TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Wirtschaftswissenschaften Professur für BWL, insbes. Industrielles Management, Prof. Dr. Udo Buscher

Non-Deep Learning Collaborative Filtering Techniques for Implicit Feedback Data in Music Recommender Systems

Markus Höfling

Born on: 27th March 1993 in Dresden

Course: Industrial Engineering 8. FS

Matriculation number: 4782795

Exposé within the research seminar Industrial Management

Supervisor

Dipl.-WIng. Daniel Zähringer

Supervising professor

Prof. Dr. Udo Buscher

Submitted on: 7th July 2024

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ACCF	Asymmetric Cosine-based Collaborative Filtering
ACM	Association of Computing Machinery
ACOS	Asymmetric Cosine
ALS	Alternating Least Squares
APLT	Average Percentage of long-tail Items
ARP	Average Recommended Popularity
CF	Collaborative Filtering
COS	Cosine
DJ	Discjockey
DJC-DB	DJcity Dataset
EUC	Euclidean Distance
IC	Item Coverage
iDCG	Ideal Discounted Cumulative Gain
k-NN	k-Nearest-Neighbors
LSA	Latent Semantic Analysis
MAE	Mean Absolute Error
MAP	Mean Average Precision
MF	Matrix Factorization
MRR	Mean Reciprocal Ranking
MRS	Music Recommender System
MSD	Million Song Dataset
nDCG	Normalized Discounted Cumulative Gain
PCC	Pearson Correlation Coefficient
Prec	Precision
RMSE	Root Mean Squared Error
RS	Recommender System
SGD	Stochastic Gradient Descent
SVD	Singular Value Decomposition

List of Symbols

α	Power Factor for Asymmetric Cosine Distance
β	Power Factor for Asymmetric User/Item-Based Utility Prediction
$c(u, i)$	The Confidence of User u Towards Item i
$e(\cdot, \cdot)$	Error Between Actual Value $r(u, i)$ and Predicted Value $q_i^\intercal p_u$
$f(u, i)$	Estimator Function for $\tilde{r}(u, i)$
Γ	Set of Long-Tail Items
H	Set of Hit Items in the Recommended List Q
I	Set of Items
$I(u)$	Set of Items Interacted with by User u
i, j	Variables Representing Random Items
$I_k(u)$	Set of k Items Interacted with by User u
λ	Regularization Term of the Matrix Factorization Objective Function
$N_i(u)$	All Neighbors of Target User u Who Have Interacted with Item i
$N_k(u)$	k-Nearest Neighbors with the Highest Similarity Score to Target User u
$p(u, i)$	The Preference of User u Towards Item i
p_u	Latent Factor Vector of User u
q	Locality Factor for Similarity Weights
$\mathcal{Q}$	Set of Queries Q
q_i	Latent Factor Vector of Item i
Q	Set of Recommended Items
R	Set of Known Utilities
$r(u, i)$	Known Utilities
$\hat{R}$	Subset of Recorded Utilities
$\tilde{R}$	Set of Unknown Utilities
$\tilde{r}(u, i)$	Unknown Utilities
S	Set of Possible Utility Values
Σ	Diagonal Matrix with Singular Values as Diagonal Elements
σ	Singular Values

U	Set of Users
$U(i)$	Set of Users Interacted with Item i
u, v	Variables Representing Random Users
$U_k(i)$	Set of k Users Interacted with Item i
X	Set of Known Interactions
$\tilde{X}$	Set of Unknown Interactions

1 Introduction

The increasing amount of data has led Recommender Systems (RSs) to become integral to many online platforms, enhancing user experience by providing personalized recommendations based on user preferences and behaviors[1]. Beyond benefiting consumers, RSs also create significant value for providers and other stakeholders[2]. For example, Netflix revealed that "75% of what people watch is from some sort of recommendation", leading to increased user interaction and decreased churn. Similarly, YouTube reported that 60% of clicks on their home screen are on recommended content[3]. These success stories highlight the potential of well-tailored recommendations across various domains[4]. Currently, most RSs operate within single domains, such as Netflix for movies and series or Spotify for music and podcasts[5]. In the realm of music streaming services like Apple Music, Deezer, Tidal, and Spotify, Music Recommender Systems (MRSs) are essential for helping users navigate vast catalogs of tracks, addressing content overload by tailoring recommendations to individual tastes[6]. This work explores non-deep learning Collaborative Filtering (CF) techniques for implicit feedback data in MRS. It examines user-based and item-based neighborhood techniques, as well as Matrix Factorization (MF) with Alternating Least Squares (ALS) optimization, applied to a real-life dataset with binary feedback provided by world's leading subscription-based digital record pool for Discjockeys (DJs) - DJcity.[7] These classical methods serve as a baseline for further research, facilitating the development and comparison of more sophisticated algorithms. By establishing a robust baseline using non-deep learning techniques, this work also aims to contribute to continuing research efforts.

[1] Vgl. Li et al. (2024): *Recent Developments in RSs*, p. 1.
[2] Vgl. Jannach / Jugovac (2019): *Business Value of RSs*, 16:1.
[3] Vgl. Davidson et al. (2010): *The YouTube Video Recommendation System*, p. 296.
[4] Vgl. Jannach / Jugovac (2019): *Business Value of RSs*, 16:3.
[5] Vgl. Dacrema et al. (2012): *Design and Evaluation of Cross-Domain Recommender Systems*, p. 485.
[6] Vgl. Perera et al. (2020): *Analysis of MRSs*, p. 1.
[7] DJcity.com

2 Recommender Systems

2.1 Data Sources

A RS is a software tool that provides personalized suggestions to users based on their preferences, behaviors, or the preferences of similar users[1]. The data used within RSs can be related to three different kinds, namely *users*, *items*, or *interactions* between the former two. *Users* in a RS hold a range of information which can be put together into a user model. This user model represents the preferences towards different items in the system and can be either a simple list of history user-item interactions or hold more explicit information such as age, sex, ...[2]. *Items* are the target objects to recommend to users in a RS. Depending on their availability, these can hold certain item features such as simple meta-data, images, raw text descriptions which therefore require more algorithms (e.g. natural language processing) to analyze[3]. *Interactions* build the connection between the users and items in a RS. These connections can either be expressed *explicit* through ratings by the user or tracked *implicitly* during the interaction through e.g. clicking, buying an item by the user. Whereas explicit feedback is preferred against implicit information and is mainly focused in the research[4], implicit feedback reflects a more realistic scenario. This indirectly reflects the user's preferences and can be mainly characterized by the following properties[5]:

1. *No negative feedback:* While explicit feedback provides both positive and negative preferences (e.g., a 5-star rating system where 1 signifies "fully like" and 5 "totally dislike"), implicit feedback lacks information about what users dislike. This fundamental asymmetry complicates the analysis of implicit feedback, as focusing solely on gathered information can lead to a skewed understanding of user preferences.

2. *Noise:* Relying solely on user interactions can leave room for misinterpretation. For instance, a playlist that is played while the listener is asleep or unable to pause

[1] Vgl. Ricci / Rokach / Shapira (2021): *RSs: Techniques, Applications, and Challenges*, p. 1.
[2] Vgl. ibid., p. 9.
[3] Vgl. ibid., p. 9.
[4] Vgl. Hu / Koren / Volinsky (2008): *Collaborative Filtering for Implicit Feedback*, p. 2.
[5] Vgl. ibid., p. 2.

or skip content, or when another account is used to interact with content during a social event, introduces noise into the implicit feedback data.

3. *Preference vs. confidence:* Explicit feedback data provides a numerical representation of user preferences, while implicit feedback indicates user confidence. Different levels of confidence in implicit signals can influence the recommendation process, highlighting items preferred by the user based on their interaction patterns (this will be apprehended in 3.4.3 more precisely).

To ensure optimal performance of a RS, accurate data is crucial[6]. The term "rating" in the context of implicit feedback can lead to confusion; hence, the term "*utility*" has already been introduced by Ricci / Rokach / Shapira (Vgl. 2021, p. 11) to denote the perceived value or relevance of an item to a user.

2.2 Music Recommender Systems

In recent years, the proliferation of music streaming services like Apple Music, Deezer, Tidal, and Spotify has vastly increased the availability of music anytime and anywhere. However, the expanding catalog sizes of these services have made it challenging for consumers to effectively navigate and discover content that aligns with their preferences[7]. Modern MRSs have emerged as an application of RS to the music domain to address this issue by aiding users in filtering the most relevant items[8]. Several characteristics distinguish the music domain from others[9]:

- *Duration:* Unlike movies or books, the consumption time for a single song is typically just a few minutes, leading to a higher volume of interactions overall.
- *Catalog size:* Music streaming platforms boast catalogues containing millions of songs, significantly larger than those of other domains. Hence, scalability of algorithms becomes a critical consideration.
- *Repeated consumption:* Songs are often replayed by users, necessitating frequent re-recommendations.
- *Sequential consumption:* Unlike books or movies that are consumed individually, music is often listened to in sequences, creating interdependencies between songs.

