 978-1-4503-6586-4/18/10…$15.00
https://doi.org/10.1145/3267471.3267474

1 INTRODUCTION

Online streaming has become the predominant way of consuming music and represents the most profitable source in the music industry in recent years [6]. The provision of automated recommendations is a common feature of online music platforms. Besides recommendations that are based on long-term preferences and that are designed to help users discover new tracks or artists, *next-track* recommendations play an important role on these sites. Recommendations of this type are, for example, used to create personalized "radios" (i.e., virtually endless playlists) based on a few seed tracks. Such recommendations are also helpful during the process of playlist construction [8, 19, 24].

In terms of the problem setting, these next-track recommendations often share characteristics of *session-based recommendation* scenarios [20]. The particularity of this scenario is that there are no long-term preference profiles available, and the recommendations therefore have to be based on a small set of ongoing user interactions. In the music domain, this means that the starting point for the recommendations is a small list of tracks that the user has just listened to or added to a playlist.

Due to their high practical relevance, session-based recommendation problems have received increased attention in recent years. Technologically, a variety of approaches were proposed in the academic literature, see [2] and [20] for recent overviews. Often, these approaches rely on comparably complex computations based on Markov Chains, distributional embeddings, and deep neural networks, or they are encoded as mathematical optimization problems. Recent works, however, suggest that in some application domains, including e-commerce and music recommendation, also comparably simple computational schemes based on nearest neighbors can lead to high prediction accuracy [11, 17, 18, 25], often at much lower computational costs.

While complex methods are usually able to outperform such simple methods , the performance differences are often not very large. Unfortunately, however, it is not always fully clear if small improvements in terms of accuracy results obtained in offline experiments will lead to a better user experience [4, 7, 13]. Practitioners therefore have to decide if the use of more complex methods pays off, in particular as some of these advanced techniques require special hardware (GPUs) and substantial computational resources to be trained efficiently.

In this paper, we present a hybrid session-based music recommendation method, which we developed in the context of the ACM RecSys 2018 Challenge [3]. The method is based on a combination of nearest-neighbor techniques, a standard matrix factorization

algorithm, and a small set of heuristics. Despite its simplicity, the method reached the 3rd position in the *creative* and the 7th place among over 100 participants in the *main* track of the competition.

Generally, prediction accuracy, i.e., being able to identify tracks that a user will like, is only one of several factors that can contribute to the perceived quality and value of a music recommender. As previous research shows, there are factors like artist diversity, musical homogeneity, transitions between tracks, freshness and general popularity and several other aspects that can influence the user's perception of the recommendations [1, 12, 16, 22]. Therefore, we believe that methods like ours can represent a viable algorithmic basis in many practical applications, as they lead to competitive accuracy results at comparably low computational costs.

The rest of the paper is organized as follows. We summarize the problem setting of the ACM RecSys 2018 Challenge in Section 2. Afterwards, we outline our technical approach in Section 3 and discuss the result in Section 4. The paper ends with a summary and an outlook on future works.

2 PROBLEM DESCRIPTION

2.1 Computational Tasks

The computational problem in the RecSys Challenge 2018 relates to a specific type of music recommendation, namely, the *playlist continuation* problem. Given certain types of information about a partial playlist, the recommendation task is to generate a list of tracks that can be added to the list. The following types of information for the partial playlist can be available to the recommender.

a) Metadata information such as the playlist title, a description, and the number of followers on the streaming service Spotify,
b) a set of seed tracks, i.e., the tracks that are already in the playlist,
c) metadata information about tracks such as artist, album and the position of each track in the playlist.

The problem setting in this challenge is quite different from the traditional "matrix completion" research abstraction, since there are no long-term user profiles for personalization. Instead, the computational task is to determine a set of suitable musical tracks given a sequence of previous tracks.The specific challenges of this scenario can be summarized as follows.

- **Cold start.** For a subset of the playlists to be continued, only the title of the playlist (or the title and the first track of the playlist) is available. In such cases, commonly used collaborative recommendation techniques can not be directly applied.
- **No long-term or user information.** As – due to privacy concerns – user names are removed from the dataset, no personalized technique based on, e.g., the users' long-term listening behavior as in [10] can be applied.

