Towards Seed-Free Music Playlist Generation

Enhancing Collaborative Filtering with Playlist Title Information

Jaehun Kim
TU Delft
Delft, Netherlands
J.H.Kim@tudelft.nl

Minz Won[*]
Universitat Pompeu Fabra
Barcelona, Spain
minz.won@upf.edu

Cynthia C. S. Liem
TU Delft
Delft, Netherlands
C.C.S.Liem@tudelft.nl

Alan Hanjalic
TU Delft
Delft, Netherlands
A.Hanjalic@tudelft.nl

ABSTRACT

In this paper, we propose a hybrid Neural Collaborative Filtering (NCF) model trained with a multi-objective function to achieve a music playlist generation system. The proposed approach focuses particularly on the cold-start problem (playlists with no seed tracks) and uses a text encoder employing a Recurrent Neural Network (RNN) to exploit textual information given by the playlist title. To accelerate the training, we first apply Weighted Regularized Matrix Factorization (WRMF) as the basic recommendation model to pre-learn latent factors of playlists and tracks. These factors then feed into the proposed multi-objective optimization that also involves embeddings of playlist titles. The experimental study indicates that the proposed approach can effectively suggest suitable music tracks for a given playlist title, compensating poor original recommendation results made on empty playlists by the WRMF model.

CCS CONCEPTS

• **Information systems** → **Collaborative filtering**; **Recommender systems**; **Music retrieval**; • **Computing methodologies** → *Neural networks*;

KEYWORDS

Hybrid Recommender System; Music Playlist Generation; Collaborative Filtering; LSTM; WRMF; Multi-Objective Function;

ACM Reference Format:
Jaehun Kim, Minz Won, Cynthia C. S. Liem, and Alan Hanjalic. 2018. Towards Seed-Free Music Playlist Generation: Enhancing Collaborative Filtering with Playlist Title Information. In *Proceedings of the ACM Recommender Systems Challenge 2018 (RecSys Challenge '18), October 2, 2018, Vancouver, BC, Canada.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3267471.3267485

[*]This research was partially conducted during the author's internship at Naver Corp.

1 INTRODUCTION

Inspired by the *2018 RecSys Challenge* [2], in this paper, we propose an approach for automatically suggesting tracks to be included in a given playlist. For this problem, various amounts of information may be present: in many cases, the playlists will already be populated with several music tracks (further referred to as "seed tracks"), and new tracks for music playlist continuation should be suggested. However, in this paper, we are interested in the most extreme completion case, in which a playlist is identified by a title, but contains no seed tracks. This "seedless" empty playlist condition can be seen as a variant of the *cold-start* problem, well-known in the domain of Recommender Systems (RS), which especially is challenging for Collaborative Filtering (CF)-based approaches [10, 13, 15].

To solve this problem, we employ the playlist title as external information to transform the playlist into a latent factor vector representation, which is used to allow the system to give track suggestions. However, due to the noisiness of the given titles, applying common word-level approaches such as Word2Vec [12] will be challenging. Alternatively, as a simpler solution, we employ character-level N-grams, which are well-known for their effectiveness in text processing [1, 8].

Further, to encode the N-gram feature, we apply Neural Collaborative Filtering (NCF), which has been attracting attention of researchers recently mostly for its effectiveness, as reported in various RS works [17, 18]. More specifically, we employ a Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM) cells [4], which is inherently designed for sequential data such as textual data [6, 16].

In the rest of the paper, we first shortly describe the empirical rationale behind our current approach, based on preliminary data analysis (Section 2). Then, we present the details about the proposed approach (Section 3) and the experimental setup for internal evaluation (Section 4), followed by the obtained results (Section 5). Finally, we discuss the results and future work in the conclusion of the paper (Section 6).

2 PRELIMINARY APPROACH & DATA ANALYSIS

2.1 Simulation of Testing Condition

In the RecSys challenge, the Million Playlist Dataset (MPD) is the official data offered for training and validation. For final evaluation of challenge submissions, a so-called *Challenge Set* is released, offering playlists to be completed according to 10 "challenge categories", ranging from playlists with a title, but without any seed, up to playlists with a title and 100 random tracks. A full definition of the 10 challenge categories can be found on the challenge website [1].