[6]Vgl. Zangerle / Bauer (2022): *Evaluating RSs: Survey and Framework*, 170:18.
[7]Vgl. Schedl et al. (2021): *MRSs: Techniques, Use Cases, and Challenges*, p. 928.
[8]Vgl. Perera et al. (2020): *Analysis of MRSs*, p. 1.
[9]Vgl. ibid., p. 1.

These conditions also effect the general recommendation task a RS has to cover. The most common task is to generate a *ranked list of items*[10]. However, MRSs encounter different scenarios, necessitating specialized tasks for RSs to perform[11]:

- *Dynamic Playlist Generation:* Personalized playlists are generated based on a user's interaction history.
- *Next Track Recommendation:* Providing users with a list of items using their historical interactions.
- *Automatic Playlist Continuation:* Users input a set of items, and the RS suggests additional items that best complement the selection. In other domains, this is commonly referred to as session-based recommendation[12].

However, the traditional model of download-based services, exemplified by platforms like Apple iTunes, Amazon, Beatport, Last.fm, and others, continues to persist in the market. Several reasons contribute to their ongoing relevance and likely sustainability in the near future. Downloaded songs remain accessible regardless of internet connectivity, ensuring high-quality, uninterrupted listening experiences unaffected by bandwidth constraints. This feature is particularly valued in domains such as concerts where uninterrupted quality is paramount[13]. Beyond that, ongoing negotiations between streaming services and music labels often result in content removals. For instance, Universal Music Group (UMG) withdrew their songs from TikTok Music effective February 1, 2024[14][15].

[10] Vgl. Ricci / Rokach / Shapira (2021): *RSs: Techniques, Applications, and Challenges*, p. 2.
[11] Vgl. Schedl et al. (2021): *MRSs: Techniques, Use Cases, and Challenges*, pp. 931–940; Perera et al. (2020): *Analysis of MRSs*, p. 5.
[12] Vgl. Jannach / Quadrana / Cremonesi (2022): *Session-Based RSs.*
[13] Vgl. Greentree (2023): *Subjective Sounds: Why you should Download Music.*
[14] Vgl. TikTok (2024): *TikTok Music.*
[15] Vgl. Group (2024): *UMG: Time Out on TikTok.*

3 Collaborative Filtering

3.1 Formalization of the Recommendation Problem

Collaborative filtering belongs to the more traditional classes of RSs. There are many more techniques how to achieve the recommendation task depending on the data available[1]. Whereby CF relies solely on user-item interaction data, *content-based filtering* matches user and item attributes to make recommendations[2] and *hybrid approaches* forms a combination out of the former both[3] just to name a few.

Before delving into various techniques for CF, the recommendation task is formally divided into two main problems and their respective models. Let U with $|U| = k$ represent the set of users $u \in U$, and I with $|I| = n$ represent the set of items $i \in I$. The set of recorded user-item interactions is stored in $X = \{(u, i) \mid r(u, i) \text{ is known}\}$ and the corresponding utilities denoted by $R = \{r(u, i) \mid (u, i) \in X\}$. Let S be the set for the possible values of $r(u, i)$, e.g., $S = [1, 5]$ for an explicit star rating from 1 to 5. Notably, not every user has interacted with every item, so there exists a set $\tilde{X} = \{(u, i) \mid r(u, i) \text{ is unknown}\}$, and in general $|R| \ll k \cdot n$ holds[4]. The unknown utilities are stored in $\tilde{R} = \{\tilde{r}(u, i) \mid (u, i) \in \tilde{X}\}$. Assuming these unknown values as $\tilde{r}(u, i) =?$ and the known values $\tilde{r}(u, i) = 1$, the matrix representation of $R \cup \tilde{R} \subset \mathbb{R}^{k \times n}$ with $k = 3$ and $n = 5$ would appear as in the following (3.1):

$$R \cup \tilde{R} = \begin{bmatrix} 1 & ? & ? & ? & 1 \\ ? & 1 & ? & ? & ? \\ ? & ? & ? & 1 & ? \end{bmatrix} \tag{3.1}$$

Here, the missing values lead to the first formulation of the recommendation problem:

[1] Vgl. Schedl et al. (2021): *MRSs: Techniques, Use Cases, and Challenges*, p. 941.
[2] Vgl. Ricci / Rokach / Shapira (2021): *RSs: Techniques, Applications, and Challenges*, pp. 12–13.
[3] Vgl. Schedl et al. (2021): *MRSs: Techniques, Use Cases, and Challenges*, p. 944.
[4] Vgl. Nikolakopoulos et al. (2021): *A Comprehensive Survey of Neighborhood-Based Methods for RSs*, p. 44.

1. Utility Prediction

The goal here is to find a proper real-valued function $f : U \times I \longrightarrow S$ which maps all existing $u, i \in U, I$ to a certain utility $f(u, i) = \tilde{r}(u, i)$ that can fill the set $\tilde{R}$ and thus fill the entire union in (3.2).

$$R \cup \tilde{R} = \begin{bmatrix} 1 & \tilde{r}(1,2) & \tilde{r}(1,3) & \tilde{r}(1,4) & 1 \\ \tilde{r}(4,2) & 1 & \tilde{r}(4,3) & \tilde{r}(4,4) & \tilde{r}(4,5) \\ \tilde{r}(5,2) & \tilde{r}(5,2) & \tilde{r}(5,3) & 1 & \tilde{r}(5,5) \end{bmatrix} \tag{3.2}$$

2. Top-N Ranking

In the literature, the second formulation of the problem does not involve calculating utilities first but rather focuses solely on the recommendation task: providing a fixed user u with a list $Q = \{i_1, \cdots, i_N\}$ of size $|Q| = N$ recommended items that the user is most interested in. The drawback of this model is evident, as each item is assumed to be equally of interest to the user by default[5]. The first problem formulation provides a more general description. Once the utilities of user-item interactions are calculated, for a fixed user u, predictions can be computed for all $i \in I$ as $f(u, i_1), \ldots, f(u, i_n)$, and items $i_1, \ldots, i_k$ $(k \leq n)$ with the highest predicted values $f(u, i)$ can be recommended[6]. Various techniques have emerged to achieve these values.

3.2 Methods of Collaborative Filtering

The term CF was first mentioned in Goldberg et al. (1992, „Tapesty“), marking a significant milestone in the history of RSs[7]. The key idea behind CF is to utilize the interaction data of users with items, leveraging the insight that a target user's rating for unseen items tends to align with ratings given by other users with similar tastes. This assumption stems from the belief that users with comparable interaction histories exhibit similar preferences and are likely to appreciate similar items, making it valuable to recommend favored items from one user to another with a comparable interaction history. CF thus addresses some of the limitations of content-based filtering by not requiring explicit information about items or users. CF can be further categorized into *neighborhood* (memory-based[8]) and *model-based* approaches. Neighborhood-based methods directly

[5] Vgl. Nikolakopoulos et al. (2021): *A Comprehensive Survey of Neighborhood-Based Methods for RSs*, p. 45.
[6] Vgl. Ricci / Rokach / Shapira (2021): *RSs: Techniques, Applications, and Challenges*, p. 12.
[7] Vgl. Hu / Koren / Volinsky (2008): *Collaborative Filtering for Implicit Feedback*, p. 1.
[8] Vgl. Li et al. (2024): *Recent Developments in RSs*, p. 2.

utilize interaction data to make predictions, which can further be distinguished as *item-* or *user*-based[9]. The work by Aiolli (2013, Vgl.) addresses similar dataset structures and focuses extensively on neighborhood-based models, influencing subsequent algorithmic developments.

3.3 Neighborhood-Based Collaborative Filtering

3.3.1 User-Based Recommendation

Neighborhood-based methods can be primarily categorized into two core concepts: *user-based* and *item-based* recommendation[10]. User-based neighborhood recommendation is a CF method that predicts the rating a user might give to an unseen item based on the ratings given by other users who share similar preferences. The underlying assumption is that users who have interacted with similar items tend to have similar tastes and preferences, thus consuming similar items[11].

	Item 1	Item 2	Item 3	Item 4	Item 5
User 1	1	1	?	1	1
User 2	?	1	1	1	1
User 3	1	?	1	1	1
User 4	1	1	?	1	?
User 5	1	?	1	?	1

Table 3.1: Example Data Neighborhood Based Method

With $U = \{\text{User 1}, \ldots, \text{User 5}\}$ and $I = \{\text{Item 1}, \ldots, \text{Item 5}\}$, the user-item interaction matrix R takes the form shown in 3.1. Assume the task is to predict the yet unknown value $\tilde{r}(1, 3)$ by $f(1, 3)$. Then, following an illustrative example for this approach within the three basic steps[12]:

1. Similarity Calculation

Identify users who have similar tastes to the target user u by calculating the similarity $\text{sim}(u, v)$ between u and every other user $v \in U \setminus \{u\}$ based on their rating profiles. Common similarity measures include Pearson Correlation Coefficient (PCC), Cosine (COS)

[9] Vgl. Nikolakopoulos et al. (2021): *A Comprehensive Survey of Neighborhood-Based Methods for RSs*, pp. 45–48.
[10] Vgl. ibid., p. 40.
[11] Vgl. Almazro et al. (2010): *A Survey Paper on Recommender Systems*, p. 3.
[12] Vgl. Herlocker et al. (1999): *An Algorithmic Framework for Performing Collaborative Filtering*, p. 231.