2.2 Data

The main dataset provided by Spotify for the challenge comprises 1 million playlists and include playlist titles, track listings and other metadata. Table 1 shows an overview of the metadata features for the tracks and the playlists, respectively. The dataset is referred to as the Million Playlist Dataset (MPD).

[1] A boolean value which indicates if the owner allows other users to modify the playlist.

Table 1: Meta data features for playlists and tracks.

Playlist		Track	
Title	Playlist Id	Track name	Track Id
Description	No. of tracks	Artist name	Artist Id
No. of artists	No. of albums	Album name	Album Id
No. of followers	No. of edits	Track duration	
List of tracks	Playlist duration	Position (in list)	
Collaborative[1]	Modification date		

Table 2: Overview of all subtasks with 1,000 lists each.

Task	1	2	3	4	5	6	7	8	9	10
Tracks	0	1	5	5	10	10	25	25	100	100
Name	Yes	Yes	No	Yes	No	Yes	Yes	Yes	Yes	Yes
Order	––	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	No

Additionally, Spotify provided the participants of the challenge with a test or "competitive" dataset consisting of 10,000 playlists, where from each playlist a certain number of tracks was hidden. Those 10,000 playlists were assigned to one of ten specific subtasks as shown in Table 2. The defined subtasks are of different complexity, starting with 1,000 playlists for which only the name was known. For other subsets, a varying number of tracks was provided to reflect the different stages of the playlist creation process. A more detailed description of the data and tasks can be found in [3].

2.3 Evaluation Procedure

The evaluation in this year's ACM RecSys Challenge was designed as an offline experiment. Submissions could be made for two different tracks, namely, the main and the creative track. For the main track, the participants were instructed to rely solely on the provided playlist dataset to train the models. For the creative track, all sorts of additional and external metadata could be used to improve the recommendation quality.

2.3.1 Competitive Evaluation. During the competition, participants could submit playlists consisting of 500 tracks for each of the 10,000 playlists of the competitive dataset, both for the main and the creative track. Each participating team was allowed to submit a solution file with their recommendations once a day for both of the tracks. Those solutions could be uploaded to a specific webpage[2] in a specific format and the uploaded files were then automatically evaluated in terms of three different performance metrics. A ranking of the current solutions of all participants was posted on a public leaderboard on a daily basis. The rank was determined by the Borda count over the rankings for the following three accuracy measures.

- *Precision@N:* For a playlist with N hidden tracks, precision is calculated as the ratio of those hidden tracks that also appear in the first N tracks of the submitted top 500 list.
- *NDCG@500:* The Normalized Discounted Cumulative Gain is a common metric in the field of information retrieval that takes the position of the correctly predicted tracks into account.

[2] https://recsys-challenge.spotify.com/

Table 3: Dataset Characteristics

Characteristic	COMPETITION	LARGE1	LARGE2	SMALL
Playlists	1M	990k	990k	99k
Tracks	2.3M	2.3M	2.3M	689k
Artists	296k	296k	296k	111k
Entries	66.3M	65.4M	65.4M	6.53M
Avg. Entries per List	66.346	66.017	66.026	65.946
Testlists	10k	10k	10k	1k
Avg. Entries	98.016	99.002	98.017	98.321
Avg. Hidden	69.916	70.902	69.917	70.221

- *Clicks@500:* The last metric was more specifically designed for the given task with respect to aspects of the Spotify user interface. As the recommendations are arranged on multiple pages with 10 entries per page, the *Clicks* metric should reflect how often a user has to navigate to the next page until a relevant track is found. Thus, the possible values range from 0 to 51 clicks, where 51 is chosen when no suitable track was included in the recommendations.

2.3.2 Local Evaluation with Additional Samples. As the submissions were restricted to one upload per day, it was necessary to create a suitable local evaluation setup to, for example, quickly test new methods or optimize parameters. Hence, we created a number of additional test samples that should represent the COMPETITION set in a good way. For each list in COMPETITION we sampled playlists of similar size from the MPD and hid the corresponding number of tracks. Furthermore, we ensured that no item cold-start situations occurred. Besides two samples with 10,000 playlists, termed LARGE1 and LARGE2, we also created a smaller sample SMALL by randomly sampling 100,000 playlists of the MPD and selecting a test set of 1,000 playlists with 100 lists per task from those. The key characteristics of all datasets are shown in Table 3.