As the ground truth for the Challenge Set is unknown during the challenge's running time, during development, we could not directly evaluate our system on this set. Instead, we drew internal training/validation splits from the MPD, where the test data was modified to meet the characteristics of the Challenge Set.

Variable	Description	Value
I^{tr}	number of training playlists	500,000
I^{vl}	number of validation playlists	5,000
J	number of tracks	~1,600,000
L	number of unique N-grams	~31,000

Table 1: Details on the data used for the experiments. The numbers of tracks J and unique N-gram tokens L is approximate, due to randomization effects across the splits.

The split process starts with the random selection of a desired amount of playlists for training and validation out of the MPD. Similar to the challenge's original setup, we keep the ratio between the size of training and validation set as 100:1. Especially, the playlists for validation are chosen and processed to mimic characteristics of the Challenge Set's various categories in terms of the number of hold-out tracks and the number of seed tracks per category. More details can be found in the Table 1. This process is repeated for three random folds.

For our challenge submission, we combined the MPD and the Challenge Set to build a total assignment matrix that contains all the playlists and the tracks for the training of submitted system.

2.2 Matrix Factorization Performance

Using the data as described in the previous section, we performed a preliminary analysis of how successful a baseline approach would be on the various challenge categories. For this, we employed several Matrix Factorization (MF) techniques such as Singular Value Decomposition (SVD) and Weighted Regularized Matrix Factorization (WRMF) [5]. MF is known as one of the most popular and effective solutions in the RS problem domain, with relatively little computational overhead [10]. After an initial experiment comparing the various factorization options, we decided to use WRMF for further development. The detailed result for this preliminary comparison can be found in Section 5.1.

WRMF is simple and well-developed for recommendation tasks in which the user-item interaction is implicitly given, such as users' listening counts or track playing time [5, 7, 10]. While playlist-track membership is rather explicit compared to "true" implicit feedback, we applied this algorithm, considering the fact that unselected items are still ambiguous: unselected tracks should not necessarily be considered as negative tracks. We trained and validated WRMF according to the setup as specified in Section 2.1.

Figure 1 visualizes the average NDCG@500 performance for WRMF over the three folds mentioned in Section 2.1. As can be

[1]https://recsys-challenge.spotify.com/challenge_readme

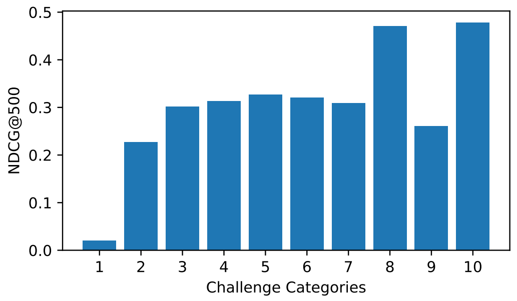

Figure 1: NDCG@500 over the 10 challenge categories using WRMF. We applied the best hyper-parameter setup as found through the procedure discussed in Section 5. It indicates the first case, where there is no seeds shows significantly worse performance compared to other cases.

seen, challenge category 1 (the "no-seed" case) yields substantially worse performance than other cases. However, this is expected, as the current factorization setup did not learn any playlist factors for empty playlists.

2.3 Title-Based Opportunities

To alleviate the problem of missing information in an empty playlist, one of the most straightforward ways is to exploit external information to characterize the playlist. The challenge data offers two additional information sources of potential use: the playlist title and the number of tracks that are held out for the validation. In our approach, we intentionally only choose the playlist title, as in real-world applications, it would be unrealistic for users to explicitly indicate the number of held-out tracks, while it would seem logical a user would indicate a playlist title before filling the playlist.