Distance, Euclidean Distance (EUC), and others[13]. For this example, the COS similarity measure (3.3) will be applied to the set of users in Table 3.1. In the case of only positive utilities in the matrix R, the COS similarity will yield values in the interval $[0, 1]$, since there are no opposing vectors in the positive space[14]. All non-observed entries will be set to 0.

$$sim(u, v) := \text{COS similarity}(u, v) = \frac{u^\intercal \cdot v}{||u|| \cdot ||v||} \tag{3.3}$$

In the special case of a binary dataset where each user is represented as a vector in the example shown in Table 3.1 as $u \in \{0, 1\}^5$, the cosine similarity simplifies as suggested in Aiolli (Vgl. 2013, p. 3). The numerator is the sum of products of $0's$ and $1's$, which equals 1 if and only if both vectors contain 1 at the same position i. This condition holds true only when both users u and v have interacted with the same item i. $I(u)$ is defined as the set of items, user u has interacted with. Then, the numerator can be expressed as $|I(u) \cap I(v)|$, indicating the number of common items interacted by both users u and v. A similar simplification applies to the denominator, as the norm is defined as $||u|| = \sqrt{\sum_{i=1}^{k} u_i^2} = \sqrt{\sum_{i=1}^{k} u_i} = \sqrt{|I(u)|}$. Therefore, the cosine similarity measure simplifies to (3.4).

$$\frac{u^\intercal \cdot v}{||u|| \cdot ||v||} = \frac{|I(u) \cap I(v)|}{\sqrt{|I(u)|} \cdot \sqrt{|I(v)|}} \tag{3.4}$$

Continuing with the example where $u :=$ User 1 $= (1, 1, 0, 1, 1)$ and $v :=$ User 2 $= (0, 1, 1, 1, 1)$ it follows the expressions $|I(u) \cap I(v)| = |\{\text{Item 2, Item 4, Item 5}\}| = 3$, $\sqrt{|I(u)|} = \sqrt{4} = 2$, $\sqrt{|I(v)|} = \sqrt{4} = 2$ and that's $sim(u, v) = \frac{3}{4} = 0.75$. The computation for other similarities follows similarly.

	User 1	User 2	User 3	User 4	User 5
User 1	1	0.75	0.75	0.866	0.577

Furthermore, Aiolli (Vgl. ibid., p. 4) suggests a generalized version of the COS similarity, known as the Asymmetric Cosine (ACOS) Similarity, which introduces a parameter $\alpha \in [0, 1]$. The ACOS is defined as in (3.5) as follows:

$$\text{ACOS similarity}(u, v) = \frac{|I(u) \cap I(v)|}{|I(u)|^\alpha \cdot |I(v)|^{1-\alpha}} \tag{3.5}$$

With this modification, the author aims to adapt standard measures to specialized do-

[13] Vgl. Li et al. (2024): *Recent Developments in RSs*, p. 2.
[14] Vgl. Nikolakopoulos et al. (2021): *A Comprehensive Survey of Neighborhood-Based Methods for RSs*, p. 55.

mains and tasks that have demonstrated significant success in the Million Song Dataset (MSD) Challenge[15], where they still hold the top position on the leaderboard. It's noteworthy that for $\alpha = 0.5$, the ACOS reduces to the standard COS and in that case $sim(u, i) = \text{ACOS similarity}(u, v)$ holds. Additionally, the authors suggest incorporating a locality factor by raising the similarity weights to the power of $q \in \mathbb{N}$[16].

2. Neighborhood Formation

Select the *k-Nearest-Neighbors (k-NN)*, denoted as $N_k(u)$, which consists of the k users with the highest similarity score $\text{sim}(u, v)$ to the target user[17]. This approach, also utilized by Aiolli (Vgl. 2013, p. 8), has demonstrated strong performance in experiments. However, a drawback is the need to compute similarities for all users to identify the most similar ones. An alternative approach, commonly used and proposed by Nikolakopoulos et al. (Vgl. 2021, p. 46), involves selecting users $N_i(u)$ who have interacted with the specific item i.

3. Utility Prediction

Consider the prediction formula (3.6) by Nikolakopoulos et al. (Vgl. ibid., p. 46):

$$f(u, i) = \frac{\sum_{v \in N_i(u)} \text{sim}(u, v) \cdot r_{vi}}{\sum_{v \in N_i(u)} \text{sim}(u, v)} \tag{3.6}$$

For the binary rating case, the basic framework proposed in Nikolakopoulos et al. (Vgl. ibid., p. 55) cannot be directly applied, as illustrated by the following example. The issue arises because $r(u, i) \in \{0, 1\}$, and the summation over all $v \in N_i(u)$, where $r(u, i) = 1$, it leads to the equation

$$f(u, i) = \frac{\sum_{v \in N_i(u)} \text{sim}(u, v)}{\sum_{v \in N_i(u)} \text{sim}(u, v)} = 1$$

This issue has already been addressed in Aiolli (Vgl. 2013, pp. 4 & 8), where two versions have been applied:

$$f(u, i) = \frac{\sum_{v \in N_k(u)} \text{sim}(u, v) \cdot r_{vi}}{\sum_{v \in N_k(u)} \text{sim}(u, v)} \tag{3.7}$$

[15] Vgl. McFee et al. (2012): *The Million Song Dataset Challenge.*
[16] Vgl. Aiolli (2013): *Efficient Top-N Recommendation for Binary Datasets*, p. 4.
[17] Vgl. Herlocker et al. (1999): *An Algorithmic Framework for Performing Collaborative Filtering*, p. 234.

$$f(u,i) = \frac{\sum_{v \in U(i)} \text{sim}(u,v)}{||sim(u,\cdot)||^{2\beta} \cdot |U(i)|^{(1-\beta)}} \tag{3.8}$$

The slight difference in the first equation is that the neighborhood $N_k(u)$ is now independent of the target item i, considering only the set of k nearest users $v \in N_k(u)$ based on their similarities to the target user u. For instance, with $k = 2$, this leads to $N_2(u) = \{\textit{User 3}, \textit{User 4}\}$ and

$$f(1,3) = \frac{0.75 \cdot 1 + 0.866 \cdot 0}{0.75 + 0.866} = \frac{0.75}{1.616} = 0.464.$$

The second equation's neighborhood results in $U(i) = \{\textit{User 2}, \textit{User 3}, \textit{User 5}\}$, which is called as Asymmetric Cosine-based Collaborative Filtering (ACCF). Applying (3.8) with $\alpha = \beta = 0.5$ to example 3.1:

$$f(1,3) = \frac{0.75 + 0.75 + 0.577}{\sqrt{0^2 + 0.75^2 + 0.75^2 + 0.866^2 + 0.577^2}^{2 \cdot 0.5} \cdot 3^{(1-0.5)}} = \frac{2.077}{1.486 \cdot \sqrt{3}} = 0.807$$

3.3.2 Item-Based Recommendation

Instead of assessing similarities between a target user and its neighbors, item-based neighborhood recommendation seeks similar items to a specific target item i. The underlying assumption is that users typically interact with items that are similar to those they have engaged with in the past[18]. Building on the groundwork laid in the previous subsection, analogous steps lead already to the formulas for the item-based case. Only substituting $U(i)$ with $I(u)$ and focusing on calculating similarities between each pair of items i, j in (3.9), rather than users u, v.

$$sim(i,j) = \text{ACOS similarity}(i,j) = \frac{|U(i) \cap U(j)|}{|U(i)|^{\alpha} \cdot |U(j)|^{1-\alpha}} \tag{3.9}$$

$$f(u,i) = \frac{\sum_{j \in N_k(i)} \text{sim}(i,j) \cdot r_{uj}}{\sum_{j \in N_k(i)} \text{sim}(i,j)} \tag{3.10}$$

$$f(u,i) = \frac{\sum_{j \in I(u)} \text{sim}(i,j)}{|I(u)|^{\beta} \cdot ||\text{sim}(i,\cdot)||^{2(1-\beta)}} \tag{3.11}$$

[18] Vgl. Almazro et al. (2010): *A Survey Paper on Recommender Systems*, p. 4.

3.3.3 User-Based vs. Item-Based Recommendation

When choosing between user-based and item-based RS, several criteria should guide the decision. Firstly, accuracy plays a significant role. User-based methods tend to excel when the number of items exceeds the number of users in the system. For instance, in MRS with a vast catalog size relative to the user count, user-based methods often perform better. Conversely, item-based methods demonstrate greater accuracy in systems where the user base surpasses the number of items available, as seen in large-scale platforms like Amazon[19].