3 TECHNICAL APPROACH

For those playlists that include tracks, we determined the top recommendations and their ranking by combining a number of established techniques. Specifically, we included item-based collaborative filtering, k-nearest-neighbor techniques, and an implicit feedback matrix factorization approach as will be described in more detail in Section 3.1. For the other playlists, where only the name is known, we use two simple string matching strategies.

The individual methods or recommendations, respectively, were finally combined in a hybrid approach, which is based on weighted combinations and switching between different strategies depending on the number of seed tracks and the occurrence of a playlist name.

Our submissions to the main and the creative track were both based on the same recommendation scheme. For the creative track, we furthermore implemented a re-ranking approach that utilizes additional track metadata that we retrieved from the Spotify API. Generally, in order to make our research reproducible, we publish the full source code on GitHub.[3]

[3]https://github.com/rn5l/rsc18/

3.1 Track-based Approaches

We adapted and customized the following track-based methods to determine a ranked list of candidate tracks.

3.1.1 Item-based Collaborative Filtering (ITEM-CF). While traditional item-to-item approaches are conceptually simple and often outperformed by sophisticated academic techniques, they are often employed in practice as they lead to good performance and also scale well [5, 15].

The ITEM-CF method used in our hybrid approach is generally based on individual item similarities but still considers all given seed tracks of a playlist. The similarity between two items is, as usual, determined by the number of their co-occurrences in other sessions. In terms of the implementation, each track is represented as a binary vector over all playlists, i.e., the entry for a specific playlist is set to 1 if the item is contained in it and to 0 otherwise. Thus, the similarity between tracks can then be computed, e.g., determining the cosine similarity, and recommendations for a single tracks are formed by the top k most similar tracks ordered by similarity.

Approaches based on item co-occurrences can lead to a relatively high popularity bias, i.e., they often tend to focus on popular items. As a technical enhancement, we therefore compute the inverse document frequency (IDF) for all tracks regarding all playlists with $idf(t) = log_{10}(\frac{|P|}{|P_t|})$, where P is the set of all (training) playlists and P_t is the set of playlists containing track t. The final list of suggestions is then determined by combining the recommendations for each single seed track weighted with the corresponding IDF of that track. Thus, more importance is given to the less popular tracks in a playlist, which resulted in a better accuracy in our experiments.

In terms of the implementation, all similarity values were precomputed for the competition set to ensure fast recommendation times.[4]

3.1.2 Session-based Nearest Neighbors (S-KNN). As recommending from short playlists is comparable to the problem of providing suggestions in the beginning of an e-commerce browsing session, we furthermore relied on a session-based recommendation approach that led to promising results for the e-commerce and music domain in our previous research [11].

The S-KNN approach does not create recommendations for each seed track individually. Instead, it considers the entire playlist to search for the most similar playlists in the training data. Tracks from those neighboring playlists are then ranked to create the recommendation list (see also [2, 8, 14]). First, we determine the k most similar playlists N_p for a given playlist p by applying a suitable similarity measure on binary vectors over the item space [2], i.e., the index for a specific track is set to 1 if the item is contained in a playlist and to 0 otherwise. When considering a playlist p, the neighbors N_p, and the similarity function $sim(p_1, p_2)$ for two playlists p_1 and p_2, the ranking for each track t is determined in the following way:

$$rank_{\text{S-KNN}}(t, p) = \Sigma_{n \in N_p} sim(p, n) \cdot 1_n(t) \cdot idf(t) \qquad (1)$$

Here, $1_n(i)$ represents an indicator function that returns 1 if neighbor n contains the track t and 0 otherwise. Again, our goal was

[4]We use and extended an implementation published as ItemKNN at https://github.com/hidasib/GRU4Rec.

to give more importance to less popular items and in general to reduce the popularity bias of this method, we added the tracks' IDF ($idf(t)$) value to the ranking calculation. Applying the cosine similarity measure with a neighborhood size of 2000 led to good results in our experiments.