Initial inspection of actual titles in the MPD reveal a few key characteristics of the data, which makes it non-trivial to employ standard word-level approaches such as WordNet or Word2Vec:

- more than 60% of titles only contain single words and about 92% of them contain less than two words, which reflects that playlist titles should be treated as words rather than sentences
- special characters (`music!!! _**, //happy place//`)
- repeated characters (`yaaaas, summerrrr`)
- shortened or abbreviated words (`biebs!!!!, favss, loml`)
- symbolic expressions such as emojis (☺☺☺)
- multiple languages appear (`otoño, 電台收藏, アニメ`)

For these reasons, we decided to employ a character-level approach to exploit the textual information. More specifically, we employ the character N-gram, which uses a short sequence of N successive characters as a low-level token.

2.4 Learning Playlist Factors From Title Information

Ideally, for the challenge, we want to find a recommender f that can be defined as follows:

$$\tilde{s}_{i,j} = f(\textit{'information on the track j', 'information on the playlist i'}) \tag{1}$$

where $\tilde{s}_{i,j}$ is the approximated membership score of track j given playlist i. MF models can be considered as a realization of this model, using only the integer index of playlist and track entities, and finding latent embeddings corresponding to these indices:

$$\tilde{s}_{i,j} = f(v_j, u_i) \tag{2}$$

where u_i and v_j refer to the latent factors for playlist i and track j, respectively. As discussed in Section 2.2, however, this framework is incapable of dealing with "seedless" playlist entities, which substantially degrades the overall performance.

As mentioned earlier, one can use the title of the playlist as external information to the playlist i to deal with such cases, while latent track factors v are kept as-is:

$$\tilde{s}_{i,j} = f(v_j, T_i) \tag{3}$$

where T_i is the title of the playlist i. Ultimately, f is expected to effectively encode T_i into a certain representation allowing inference of the relationship between two given entities i and j. Intuitively, there are a few ways to find f, as described below.

Direct learning of the entity factor. Here, one tries to find a transformer g approximating an u_i that is pre-derived from other algorithms such as MF, with use of external information of i, being T_i in our case:

$$u_i \approx g(T_i). \tag{4}$$

Recommendation criterion optimization. Alternatively, one can directly find f by optimizing a relevant criterion which maximizes the recommendation performance. Related to the music data domain, such approaches employing external information of the entity are discussed mostly for the item domain [17, 18].

In [17], the authors tested these approaches (in other words: finding either g in (4), or f in (3) directly). The authors discuss that the former case in general works decently, while the latter setup may not give a better result. On the other hand, [18] analytically proved the former case should be inferior to the latter case, as it does not optimize the model to give better recommendations.

Considering these previous works and insights, we hypothesized that learning a model from scratch using recommendation criteria is fundamentally better, but might slow down the learning process. On the other hand, letting the model learn an approximation of pre-trained factors will achieve decent performance faster, but might lead to an irrelevant solution, since no actual recommendation criteria are incorporated. Balancing the trade-offs, in order to get reasonable and sensible performance within a relatively short training time, we set up a multi-objective loss function, as will be described in the next section.

3 IMPLEMENTATION

3.1 Model

3.1.1 Weighted Regularized Matrix Factorization. The base model of the proposed approach is the WRMF algorithm [5], which is learned by solving the following optimization problem:

$$\min_{u^*, v^*} \sum_{i,j} c_{i,j}(s_{i,j} - u_i^\top v_j)^2 + \lambda(\sum_i \|u_i\|^2 + \sum_j \|v_j\|^2) \tag{5}$$

where $s_{i,j}$ refers to the binary membership function between playlist $i \in \{1, 2, ..., I\}$ and track $j \in \{1, 2, ..., J\}$, and $u_i \in \mathbb{R}^d$ and $v_j \in \mathbb{R}^d$ are the latent factors for the ith playlist and the jth track, respectively. λ is the coefficient controlling the regularization of the model. $c_{i,j}$ is the confidence that controls the belief regarding reliability of the membership function, given by:

$$c_{i,j} = 1 + \alpha s_{i,j} \tag{6}$$

where α is the coefficient controlling the confidence of the membership assignment. If α is high, it means the algorithm will rely more on the tracks that are already assigned in the playlist, which implies the model ultimately will treat unseen tracks as "negative" tracks. On the contrary, if α is small, the algorithm depends less on the given assignment, causing the model to give more general suggestions.