3.4 Model-Based Methods

3.4.1 Drawbacks of Neighborhood-Based Methods

Nearest neighbor algorithms in RS encounter challenges related to sparsity and scalability. Sparsity arises when many users rate too few products, resulting in limited recommendation coverage and reduced accuracy. Scalability poses another issue, as computational demands increase with the number of users and products, making it difficult to scale for large systems[20]. While neighborhood-based methods focus on relationships between users or items, model-based approaches aim to map both items and users into an embedding space characterized by latent factors. Hence, these methods are often referred to as "latent factor models"[21]. This category of methods includes techniques such as Latent Semantic Analysis (LSA) using Singular Value Decomposition (SVD), neural networks, and factorization-based models like MF[22] which of one will be further explained in detail.

3.4.2 Matrix Factorization

One of the most successful applications of latent factor models is MF[23]. A notable implementation of MF, successfully applied during the Netflix Prize competition that began in 2006, utilized SVD[24]. This competition garnered significant interest due to its dataset size, which was orders of magnitude larger than typical datasets used in CF research. Recent

[19] Vgl. Nikolakopoulos et al. (2021): *A Comprehensive Survey of Neighborhood-Based Methods for RSs*, pp. 49–51.
[20] Vgl. Sarwar et al. (2000): *Application of Dimensionality Reduction in RS*, pp. 3–4; Nikolakopoulos et al. (2021): *A Comprehensive Survey of Neighborhood-Based Methods for RSs*, p. 63.
[21] Vgl. Koren / Rendle / Bell (2021): *Advances in Collaborative Filtering*, p. 92.
[22] Vgl. ibid., p. 102.
[23] Vgl. Koren / Bell / Volinsky (2009): *Matrix Factorization Technique*, p. 43.
[24] Vgl. Bennett / Lanning, et al. (2007): *The Netflix Prize*, p. 4.

studies indicate that MF remains relevant despite advancements such as neural collaborative filtering, as proposed by He et al. (2017). Rendle et al. (2020) demonstrated in their 2020 paper "Neural Collaborative Filtering vs. Matrix Factorization Revisited" that the simple dot product model of MF often outperforms more complex neural network approaches, particularly when hyper-parameters are appropriately tuned.

As previously noted, MF belongs to the family of latent factor models, which aim to uncover latent relationships by projecting both users and items into a lower-dimensional latent factor space[25]. In this section, the focus will be on MF models induced by SVD. SVD is the decomposition of a matrix, such as the user-item matrix $R \in \mathbb{R}^{m \times n}$ (with m users and n items), into three components typically denoted as U, Σ, and V, where $R = U\Sigma V^\intercal$. Here, $U \in \mathbb{R}^{m \times m}$, $V \in \mathbb{R}^{n \times n}$, and $\Sigma \in \mathbb{R}^{m \times n}$ is a diagonal matrix $\Sigma = \text{diag}(\sigma_1, \sigma_2, \cdots, \sigma_n)$. The dimensions imply that U corresponds to user factors and V to item factors. The values σ_i are known as the singular values of the matrix R.

Consider users u and v, items i and j, and a set R containing all observed user-item interactions where $r(u, i) = 1$ for the implicit binary case. The objective remains to predict the missing ratings $\tilde{r}(u, i) \in \tilde{R}$ by $f(u, i)$. Assume the latent factor space has dimension l, and each user and item is associated with a latent representation $p_u \in \mathbb{R}^l$ and $q_i \in \mathbb{R}^l$ respectively. Each value within p_u and q_i represents the extent of a certain latent factor. The utility of a user u towards an item i is then approximated by the dot product of these vectors: $\tilde{r}(u, i) = q_i^\intercal p_u$. The key challenge is to compute all latent representations $p_u, q_i \in \mathbb{R}^l$ for all u, i. This is typically achieved using optimization techniques such as Stochastic Gradient Descent (SGD) or Alternating Least Squares (ALS) on a regularized version of the general objective function (3.12)[26].

$$\min_{q_i, p_u} \sum_{(u,i) \in X} (r(u, i) - q_i^\intercal p_u)^2. \tag{3.12}$$

In CF, the MF approach is closely connected to the SVD of the user-item matrix. However, the user-item matrix is typically very sparse, which poses several challenges since traditional SVD methods are not designed to handle missing data. It's important to note that for recommender systems, missing information cannot simply be assumed as zero. Therefore, early adopters of SVD in RSs addressed this issue by filling the missing values $\tilde{r}(u, i)$ with the average of ratings as a preparation step before the actual decomposition task[27]. Moreover, storing the entire matrix is resource-intensive and impractical due to its size. An alternative approach involves only considering the observed ratings in the

[25] Vgl. Koren / Bell / Volinsky (2009): *Matrix Factorization Technique*, p. 44.
[26] Vgl. Zangerle / Bauer (2022): *Evaluating RSs: Survey and Framework*, 170:3.
[27] Vgl. Sarwar et al. (2000): *Application of Dimensionality Reduction in RS*, p. 6.

dataset R. Here, p_u and q_i represent the parameters of the model, which are learned by minimizing the difference between the predicted ratings and the known ratings using a least squares objective[28].

It is also recognized that fitting parameters too closely to the known utilities can hinder the model's ability to generalize and make predictions for unknown ratings[29].

3.4.3 Optimization Techniques

Stochastic Gradient Descent

MF models based on SVD have generally shown a propensity for overfitting. Therefore, considerable research has focused on suitable regularization techniques, such as those proposed by Funk (2006)[30], which penalize the squared norm of the latent vectors and involve selecting an appropriate learning rate λ through cross-validation.

$$\min_{q_i,p_u} \sum_{(u,i)\in X} (r(u,i) - q_i^\intercal p_u)^2 + \lambda(||q_i||^2 + ||p_u||^2). \tag{3.13}$$

Funk (ibid., Vgl.) addressed solving (3.4.3) by employing SGD, where the error $e_{ui} = r(u,i) - q_i^\intercal p_u$ is computed and parameters are updated in each iteration.

- $q_i \leftarrow q_i + \lambda \cdot (e_{ui} \cdot p_u - \lambda q_i)$
- $p_u \leftarrow p_u + \lambda \cdot (e_{ui} \cdot q_i - \lambda p_u)$

Implicit Alternating Least Squares Optimization

While the method initially introduced was applied solely to explicit datasets[31], recent techniques advocate for the use of ALS, which has also been applied to implicit datasets since the early stages of research, borrowing concepts from the aforementioned regularized formula[32]. The algorithm described in Koren (2008) was defined for the general implicit data case, assuming $r(u,i) \in \mathbb{R}_+$. The authors propose defining a binary variable $p(u,i)$ that binarizes $r(u,i)$ as $p(u,i) = 1$ if $r(u,i) > 0$ and $p(u,i) = 0$ if $r(u,i) = 0$:

[28] Vgl. Koren / Rendle / Bell (2021): *Advances in Collaborative Filtering*, p. 94.
[29] Vgl. Koren / Rendle / Bell (2021): *Advances in Collaborative Filtering*, p. 94; Funk (2006): *Netflix Update: Try This At Home.* Funk (2006): *Netflix Update: Try This At Home.*
[30] Vgl. Koren / Rendle / Bell (2021): *Advances in Collaborative Filtering*, p. 94; Paterek (2007): *Improving Regularized SVD for CF*, p. 40.
[31] Vgl. Hu / Koren / Volinsky (2008): *Collaborative Filtering for Implicit Feedback*, p. 2.
[32] Vgl. Hu / Koren / Volinsky (2008): *Collaborative Filtering for Implicit Feedback*; Rendle et al. (2021): *Revisiting the Performance of iALS.*

This variable indicates whether a user has interacted with a specific item, referred to as the *preference*. However, it does not signify whether the user actually liked the item. To address this, a variable for the *confidence* is introduced, denoted as $c(u,i) = 1 + a \cdot r(u,i)$, where $r(u,i)$ is the observed interaction and a is a constant that scales the confidence level. For instance, if a user spends more time listening to a song, including repeated plays, a higher confidence that the user liked it, is inferred. Conversely, if a song is skipped after a few seconds, lower confidence in the user's preference can be concluded[33]. In the binary case, where each user-item interaction is represented by a single interaction (e.g. a download), it can reasonably inferred that a downloaded item is liked by the user. This assumption has also been validated in studies such as Volkovs / Yu (2015), which demonstrated effective results (Vgl. ibid., p. 7). The objective remains to predict the unknown latent factors $p_u \in \mathbb{R}^f$ and $q_i \in \mathbb{R}^f$ for each user u and item i, which are representative of their preferences, $p(u,i) = q_i^\intercal p_u$. Building on these principles and the insights from the formula above, the following optimization function (3.14), denoted as $\mathcal{L}$, is defined as follows:

$$\mathcal{L} := \sum_{u,i} c(u,i)(r(u,i) - p_u^T q_i)^2 + \lambda \sum_u \|p_u\|^2 + \lambda \sum_i \|q_i\|^2 \tag{3.14}$$

For ALS, either p_u or q_i is fixed, transforming the equation into a quadratic function solvable through explicit methods. This alternation in fixing vectors defines the ALS optimization algorithm, where each iteration reduces the overall objective function. The formula is derived by initially fixing the item factors q_i and computing the minima through the derivative of the objective function (3.14); similarly, fixing the user factors p_u follows suit[34]. Introducing Q / P as the matrix of item / user factors, C_u / C_i as the diagonal matrix with (i,i)-th element $c(u,i)$, and $r(u)$ / $r(i)$ as the vector of preference values for user u / item i:

- $q_i \leftarrow q_i = \left(\lambda I + P^T C_i P\right)^{-1} P^T C_i r(i)$
- $q_i \leftarrow q_i = \left(\lambda I + P^T C_i P\right)^{-1} P^T C_i r(i)$[35]

After computing all latent vectors, the predicted preferences for recommendation can be expressed as $\hat{p}_{ui} = p_u^\intercal q_i$[36]. It's important to note that in the binary case, where $c(u,i) = 1$ for all u, i, the predicted rating simplifies to $\hat{r}_{ui} = \hat{p}_{ui}$.