Given the current playlist p, it is in practice probably too time consuming to scan the full set of existing playlists. Hence, our implementation, as explained in more detail in [11], relies on precomputed in-memory index data structures and on neighborhood sampling to ensure fast prediction times. First, playlists are only considered as neighbor candidates if they contain at least one track from the current playlist. Second, we introduce a sampling parameter m that focuses on those m playlists which were most recently edited. In our final submission, we did *not* apply this sampling to leverage all available information. In other submissions, considering only the most recently edited 5000 playlists did however not lead to a major decrease in accuracy. In fact, the *Clicks* metric did even improve with $m = 5000$ in some cases.

IDF Extension (IDF-KNN). As adding the IDF values to the ranking calculation of S-KNN resulted in a leap in terms of the accuracy, we implemented an alternative to the method that includes the TF-IDF concept when calculating playlist similarity scores. Here, instead of representing playlists as binary vectors over the item space, we represent them with a slightly modified TF-IDF vector.

To calculate the TF-IDF score for each track and playlist we use the standard formulation $tfidf(p,t) = tf(p,t) \cdot idf(t)$ with a small adjustment in the term frequency. For a playlist p and track t the TF part is calculated as

$$tf(p,t) = \frac{1}{|p| + s} \tag{2}$$

where the numerator is 1 because each track should only appear once in a playlist, and s is a parameter that was introduced to address playlists that have a small number of tracks. In practice, the parameter prevents playlists with only a few tracks from having very high TF weights. Through experimentation, we have found a value of $s = 50$ to work well for this dataset.

Although the technical playlist representation differs from the original S-KNN formulation, the ranking is still calculated in a very similar way:

$$rank_{\text{IDF-KNN}}(t,p) = \Sigma_{n \in N_p} sim(p,n) \cdot 1_n(t) \tag{3}$$

Here, the distance is again determined with the cosine similarity measure. In contrast, however, the $idf(p,t)$ term from Equation 3 is left out and, with $k = 1000$, a fewer number of neighbors led to the best results.

3.1.3 Matrix Factorization (ALS-MF). As the last track-based component of our hybrid, we included a matrix factorization (MF) technique as a popular collaborative filtering recommendation approach. In our setting, the playlists correspond to the users and the contained tracks are implicit feedback signals to construct a sparse binary user-item "rating" matrix. To rank items for a user or playlists, respectively, we tested two implicit feedback MF techniques, in particular Bayesian Personalized Ranking (BPR) and an optimized ALS-based method[5] [9, 21, 23]. Experimentally, we found good accuracy results with 300 latent factors in 10 iterations and the regularization parameter set to 0.08 when applying the ALS-based method.

However, due to the fact that we have to provide recommendations for a newly created playlist, the situation maps to a user cold-start scenario for the MF techniques. We have overcome this problem by constructing the user latent factors from all item latent factors corresponding to the seed tracks of a playlist. Instead of using the mean of all item factors, we, again, incorporate IDF weights to emphasize on less popular tracks. Thus, the rank of a track t is calculated the following way:

$$rank_{\text{ALS-MF}}(t,p) = TL_t \cdot \left(\frac{1}{\sum_{s \in p} idf(s)} \sum_{s \in p} idf(s) \cdot TL_s\right) \tag{4}$$

where TL are the tracks' latent factors and the latter part combines all factors for playlist p as the weighted arithmetic mean with the IDF. The final ranking score is then calculated as the dot product. Our experiments did not show a major decrease in accuracy compared to classic MF when applying this procedure.

3.2 Name-based Approaches

Besides the track-based approaches, we propose two simple name-based techniques for the special recommendation scenario, in which we only know the playlist name but have not a single seed track to base recommendations on.

3.2.1 String Matching (STRING-MATCH). With STRING-MATCH, we implemented a straightforward approach to find suitable tracks only based on the name of a playlist. We simply collect all playlists with the most similar name from the MPD and recommend the tracks from those playlists sorted by the number of occurrences in those lists in descending order. In terms of our implementation, to improve the matching quality and make the approach more flexible, we first preprocessed the playlist names with the Natural Language Toolkit.[6] Specifically, we tokenized the strings and applied the Porter stemming algorithm. In case no exact match is found in the set of known playlist names, we replaced the name by the most similar known name based on the Levenshtein distance.[7]

3.2.2 Title Factorization (TITLE-MF). Our second name-based technique, TITLE-MF, extends STRING-MATCH and aims to include tracks from playlists with similar names based on a MF technique. Again, the NLTK toolkit is used to preprocess the list names. Then, a user-item "rating" matrix is constructed with the unique names as the users, the tracks as the items and number of co-occurrences as the rating. After factorizing the matrix, again, with the same ALS-based approach, we can use the names' latent factors to find similar names in terms of the tracks contained in the playlists. In the prediction phase, we in contrast to STRING-MATCH use the k most similar names instead of only one. For each similar name, the number of track occurrences is counted and multiplied with the name similarity. Finally, the weighted counts are combined to a single recommendation list by summing up the individual scores.