3.1.2 Multi-Objective Optimization for Recurrent NCF. After the initial base model is trained as above, we train a RNN model that maps the sequence of N-gram tokens into the same space as the pre-trained factors. Among several options to achieve this [17], we choose to use a multi-objective approach to accelerate the learning process while not losing accuracy:

$$\min_{\Theta^*} \lambda^{\mathcal{L}} SGNS(\tilde{u}_i, v_{j^+}, v_{j^-}) + (1 - \lambda^{\mathcal{L}}) MSE(\tilde{u}_i, u_i) + \lambda^{\Theta} \|\Theta\|^2 \tag{7}$$

where λ^{Θ} controls the amount of L_2 regularization of the set of model parameters Θ of the neural network, and $\lambda^{\mathcal{L}}$ is the mixing coefficient between the two main learning objectives. These objectives are the Mean Squared Error (MSE) between the pre-trained playlist factor u and the approximated playlist factor $\tilde{u}$, and Skip-Gram Negative Sampling (SGNS) ensuring that $\tilde{u}$ gives reasonable recommendations.

3.1.3 Title Encoder. Further, $\tilde{u}_i$ in (7) is the ith playlist factor conditioned by the playlist title, which is derived from the Long Short-Term Memory (LSTM) based RNN[4] as follows:

$$\tilde{u}_i = LSTM(E^{T_i}; \Theta) \tag{8}$$

where $E^{T_i} = \langle E_{t_1}, E_{t_2}, ..., E_{t_L} \rangle$ is a sequence of learnable embedding vectors $E_{t_l} \in \mathbb{R}^{d^E}$ representing T_i, and $T_i = \langle t_1, t_2, ..., t_L \rangle$ is the title of the ith playlist, which is represented as a sequence of integer indices t_l corresponding to each unique character N-gram. In this work, we use a standard 1-layer LSTM which has d^h dimensional hidden states h_l that correspond to N-grams at each step l to encode sequential dependency. Note that for simplicity, we set the dimensionality d^h to be equal to d.

The detailed process is illustrated in Figure 2. The green boxes indicate the states relevant to the text encoder, which are used for the processing of playlist titles. They contain the embeddings of the N-gram tokens corresponding to the ith playlist title (lowest row), hidden states of each token input over sequential steps (middle row), and the playlist embedding $\tilde{u}_i$ that summarizes the textual input data (top row). The blue box refers to the latent factor v of

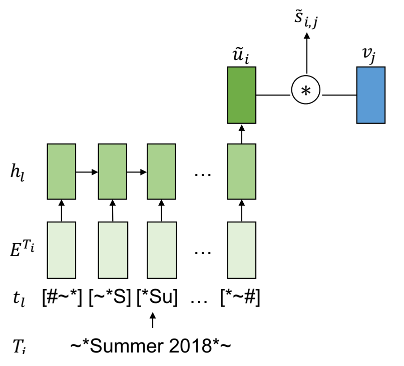

Figure 2: Illustration of the proposed approach.

the jth item, which is multiplied by $\tilde{u}_i$ to get the score $\tilde{s}_{i,j}$. For the rest of the paper, we refer to this model as *RNCF*, as abbreviation for *Recurrent NCF*. In the following subsections, we will focus in more detail on the two objectives.

3.1.4 *Mean Squared Error.* The *MSE* objective is used to approximate the pre-trained playlist factors u as follows:

$$MSE(\tilde{u}_i, u_i) = \frac{1}{m}\sum_i^m \|u_i - \tilde{u}_i\|^2 \tag{9}$$

where m is the number of observations in a mini-batch. As discussed in [17], it is not directly optimizing the model to maximize its recommendation performance. To compensate for this potential problem, as discussed in the previous subsection, we introduced *SGNS* as another main objective.