It's worth mentioning that many other regularization terms and ideas have been introduced on known datasets, which is not covered in this work but are still noteworthy. These

[33] Vgl. Hu / Koren / Volinsky (2008): *Collaborative Filtering for Implicit Feedback*, pp. 3–4.
[34] Vgl. ibid., p. 4.
[35] Vgl. ibid., p. 4.
[36] Vgl. ibid., p. 5.

include Bayesian Personalized Ranking (BPR)[37], Probabilistic MF[38], Logistic MF[39], and improvements of methods mentioned earlier[40].

3.5 Common Problems

CF has several notable drawbacks. One significant issue is the *cold-start problem*, where the system struggles to recommend newly introduced items to users or existing items to new users due to its reliance on past user behavior data, such as previous transactions or product ratings. This reliance means that CF requires existing data to make accurate recommendations, and without sufficient historical data, it becomes ineffective. Consequently, while CF can be more accurate than content-based techniques in certain aspects, its dependency on past user behavior limits its ability to address new items in the system[41].

[37] Rendle et al. (2012): *BPR: Bayesian Personalized Ranking from Implicit Feedback.*
[38] Mnih / Salakhutdinov (2007): *Probabilistic Matrix Factorization.*
[39] Johnson et al. (2014): *Logistic Matrix Factorization for Implicit Feedback Data.*
[40] Koren (2008): *Factorization Meets the Neighborhood*; Paterek (2007): *Improving Regularized SVD for CF.*
[41] Vgl. Hu / Koren / Volinsky (2008): *Collaborative Filtering for Implicit Feedback*, p. 1.

4 Evaluation and Experiments

4.1 Evaluation Methodologies

Evaluation within the field of RSs has gained significant prominence in response to recent reproducibility crises. In 2019, the paper by Ferrari Dacrema / Cremonesi / Jannach (2019) garnered attention by winning the Best Paper Award at the Association of Computing Machinerys (ACMs) Conference Series on Recommender Systems (RecSys)[1]. The authors highlighted that less than 50% of attempts were able to reproduce its results, underscoring the importance of reproducibility practices[2], such as providing accessible code, dataset, and documentation of the train/test split. Evaluation in RSs typically involves two main methods: *offline evaluation* and *online evaluation*. Offline evaluation utilizes pre-collected historical data to assess algorithmic performance by simulating user behavior through partitioning data into training and testing sets. This method offers efficiency and repeatability for algorithm comparison but lacks direct user interaction, thereby limiting insights into real-world performance. In contrast, online evaluation occurs in real-time, allowing users to interact naturally with the system and providing immediate feedback on effectiveness and user satisfaction, often through metrics like click-through rates and conducted via A/B testing[3].

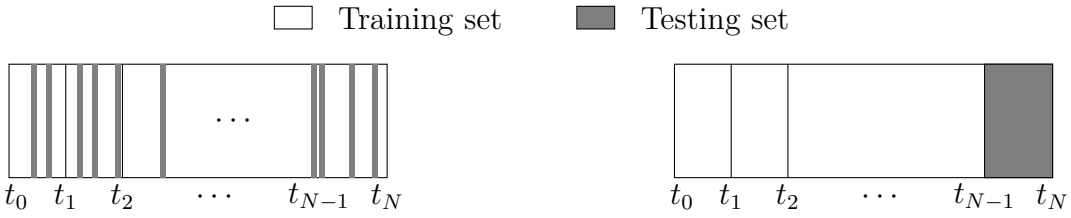
4.2 Dataset Splitting

In offline scenarios, it is crucial to simulate the target application as closely as possible[4]. This involves simulating user behavior within the evaluation task by partitioning historical data into train and test sets, where some user-item interactions are **hidden** (not removed entirely in the sense of changing the datasets shape). These hidden interactions represent the items that the algorithm should predict or recommend to the specific user. Both the train and test sets maintain the same overall size, but differ in the specific interactions

[1] ACM (2019): *RecSys Best Paper Awards*.
[2] Vgl. Zangerle / Bauer (2022): *Evaluating RSs: Survey and Framework*, 170: 9-10.
[3] Vgl. Cañamares / Castells / Moffat (2020): *Offline Evaluation Options for RSs*, p. 388.
[4] Vgl. Shani / Gunawardana (2011): *Evaluating Recommendation Systems*, p. 552.

they contain. Various strategies exist for hiding particular interactions as illustrated in 4.1[5]:

(a) Randomly hide certain elements in a specified manner, for example, allocate 80% for training and 20% for testing.
(b) **Hide** all items after a specific (or randomly selected) time-stamp.



(a) Random split strategy (b) Split after t_{N-1}

Figure 4.1: Comparison of Split Strategies

It is worth mentioning that there are even more ways to split the dataset depending on the objective. For instance, the $leave - k - out$ split, assuming each user has interacted with at least k items[6], or the $leave - k - last - out$ split based on timestamps and the latest interacted items, which combines these methods[7].

4.3 Evaluation Metrics

Given the knowledge about the items in the test set, the algorithm is evaluated to determine whether it can recommend the items from this set to the specific user. The following table 4.1 provides an overview of widely used metrics[8]. Beyond the classification in the literature, the main categories can be formulated as follows: *accuracy metrics*[9], *ranking metrics*[10], and *beyond accuracy metrics*[11]. While the first category is traditional and well-established in other research fields, the second is more pertinent to RSs, where the

[5]Vgl. Shani / Gunawardana (2011): *Evaluating Recommendation Systems*, p. 552.
[6]Vgl. Cañamares / Castells / Moffat (2020): *Offline Evaluation Options for RSs*, p. 393.
[7]Vgl. ibid., p. 393.
[8]Vgl. Shani / Gunawardana (2011): *Evaluating Recommendation Systems*, p. 584; Zangerle / Bauer (2022): *Evaluating RSs: Survey and Framework*, pp. 20–21.
[9]Vgl. Anelli et al. (2022): *Top-N Recommendation Algorithms*, p. 5.
[10]Vgl. Shani / Gunawardana (2011): *Evaluating Recommendation Systems*, p. 577.
[11]Vgl. Duricic et al. (2023): *Beyond-Accuracy: A Review on Diversity, Serendipity, and Fairness in RSs*, p. 2.

output is a list of k recommendations and ranking plays a crucial role[12]. The significance of beyond-accuracy dimensions in RSs, such as diversity, serendipity, novelty, and fairness, is increasingly recognized for their impact on user satisfaction. These aspects mitigate content overspecialization, promote user discovery, and ensure the system does not unfairly discriminate against certain users or item providers[13].

Main Category	Metrics
Accuracy Metrics	Mean Absolute Error (MAE)
	Root Mean Squared Error (RMSE)
	Precision, Recall, F1 Score, Hit Rate
Ranking Metrics	Mean Average Precision (MAP)
	Mean Reciprocal Ranking (MRR)
	Normalized Discounted Cumulative Gain (nDCG)
Beyond Accuracy Metrics	Novelty, Diversity, Coverage, Serendipity, Fairness

Table 4.1: Overview of Evaluation Metrics

For the evaluation, the RS usually generates recommendations for several users. Therefore, $\mathcal{Q} = \{Q_1, Q_2, \cdots, Q_l\}$ denotes the l the queries made by the RS. Each query $Q = \{i_1, \cdots, i_N\}$ stands for the recommended list of N items.

The MAE and RMSE measure the accuracy of rating predictions, indicating how far the predicted $\tilde{r}_{ui}$ deviates from the actual $r(u, i)$[14]. It's important to note that while these metrics have historically been applied in the literature, they do not effectively assess the performance of top-N recommendation algorithms, as noted by Cremonesi / Koren / Turrin (Vgl. 2010, p. 1). Instead, ranking evaluation metrics are more commonly used for evaluating classical item RS[15].