[5] The implementations are provided at https://github.com/benfred/implicit.

[6] https://www.nltk.org/

[7] We used the library available at https://github.com/seatgeek/fuzzywuzzy.

Table 4: Effectiveness of the tested techniques and hybrids in the evaluation for all datasets. For COMPETITION, no technique was tested individually and the results in the first group were obtained a hybrid combination with STRING-MATCH.

	COMPETITION			LARGE1			LARGE2			SMALL		
	RP	NDCG	Clicks	RP	NDCG	Clicks	RP	NDCG	Clicks	RP	NDCG	Clicks
S-KNN	0.199	0.361	2.340	0.182	0.367	3.086	0.181	0.366	3.032	0.198	0.398	2.253
ITEM-CF	0.202	0.365	2.393	0.187	0.374	3.066	0.185	0.372	3.055	0.205	0.379	2.571
IDF-KNN	0.204	0.368	2.338	0.188	0.374	3.067	0.187	0.374	3.046	0.208	0.411	2.302
ALS-MF	––	––	––	0.151	3.501	0.300	0.150	0.299	3.539	0.184	0.375	2.544
WEIGHTED	0.208	0.373	2.260	0.192	0.381	2.959	0.191	0.380	2.935	0.209	0.410	2.265
STRING-MATCH	––	––	––	0.096	0.215	6.925	0.093	0.207	7.182	0.097	0.209	6.828
TITLE-MF	––	––	––	0.100	0.223	7.311	0.097	0.218	7.514	0.102	0.223	7.617
WEIGHTED	––	––	––	0.100	0.224	7.021	0.097	0.218	7.251	0.101	0.220	7.012
SWITCH	0.209	0.375	2.122	0.201	0.397	1.726	0.199	0.395	1.775	0.214	0.416	1.915
META-RERANK	0.209	0.375	2.116	0.201	0.398	1.724	0.199	0.395	1.765	0.214	0.416	1.910

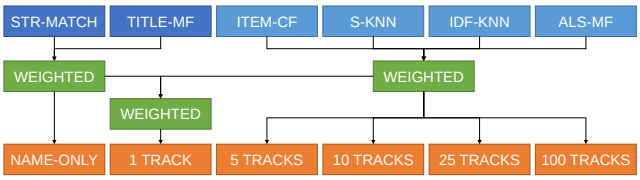

Figure 1: Overview of our final hybrid composition.

3.3 Hybridization

In order to improve our recommendations, we combined all of the previously explained methods, using three types of hybrid strategies. Figure 1 shows an overview of the overall architecture.

- *Filling:* The first type of hybrid we used in some cases to generate the final recommendation lists is the "filling" hybrid. Whenever one of the methods, mostly the STRING-MATCH approach, did not return enough tracks to recommend (500) the list was filled up with the most popular tracks from the training data.
- *Weighted:* These hybrids combine the recommendation lists of two or more methods based on the ranking scores with specific weights. As the scale of those scores can differ largely, all scores were min-max normalized for the top 500 tracks. Furthermore, we optimized the weights on the LARGE1 sample. Besides an individual combination of all name-based and all tracks-based methods, we also created a weighted hybrid of those two combinations, which showed superior performance for playlists with a name and one single seed track.
- *Switching:* The switching hybrids in general alter between different techniques or models depending on some specified context. In our scenario, the context is given as the number of seed tracks. For name-only playlists, we used a weighted hybrid of both name-based techniques. When there was more than one seed track, we applied the weighted hybrid of all track-based methods. Finally, when there was one seed track and a playlist name, the weighted combination of both hybrids was used.