3.1.5 *Skip-Gram Negative Sampling.* SGNS is a loss function originally developed for the learning of a word embedding model, which is also often used for learning a recommender system [11, 12]. It maximizes the likelihood of the model while minimizing the likelihood of "negative" item suggestions for corresponding users (where in our setup, we consider playlists rather than users). It can be formulated as follows:

$$SGNS(i, j^+, j^-) = -\frac{1}{m}\sum_i^m \Big[logP(s_{i,j^+} = 1|i, j^+, \Theta) + \sum_{j^-} logP(s_{i,j^-} = 0|i, j^-, \Theta) \Big] \tag{10}$$

where $P(s_{i,j^+} = 1|i, j^+, \Theta)$ refers to the likelihood that the playlist i contains track j^+ given model parameter Θ, where j^+ indicates a track that are already was assigned to the playlist i. $P(s_{i,j^-} = 0|i, j^-, \Theta)$, on the other hand, is the likelihood that the playlist i does not have tracks $j^- \in \{j_1^-, ..., j_K^-\}$, which are sampled from the tracks unassigned to the playlist i. Eventually, this objective gives a higher score on the positive tracks and lower score on the negative tracks, given playlist i. This can be re-written as follows:

$$SGNS(\tilde{u}_i, v_{j^+}, v_{j^-}) = -\frac{1}{m}\sum_i^m \Big[log(\sigma(\tilde{u}_i^\intercal v_{j^+})) + \sum_{j^-} log(\sigma(-\tilde{u}_i^\intercal v_{j^-})) \Big] \tag{11}$$

where σ is the sigmoid function.

3.2 Training

As for the training of WRMF, the solution for the model is derived by the Alternating Least Square (ALS) algorithm which is described in the original paper [5]. There are two main advantages of this algorithm: easier parallelism and less hyper-parameters. [5] introduced how the algorithms can easily be distributed, which is preferable for modern computing environments with multi-core CPUs or GPUs. Also, this algorithm updates each iteration's solution by a least square approach, which means that learning hyper-parameters such as the *learning rate* is not necessary. We used the `implicit`[2] library to accelerate experiments. We run 15 iterations for the pre-training.

The RNN is trained by a standard mini-batch stochastic gradient algorithm, using the ADAM [9] optimization technique. Within the training loop, we uniformly sampled K negative samples j^- corresponding to the given triplet of $(i, j^+, s_{i,j})$, which is randomly selected from the dataset.

3.3 Aggregation

After the learning of the RNN, one can combine the approximated playlist factors $\tilde{u}$ and the original u in several way. In this paper, we simply replaced the playlist factors without seed tracks by $\tilde{u}$:

$$\tilde{s}_{i,j} = \begin{cases} u_i^\intercal v_j & \text{if } |j^+| > 0 \\ \tilde{u}_i^\intercal v_j & \text{if } |j^+| = 0 \end{cases} \tag{12}$$

where $|j^+|$ indicates the number of positive samples already assigned to the playlist i. In the testing phase, it refers to the number of seed tracks given to the playlist. After the above aggregation, we refer to the system as Hybrid RNCF (HRNCF), to distinguish the final stage of the system from standard RNCF.

4 EXPERIMENTAL SETUP

Variable	Description	Value
d	dimensionality of latent factors	1,000
d^E	dimensionality of N-gram embedding	300
N	number of characters for the N-gram	3
α	WRMF confidence coefficient	100
λ	WRMF regularization coefficient	0.001
$\lambda^{\mathcal{L}}$	mixing coefficient between MSE and SGNS	0.5
λ^{Θ}	L_2 regularization coefficient for NCF model	0.0001
m	size of the mini-batch	1,024
K	number of the negative samples sampled	4

Table 2: Detailed setup of the hyper-parameters that are used for the proposed approach.

A series of experiments was conducted to determine the optimal hyper-parameter setting for the approach. The detailed final hyper-parameter setup is described in Table 2; Section 5 describes how these parameters were chosen. In addition, we set the learning rate of the NCF model as 0.0005 and the number of maximum iterations r for early stopping as 50,000, to prevent the RNCF model from overfitting. For evaluation, we used the main three metrics proposed by the challenge: Normalized Discounted Cumulative Gain (NDCG),

[2]https://github.com/benfred/implicit

R-precision (RPREC), and the Recommended Songs clicks (CLICKS). The details on the metrics can be found at the challenge overview web site[3]. All the metrics are calculated with a cut-off at 500.