Accuracy Metrics

Precision

The Precision (Prec)[16] in (4.1) measures the proportion of relevant items among the recommended items. The notation @N denotes that the evaluation metric has been applied to a recommended list of N items[17]:

[12] Vgl. Shani / Gunawardana (2011): *Evaluating Recommendation Systems*, p. 577.
[13] Vgl. Duricic et al. (2023): *Beyond-Accuracy: A Review on Diversity, Serendipity, and Fairness in RSs*, p. 2.
[14] Vgl. Shani / Gunawardana (2011): *Evaluating Recommendation Systems*, p. 572.
[15] Vgl. Rendle (2021): *Item Recommendation from Implicit Feedback*, p. 147.
[16] Vgl. Liu et al. (2009): *Learning to Rank for Information Retrieval*, p. 9.
[17] Vgl. Anelli et al. (2022): *Top-N Recommendation Algorithms*, p. 8.

$$\text{Prec@N} = \frac{\text{Number of relevant items retrieved}}{\text{Number of items retrieved}} = \frac{1}{N}\sum_{i=1}^{N} rel_i \tag{4.1}$$

where rel_i equals to 1 if the i-th element is relevant and 0 otherwise.

Ranking Metrics

Normalized Discounted Cummulative Gain

By comparing the relevance of items, the ranking quality of a recommendation can be evaluated using the nDCG in (4.2)[18]. It is defined as:

$$\text{DCG}_N = \sum_{i=1}^{N} \frac{2^{rel_i} - 1}{\log_2(i+1)}, \quad \text{nDCG}_N = \frac{\text{DCG}_N}{\text{IDCG}_N} \tag{4.2}$$

whereby iDCG_k is the iDCG, which is the Discounted Cumulative Gain for the perfect ranking of the provided list.

Mean Reciprocal Ranking

The MRR[19] (4.3) on the other hand, evaluates the ranking quality by only considering the position of the *first* relevant item:

$$\text{MRR} = \frac{1}{|\mathcal{Q}|}\sum_{q=1}^{|\mathcal{Q}|} \frac{1}{rank_q} \tag{4.3}$$

where $rank_q$ is the rank position of the first relevant item for the q-th recommended list of items Q_q.

Mean Average Precision

The MAP in (4.4) calculates the mean of the average precision scores for a set of queries $\mathcal{Q}$, whereby $|R_k|$ is the number of hits in the corresponding recommended list ${Q_k}$[20]:

$$\text{AP}_k = \frac{1}{|R_k|}\sum_{i=1}^{k} \text{Precision@i} \cdot rel_i, \quad \text{MAP} = \frac{1}{|\mathcal{Q}|}\sum_{q=1}^{|\mathcal{Q}|} \text{AP}_q \tag{4.4}$$

[18] Vgl. Shani / Gunawardana (2011): *Evaluating Recommendation Systems*, p. 582; Liu et al. (2009): *Learning to Rank for Information Retrieval*, p. 9.
[19] Vgl. Shani / Gunawardana (2011): *Evaluating Recommendation Systems*, p. 576.
[20] Vgl. Anelli et al. (2022): *Top-N Recommendation Algorithms*, 5, Github Metrics.

Beyond Accuracy Metrics

Each category of beyond-accuracy metrics encompasses various metrics and measures[21]. For simplicity, the following explanation focuses only on a subset related to coverage and fairness.

Category	Metrics
Coverage	Item Coverage (IC)
Fairness	Average Recommended Popularity (ARP), Average Percentage of long-tail Items (APLT)

Table 4.2: Beyond Accuracy Evaluation Metrics

Item Coverage

The item coverage (4.5) reflects the proportion of total recommended items $|\bigcup_{Q \in \mathcal{Q}} Q_i|$ compared to the entire dataset with $|I|$ items[22].

$$\text{Coverage} = \frac{\text{Number of recommended items}}{\text{Total items in catalog}} = \frac{\left|\bigcup_{Q \in \mathcal{Q}} Q\right|}{|I|} \tag{4.5}$$

Average Recommended Popularity

The popularity of an item is determined by the total number of interactions it receives, denoted by $U(i)$ again, as introduced in 3.3.1. ARP (4.6) refers to the mean popularity of items recommended to a particular user[23].

$$\text{ARP} = \frac{1}{|\mathcal{Q}|} \sum_{Q \in \mathcal{Q}} \left(\frac{1}{|Q|} \sum_{i \in Q} U(i) \right) \tag{4.6}$$

Average Percentage of Long-Tail Items

Closely related to the former measure is the so-called "long-tail" phenomenon in RSs. The distribution of total downloads of items typically appears very skewed, as further described in 4.4. Recommending only highly popular items does not promote the discovery of the

[21] Duricic et al. (2023): *Beyond-Accuracy: A Review on Diversity, Serendipity, and Fairness in RSs*, Vgl. Kaminskas / Bridge (2016): *Diversity, Serendipity, Novelty, and Coverage.*

[22] Vgl. Duricic et al. (2023): *Beyond-Accuracy: A Review on Diversity, Serendipity, and Fairness in RSs*, p. 5; Kaminskas / Bridge (2016): *Diversity, Serendipity, Novelty, and Coverage*, 2:15.

[23] Vgl. Abdollahpouri / Burke / Mobasher (2019): *Managing popularity bias in recommender systems with personalized re-ranking*, p. 3.

entire catalog of a given e-commerce service and disregards users with niche tastes. Due to the inherent bias of RSs towards very popular items[24], the APLT in (4.7), where the set of items in the long tail is denoted by Γ, provides insight into this bias[25].

$$\text{APLT} = \frac{1}{|\mathcal{Q}|} \sum_{Q \in \mathcal{Q}} \frac{|\{i \mid i \in (Q \cap \Gamma)\}|}{|Q|} \tag{4.7}$$

4.4 Tests on the Industry Dataset

4.4.1 Data Preparation

The objective of the following tests is to apply state-of-the-art non-deep-learning-based CF techniques to a real-life dataset within an unknown timespan provided by *DJcity*. Launched in 2000, DJcity became the world's leading subscription-based digital record pool for professional DJs. A hand-curated catalog of new music, including club-ready versions, made DJcity the most trusted and respected site for DJs to discover new music.[26] The dataset, which includes download-based implicit feedback from existing users to items, has been anonymized, and is denoted as *DJC-DB*. In Table 4.3, the characteristics of the DJC-DB are outlined. For offline experiments, careful attention must be given to data selection. Prior to performing test-train splitting, it is recommended to exclude users and/or items with fewer than a specified threshold p of observed interactions[27]. This selection is known as the p-core of items/users[28]. For instance, in Bennett / Lanning, et al. (Vgl. 2007, p. 2), only users with at least 20 ratings are considered. This criterion is also applied to datasets like *MovieLens-1M*[29], which includes users with a minimum of 20 ratings, and similarly to *Epinions*[30]. For the upcoming experiments, the p-core is set to 20 to ensure both training and testing sets have a sufficient number of items.

[24] Vgl. Abdollahpouri / Burke / Mobasher (2019): *Managing popularity bias in recommender systems with personalized re-ranking*, p. 1.
[25] Vgl. ibid., p. 3.
[26] Website: About DJcity
[27] Vgl. Cañamares / Castells / Moffat (2020): *Offline Evaluation Options for RSs*, p. 3.
[28] Vgl. Anelli et al. (2022): *Top-N Recommendation Algorithms*, p. 4.
[29] Harper / Konstan (2015): *The Movielens Datasets: History and context*, Vgl.
[30] Vgl. Abdollahpouri / Burke / Mobasher (2019): *Managing popularity bias in recommender systems with personalized re-ranking*, p. 3.

	p-core	# interactions	# users	# items	sparsity
DJC-DB raw	0-core	23.545.542	70.309	37.408	0.68%
DJC-DB filtered	20-core	17.665.904	58.747	37.370	0.8%

Table 4.3: DJC-DB Characteristics

As noted in the discussion on ARP, the data is heavily skewed due to the predominance of popular items and the occurrence of a long tail, as indicated by the statistics in Table 4.4. This long tail can be analyzed in various ways, such as applying Pareto's principle as discussed in Abdollahpouri / Burke / Mobasher (Vgl. 2019, p. 1), or employing more complex long-tail models as proposed in Celma / Cano (Vgl. 2008, p. 2).

	min	max	mean	median
items per user	20	32850	300	133
user per items	21	14037	473	205

Table 4.4: DJC-DB Statistics

Anelli et al. (Vgl. 2022, p. 2) defines the short-head as comprising 20% of the total interactions. Applied to the DJC-DB, this means that 20% of the total downloads involve only 2% / 1.3% of the overall 37,370 items / 58,747 users.

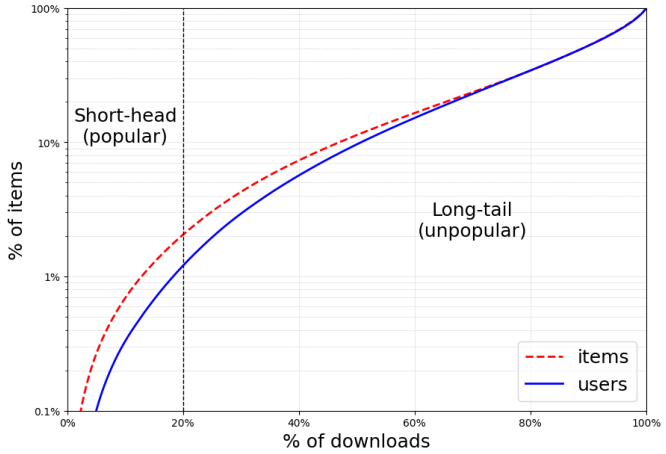


Figure 4.2: Interaction Distributions

4.4.2 Hyper-Parameter Tuning

The setup for neighborhood-based approaches is illustrated in the table below, where $U_k(i)$ / $I_k(u)$ denotes the k neighbors of all users / items that have interacted with item i / user u, respectively.