3.4 Meta Re-Ranking for the Creative Track

With the goal of further improving the quality of our recommendations for the *creative* track, we implemented a re-ranking strategy based on track meta data. To this purpose, we crawled musical features of all tracks in the MPD using Spotify's API.[8] The retrieved features include, e.g., the loudness, the tempo or the energy of a track.

If the mean standard deviation of the features of the seed tracks was below a certain threshold, the rank was adjusted as follows:

$$rank_{\text{META-RERANK}}(t,p) = rank_{\text{BASE}}(t,p) + rank_{\text{BASE}}(t,p) \cdot sim(t,p) \quad (5)$$

Here, $sim(t,p)$ represents the cosine similarity between the tracks features and the mean of the playlists features.

4 RESULTS AND OBSERVATIONS

At the end of the competition, our team placed 7$^{\text{th}}$ in the main and 3$^{\text{rd}}$ in the creative track among the over 100 registered teams. Our final accuracy scores were 5% to 7% lower than those of the winners (0.225 vs. 0.209, 0.396 vs. 0.375, and 1.822 vs. 2.122). In the following, we discuss the relative effectiveness of the used methods, their sometimes conflicting impact on the different accuracy measures, and other observations.

First, regarding the track-based techniques, IDF-KNN can be identified as the most accurate method across most samples and also for the competitive set. ITEM-CF is usually more precise than S-KNN, but the latter method manages to rank the items in a better way, which in general results in a better NDCG and *Clicks* metric. ALS-MF is the overall worst method regarding all metrics. However, when combining the different methods in a weighted hybrid, the performance increases when the matrix factorization approach is included. Again, the hybrid approach leverages the strengths of all methods. In most cases, all performance values increase. Only for the *Clicks* rank S-KNN remains the best approach for the small sample.

Second, regarding the name-based approaches, TITLE-MF always improved the NDCG and the precision metric. At the same time, the *Clicks* metric usually worsened significantly. For the LARGE1 sample, for example, the NDCG and precision are improved by 4%, while there was a 2% drop in terms of the *Clicks* metric. This might be due to the fact that TITLE-MF has a higher popularity tendency, which is advantageous to find more suitable tracks. However, some of the relevant tracks that are found by STRING-MATCH based on

[8]The API was accessed with the library spotipy (https://github.com/plamere/spotipy).

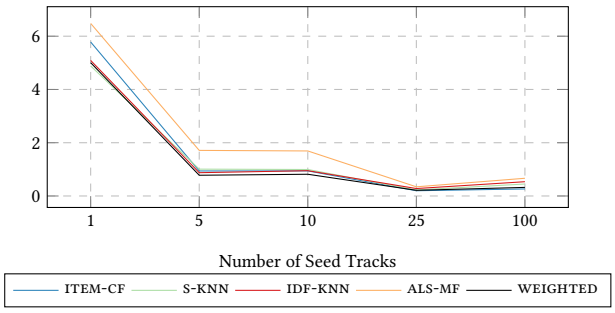

Figure 2: Clicks@500 per number of seed tracks for LARGE2.

the name are pushed back in the list as they are not very popular. The weighted hybrid balances the two methods and improves the *Clicks* metric considerably over TITLE-MF for all samples. At the same time, the NDCG and precision are not affected heavily. We tested weights from 0.9 to 0.1 and found 0.5 to be the overall best configuration in most cases.

Figure 2 shows a comparison of the track-based approaches in terms of the *Clicks* metric for the different numbers of seed tracks given in the test set, i.e., 1, 5, 10, 25, and 100. It is interesting to see that S-KNN is the most accurate technique for a single seed track while ITEM-CF works best for 100 seed tracks. Also, this behavior is reproducible for most samples and metrics. Our hybrid approach manages to balance this effect well, but ultimately it might be a better approach to optimize a weighted hybrid depending on the number of provided seed tracks.

Finally, our experiments show small but constant improvements when applying the metadata re-ranking approach, in particular in terms of the *Clicks* metric. We believe that this re-ranking strategy can be further improved, but decided to focus on other aspects in the competitions.

5 SUMMARY

We presented a hybrid method to the problem of finding continuations for playlists, which is a highly relevant problem in practice. Overall, a combination of relatively simple neighborhood-based approaches led to competitive accuracy results in the ACM RecSys Challenge 2018 for music recommendations.