4.1 Baselines

We also compare the proposed approach with several baselines. First, we employed a random recommendation (*Rand*) and the most popular recommender (*MP*) as the most naive baselines to show the lower bound of the performance. Also, we applied the SVD algorithm as a baseline for the matrix factorization method, comparing with the WRMF.

We also introduce a naive text similarity based system. This system uses the same character N-grams as the proposed system, which are then used to build a bag-of-N-grams representing a playlist title. For each query playlist, the title representation is built, and the top-*M* closest playlists are retrieved from the training set employing the cosine similarity. Based on these, the 500 most frequent tracks are used for the recommendation.

5 RESULTS & DISCUSSION

5.1 Overall Results

Model	Category	NDCG	CLICKS	RPREC
Rand	No-Seed	0.0001	50.8413	0.0000
Rand	Only-Seed	0.0002	50.4310	0.0000
Rand	All	0.0002	50.4720	0.0000
MP	No-Seed	0.0242	34.2987	0.0072
MP	Only-Seed	0.0279	26.7686	0.0101
MP	All	0.0275	27.5218	0.0098
SVD_{200}	No-Seed	0.0422	25.9233	0.0115
SVD_{200}	Only-Seed	0.2743	3.9010	0.1365
SVD_{200}	All	0.2511	6.1345	0.1240
$WRMF_{200}$	No-Seed	0.0245	33.552	0.0057
$WRMF_{200}$	Only-Seed	0.3353	1.9492	0.1647
$WRMF_{200}$	All	0.3040	5.1646	0.1488
$NGRAM_{100}$	No-Seed	0.1796	11.3213	0.0744
$NGRAM_{100}$	Only-Seed	0.1808	8.3688	0.0852
$NGRAM_{100}$	All	0.1807	8.6641	0.0841
$WRMF_{1k}$	No-Seed	0.0215	31.1713	0.0049
$WRMF_{1k}$	Only-Seed	0.3331	1.8723	0.1694
$WRMF_{1k}$	All	0.3019	4.7992	0.1529
$RNCF_{1k}$	No-Seed	**0.1866**	**11.2493**	**0.0760**
$RNCF_{1k}$	Only-Seed	0.1901	7.8312	0.0902
$RNCF_{1k}$	All	0.1897	8.1699	0.0888
$HRNCF_{1k}$	No-Seed	0.1866	11.2493	0.0760
$HRNCF_{1k}$	Only-Seed	0.3331	1.8723	0.1694
$HRNCF_{1k}$	All	**0.3185**	**2.8100**	**0.1601**
$HRNCF^{*}_{1k}$	All	**0.3394**	**2.2665**	**0.1924**

Table 3: Comparison between the baseline models and the proposed HRNCF model.

The overall results are described in Table 3. *Rand* refers to the results from random suggestion, and *MP* to the most popular recommendation. SVD_{200} and $WRMF_{200}$ are baseline MF models we tested in the preliminary experiment, with a dimensionality of $d = 200$. $NGRAM_{100}$ is the text based baseline recommendation system introduced in section 4.1.

Further, $WRMF_{1k}$ refers to the baseline WRMF with d =1,000. The $RNCF_{1k}$ model uses WRMF as a pre-trained model. *HRNCF* is the proposed system, which is the solution we submitted to the RecSys 2018 Challenge's Creative Track. The last row of the table is the final result of our actual submission, ranked 10th ultimately as known from the public leaderboards. Finally, *No-Seed* and *Only-Seed* refers to the performance when only taking account the case in which no playlist seeds are provided, and when considering all other cases, respectively.