Algorithm	Steps		
	1. Similarity	2. Neighborhood	3. Prediction
UserkNN	ACOS (3.5)	$N_k(u)$	as in (3.7)
UserAsym	ACOS (3.5)	$U(i)$ and $U_k(i)$	as in (3.8)
ItemAsym	ACOS (3.9)	$I(u)$ and $I_k(u)$	as in (3.11)

Table 4.5: Best Parameter Combinations for Neighborhood Algorithms

Furthermore, the ALS MF is denoted by *ALS-MF* in tests and as an additional baseline comparison, a simple non-personalized algorithm, denoted as *MostPop*, recommends each user with the same "most popular" items based on total interactions[31].

The introduced algorithms come with several parameters to tune: α for the asynchronous cosine, β and q for the scoring function, and the neighborhood size $|N(*)|$ for neighborhood approaches, where $*$ stands for either users or items. In contrast, the *ALS-MF*, already implemented in a very efficient way through python library implicit[32], includes further parameters such as the number of latent factors l, the regularization term λ, and the a parameter for the calculation of $c(u, i)$. Therefore, a hyper-parameter search is conducted on different subsets $\widetilde{R} \subset R$ of the entire DJC-DB, regarding the computational effort. For the entire DJC-DB, a leave-k-out train/test split with $k = 10$ is used, sampling 100 random users for the test scenario. The table 4.6 presents the best parameter settings based on the nDCG@10 metric. The entire results as well as the search space of the parameter search can be found in the appendix in the figures A.1 and A.2 and tables A.1 and A.2. Additionally here, the number of *iterations* of the ALS is recorded.

[31] Vgl. Anelli et al. (2022): *Top-N Recommendation Algorithms*, p. 6.
[32] Frederickson (2018): *Fast Python Collaborative filtering for Implicit Datasets*, Vgl.

Algorithm	Parameter								
	α	β	q	l	λ	a	$\lvert N(*)\rvert$	$\lvert\widetilde{R}\rvert$	nDCG@10
UserkNN	0.5	0.0	4	/	/	/	100	500,000	0.4796
UserAsym	0.0	0.7	1	/	/	/	200	500,000	0.4808
ItemAsym	0.8	1.4	2	/	/	/	$\lvert I(u)\rvert$	100,000	0.3522
ALS-MF	/	/	/	65	1.0	1	/	5,000,000	0.3405

Table 4.6: Best Parameter Combinations for Algorithms

4.4.3 Results

The final tests are conducted on the entire DJC-DB consisting of 17.665.904 rows by making *Top-10* recommendations to 100 random users. This process is repeated **five times** for each of the top three hyper-parameter settings identified during the tuning session. Subsequently, the mean of the **best five runs** is calculated and put together in Table 4.7, sorted in descending order of nDCG values, with the best values shown in bold and the second highest values underlined.

Algorithm	Top@10						
	Pre	MRR	nDCG	MAP	IC	ARP	APLT
ALS-MF	**0.116**	**0.3587**	**0.4040**	**0.2643**	**0.01663**	5179	0.00
UserkNN	<u>0.111</u>	<u>0.3288</u>	<u>0.3934</u>	<u>0.2544</u>	0.01540	5831	0.002
UserAsymkNN	0.0980	0.2849	0.3295	0.2105	<u>0.01616</u>	5300	**0.016**
ItemAsymkNN	0.0642	0.2374	0.2838	0.2035	0.005823	9836	<u>0.0104</u>
MostPop	0.0118	0.04305	0.05797	0.04223	0.001054	12478	0.00

Table 4.7: Mean of 3 best parameter configurations on entire data set

Contrary to the results in Aiolli (Vgl. 2013, p. 5), the *ALS-MF* methods performed surprisingly well compared to the neighborhood methods. One reason for this could be that Aiolli (ibid., Vgl.) conducted their tests on the large-scale MSD, which comprises over 1 million users and $350,000$ items with a sparsity of 0.01%[33], making it several times larger than the DJC-DB. Another noteworthy aspect is the performance of nDCG in comparison to coverage and average recommendation popularity. Therefore, *ALS-MF* achieved higher coverage and accuracy, contributing to novelty and overall fairness.

[33]Vgl. Aiolli (2013): *Efficient Top-N Recommendation for Binary Datasets*, p. 1.

5 Conclusion and Future Work

With the applied non-deep learning CF techniques in MRS, a baseline on a real-life dataset with binary feedback has been established. However, in recent years, RSs have evolved from simple neighborhood methods to sophisticated latent factor models such as MF. Nowadays, deep learning models dominate the field in both academic research and industrial applications due to their ability to capture non-linear user-item relationships and incorporate diverse data sources. Traditional CF methods provide valuable insights and a solid foundation but exhibit limitations in handling complex user-item interaction patterns and sequential dependencies inherent in music consumption[1]. These methods typically assume that all user-item interactions are equally important[2], which is often not the case in real-world scenarios where formulating the recommendation problem as a sequential decision problem can more accurately reflect user-item interactions by accounting for dynamic and long-term interactions[3]. The existence of timestamps of user-item interactions allows for chronological ordering, enhancing user representations by exploring sequential patterns[4]. This type of data can be transferred into a sequence graph, enabling sequential recommendations[5]. Applying graph-learning-based algorithms like graph neural networks has inspired many recommendation models[6]. This continuation will extend the current study by incorporating state-of-the-art methods, several of which have already been highlighted by Anelli et al. (2023) but have not yet been applied to a large-scale dataset (Vgl. ibid., p. 355) in the music domain. Furthermore, the current implemented methods will be applied to at least one benchmark dataset to enhance comparability with existing studies.

[1] Vgl. Perera et al. (2020): *Analysis of MRSs*, p. 1; Fang et al. (2020): *Deep learning for sequential recommendation: Algorithms, influential factors, and evaluations*, 1:1.
[2] Vgl. Fang et al. (2020): *Deep learning for sequential recommendation: Algorithms, influential factors, and evaluations*, 1:1.
[3] Vgl. Afsar / Crump / Far (2022): *Reinforcement Learning Based Recommender Systems: A Survey*, p. 1; Fang et al. (2020): *Deep learning for sequential recommendation: Algorithms, influential factors, and evaluations*, 1:2.
[4] Vgl. Wu et al. (2022): *Graph Neural Networks in Recommender Systems: A Survey*, 97:5.
[5] Vgl. ibid., 97:7, 97:12.
[6] Vgl. ibid., 97:2.