We believe that our insights are particularly helpful for practitioners. They show that good accuracy results can be achieved with conceptually simple and computationally light-weight methods. Generally, while prediction accuracy is an important factor in music recommendation, we believe that future works should be directed more towards a balanced approach that considers multiple potential quality factors in parallel.

ACKNOWLEDGMENTS

We thank the organizers of the RecSys Challenge 2018 for the opportunity to participate in this interesting and challenging competition.

REFERENCES

[1] Wietse Balkema and Ferdi van der Heijden. 2010. Music Playlist Generation by Assimilating GMMs into SOMs. *Pattern Recognition Letters* 31, 11 (2010), 1396–1402.

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

[3] Ching-Wei Chen, Paul Lamere, Markus Schedl, and Hamed Zamani. 2018. RecSys Challenge 2018: Automatic Music Playlist Continuation. In *Proceedings of the 12th ACM Conference on Recommender Systems (RecSys '18)*.

[4] Paolo Cremonesi, Franca Garzotto, Sara Negro, Alessandro Papadopoulos, and Roberto Turrin. 2011. Comparative Evaluation of Recommender System Quality. In *CHI '11 Extended Abstracts on Human Factors in Computing Systems (CHI EA '11)*. 1927–1932.

[5] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, and Dasarathi Sampath. 2010. The YouTube Video Recommendation System. In *RecSys '10*. 293–296.

[6] Joshua P. Friedlander. 2017. News and Notes on 2017 Mid-Year RIAA Revenue Statistics. Online. https://www.riaa.com/wp-content/uploads/2017/09/RIAA-Mid-Year-2017-News-and-Notes2.pdf

[7] 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.

[8] Negar Hariri, Bamshad Mobasher, and Robin Burke. 2012. Context-aware Music Recommendation Based on Latenttopic Sequential Patterns. In *Proceedings of the Sixth ACM Conference on Recommender Systems (RecSys '12)*. 131–138.

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

[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)*. 1635–1642.

[11] Dietmar Jannach and Malte Ludewig. 2017. When Recurrent Neural Networks Meet the Neighborhood for Session-Based Recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems (RecSys '17)*. 306–310.

[12] Iman Kamehkhosh, Dietmar Jannach, and Geoffray Bonnin. 2018. How Automated Recommendations Affect the Playlist Creation Behavior of Users. In *Proceedings of the Workshop on Intelligent Music Interfaces for Listening and Creation at IUI 2018*.

[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] Lukas Lerche, Dietmar Jannach, and Malte Ludewig. 2016. On the Value of Reminders Within E-Commerce Recommendations. In *Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization (UMAP '16)*. 27–35.

[15] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing* 7, 1 (2003), 76–80.

[16] Beth Logan. 2002. Content-Based Playlist Generation: Exploratory Experiments. In *Proceedings of 3rd International Conference on Music Information Retrieval (ISMIR '02)*. 295–296.

[17] Malte Ludewig and Dietmar Jannach. 2018. Evaluation of Session-Based Recommenation Algorithms. (2018). arXiv:1803.09587

[18] Malte Ludewig, Michael Jugovac, and Dietmar Jannach. 2017. A Light-Weight Heuristic Approach to Recipient Determination When Recommending New Items. In *Proceedings of the ACM RecSys Challenge 2017 Workshop*. Como, Italy.

[19] Brian McFee and Gert R. G. Lanckriet. 2012. Hypergraph Models of Playlist Dialects. In *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR '12)*. 343–348.

[20] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. 2018. Sequence-Aware Recommender Systems. *Comput. Surveys* 51 (2018), 1–36. Issue 4.

[21] 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.

[22] Andy M. Sarroff and Michael Casey. 2012. Modeling and Predicting Song Adjacencies In Commercial Albums. In *Proceedings of the 9th Sound and Music Computing Conference (SMC '12)*.

[23] 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.

[24] Andreu Vall, Matthias Dorfer, Markus Schedl, and Gerhard Widmer. 2018. A Hybrid Approach to Music Playlist Continuation Based on Playlist-Song Membership. 1805.09557 (2018). arXiv:1805.09557

[25] Koen Verstrepen and Bart Goethals. 2014. Unifying Nearest Neighbors Collaborative Filtering. In *Proceedings of the 8th ACM Conference on Recommender Systems (RecSys '14)*. 177–184.