As described, the proposed approach (HRNCF) achieves best performance in the *All* case. This is expected, as HRNCF combines the better aspects of both WRMF and RNCF. Note that the HRNCF's performance on the *No-Seed* and *Only-Seed* cases are exactly the same as the performance of the RNCF and the WRMF, due to the aggregation strategy described in Section 3.3.

In comparison to the *Rand* or the *MP* based suggestions, WRMF generally shows substantially better performance. However, on the *No-Seed* case, WRMF performance only marginally outperforms *MP*. This is anticipated, since only random factors are used as playlist factors. As a consequence, suggestions are highly dependent of the size of the track factors, which largely follow occurrence frequency in the dataset.

Regarding the $NGRAM_{100}$, we choose the number of nearest playlist M as 100, which is shown as the best within our search range $M \in \{50, 100, 250, 500\}$. Notably, while it is indicated that the RNCF approach is better than the $NGRAM_{100}$, the gap between the two is not substantial despite the simplicity of the $NGRAM_{100}$. It implies that the sequential dependency between the N-grams can be either trivial information for given task or sub-optimally learned from the training process of the RNN.

5.2 WRMF

To select the best hyper-parameters, we conducted a grid search on the important parameters. As for the dimensionality of the latent factor d, we verified 6 different setups $\{20, 50, 100, 200, 500, 1000\}$. As illustrated in Figure 3, in our current experiments, major performance gains are found until $d = 500$. However, due to computational limitations, it was infeasible for us to investigate $d > 1000$. Further research will be needed to verify performance for larger choices of d. It turned out that α also affects the performance substantially, as shown in Figure 4. We tested $\alpha \in \{10^0, 10^1, 10^2, 10^3\}$, where $\alpha = 10^2$ turned out the best choice.

5.3 Multi-Objective RNCF

We also investigated the effect of mixing coefficient $\lambda^{\mathcal{L}}$ between the losses. For the search range, we tried $\lambda^{\mathcal{L}} \in \{0, 0.25, 0.5, 0.75, 1\}$. As shown in Figure 5, $\lambda^{\mathcal{L}} = 0.75$ or $\lambda^{\mathcal{L}} = 0.5$ are tentatively better than the other options. We decided to use 0.5, as it yields better performance on the *No-Seed* case, which is our main reason for employing the RNCF model. However, considering that the curve shows sharp changes between the tried values, more research is needed to find a truly optimal $\lambda^{\mathcal{L}}$.

[3]https://recsys-challenge.spotify.com/rules

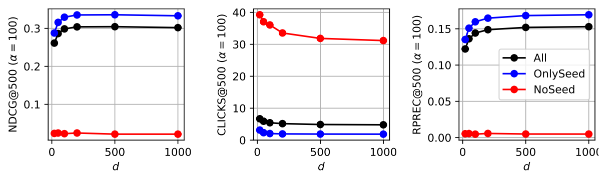

Figure 3: Performance of WRMF with respect to d.

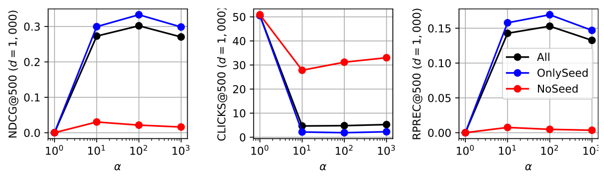

Figure 4: Performance of WRMF with respect to α.

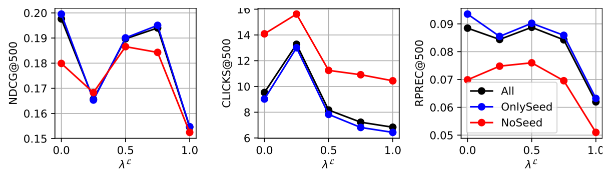

Figure 5: Performance of RNCF with respect to $\lambda^{\mathcal{L}}$.

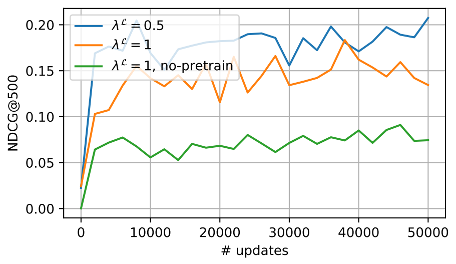

Figure 6: Effect of the pre-trained factors in RNCF learning.

We also tested a RNCF model that does not employ pre-trained MF factors. In this case, one also needs to find track factors v that minimize the SGNS loss. As [18] pointed out, this approach is fundamentally better in terms of minimizing recommendation error. However, as shown in Figure 6, within our current empirical investigations, learning progress for this approach appears much slower than in the other approaches we tested.

6 FUTURE WORK

In this work, we introduced a hybrid approach that employs playlist title information for "seedless" music playlist generation. Our empirical investigations indicate that employing a hybrid RNCF model can indeed help in solving the problem.

In addition to the proposed method, one can also examine content-based approaches. Although CF-based recommender systems are powerful and generally surpass the content-based approaches [14], they can miss items in the long tail due to the scarcity of usage data. To this end, previous research attempted to learn latent factors of CF from audio content [17]; other previous work demonstrated the versatility of pre-trained convnet features which were transferred from an automatic music tagging network [3].

As the given data for the challenge (playlist title and seed tracks), can be directly related to audio content, one can exploit their correlation for the music playlist generation. Though we did not deal with content-based approaches in this paper, an audio crawling method will be shared on our repository[4].

ACKNOWLEDGMENTS

This work was carried out on the Dutch national e-infrastructure with the support of SURF Cooperative, and partially funded by the predoctoral grant MDM-2015-0502-17-2 from the Spanish Ministry of Economy and Competitiveness linked to the Maria de Maeztu Units of Excellence Programme (MDM-2015-0502).

REFERENCES

[1] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606* (2016).

[2] 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)*. ACM, New York, NY, USA.

[3] Keunwoo Choi, György Fazekas, Mark B. Sandler, and Kyunghyun Cho. 2017. Transfer Learning for Music Classification and Regression Tasks. In *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017, Suzhou, China, October 23-27, 2017*. 141–149.

[4] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9, 8 (Nov. 1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[5] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 8th IEEE International Conference on Data Mining*. 263–272.

[6] Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991* (2015).

[7] Christopher C Johnson. 2014. Logistic matrix factorization for implicit feedback data. In *Advances in Neural Information Processing Systems*. Vol. 27.

[8] Ioannis Kanaris, Konstantinos Kanaris, Ioannis Houvardas, and Efstathios Stamatatos. 2007. Words versus character n-grams for anti-spam filtering. *International Journal on Artificial Intelligence Tools* 16, 06 (2007), 1047–1067.

[9] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[10] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 8 (2009), 30–37.

[11] Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*. 2177–2185.

[12] 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*. 3111–3119.

[13] Yue Shi, Martha Larson, and Alan Hanjalic. 2014. Collaborative Filtering Beyond the User-Item Matrix: A Survey of the State of the Art and Future Challenges. *ACM Comput. Surv.* 47, 1, Article 3 (May 2014), 45 pages. https://doi.org/10.1145/2556270

[14] Malcolm Slaney. 2011. Web-scale multimedia analysis: Does content matter? *IEEE MultiMedia* 18, 2 (2011), 12–15.

[15] Xiaoyuan Su and Taghi M Khoshgoftaar. 2009. A survey of collaborative filtering techniques. *Advances in artificial intelligence* 2009 (2009).

[16] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. LSTM Neural Networks for Language Modeling. In *INTERSPEECH*.

[17] Aäron van den Oord, Sander Dieleman, and Benjamin Schrauwen. 2013. Deep content-based music recommendation. In *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 2643–2651.

[18] Xinxi Wang and Ye Wang. 2014. Improving Content-based and Hybrid Music Recommendation using Deep Learning. In *Proceedings of the ACM International Conference on Multimedia, MM '14, Orlando, FL, USA, November 03 - 07, 2014*. 627–636. https://doi.org/10.1145/2647868.2654940

[4]https://github.com/eldrin/recsys18-spotify-spotif-ai.git