A Appendix

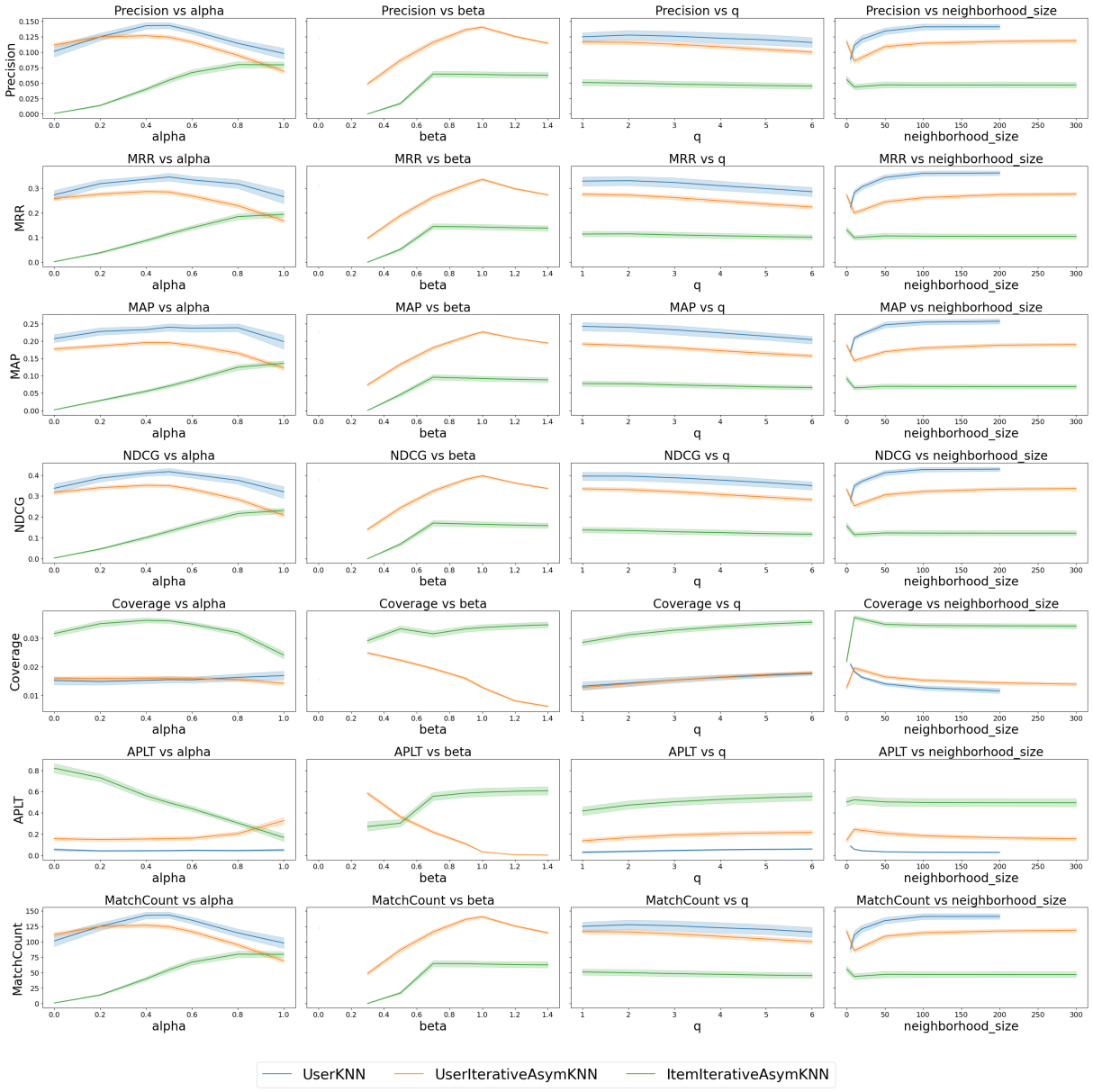


Figure A.1: Hyperparameter Analysis for Neighborhood-based Algorithms

[0]Note: neighborhood size = 0 belongs to the corresponding full neighborhoods $I(u)$ and $U(i)$

Parameter	Search Space
α	$\{0, 0.2, 0.4, 0.5, 0.6, 0.8, 1\}$
β	$\{0.3, 0.5, 0.7, 0.9, 1, 1.2, 1.4\}$
q	$\{1, 2, 3, 4, 5, 6\}$
$\lvert N(*)\rvert$	$\{\lvert I(u)\rvert, 10, 50, 100, 200, 300\}$

Table A.1: Parameter Search Space for Neighborhood-based Algorithms

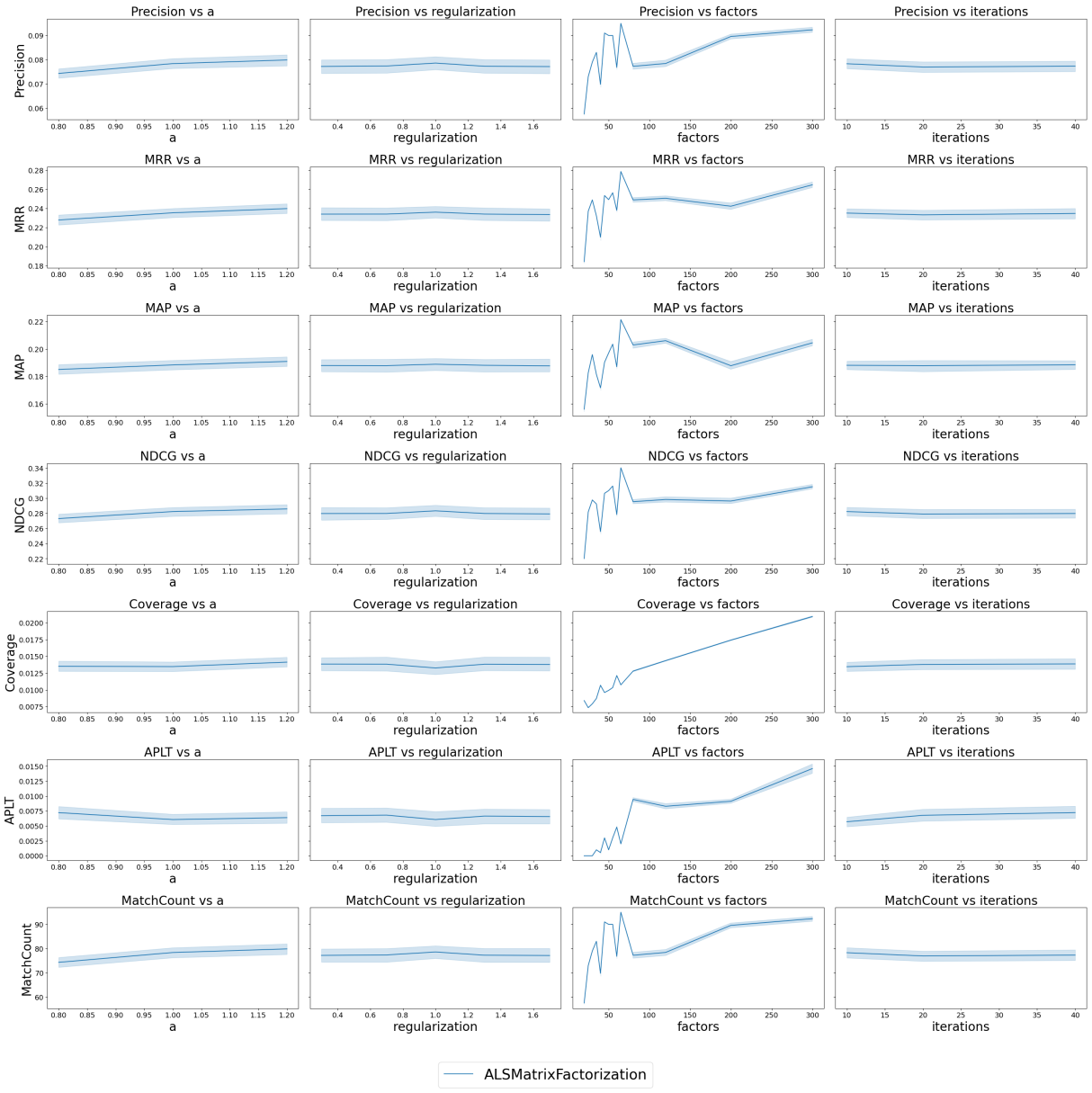


Figure A.2: Hyperparameter Analysis for *ALS-MF*

Parameter	Search Space
l	$\{25, 30, 35, 40, 45, 50, 55, 60, 65, 120, 200, 300\}$
λ	$\{0.3, 0.7, 1, 1.3, 1.7\}$
a	$\{0.8, 1, 1.2\}$

Table A.2: Parameter Search Space for *ALS-MF*

Bibliography

Abdollahpouri, H. / Burke, R. / Mobasher, B. (2019): Managing popularity bias in recommender systems with personalized re-ranking. In: *arXiv preprint arXiv:1901.07555*,

ACM (2019): RecSys Best Paper Awards. `https://recsys.acm.org/best-papers/`. [Online; accessed 27-06-2024].

Afsar, M. M. / Crump, T. / Far, B. (2022): Reinforcement Learning Based Recommender Systems: A Survey. In: *ACM Computing Surveys*, vol. 55, no. 7, pp. 1–38.

Aiolli, F. (2013): Efficient Top-N Recommendation for Very Large Scale Binary Rated Datasets. In: *Proceedings of the 7th ACM conference on Recommender systems*, pp. 273–280.

Almazro, D. / Shahatah, G. / Albdulkarim, L. / Kherees, M. / Martinez, R. / Nzoukou, W. (2010): A Survey Paper on Recommender Systems. In: *arXiv preprint arXiv:1006.5278*,

Anelli, V. W. / Bellogín, A. / Di Noia, T. / Jannach, D. / Pomo, C. (2022): Top-N Recommendation Algorithms: A Quest for the State-of-the-Art. In: *Proceedings of the 30th ACM Conference on User Modeling, Adaptation and Personalization*, pp. 121–131.

Anelli, V. W. / Malitesta, D. / Pomo, C. / Bellogín, A. / Di Sciascio, E. / Di Noia, T. (2023): Challenging the myth of graph collaborative filtering: a reasoned and reproducibility-driven analysis. In: *Proceedings of the 17th ACM Conference on Recommender Systems*, pp. 350–361.

Bennett, J. / Lanning, S., et al. (2007): The Netflix Prize. In: *Proceedings of KDD Cup and Workshop*. Volume 2007. New York, p. 35.

Cañamares, R. / Castells, P. / Moffat, A. (2020): Offline Evaluation Options for Recommender Systems. In: *Information Retrieval Journal*, vol. 23, no. 4, pp. 387–410.

Celma, Ò. / Cano, P. (2008): From hits to niches? or how popular artists can bias music recommendation and discovery. In: *Proceedings of the 2nd KDD workshop on large-scale recommender systems and the netflix prize competition*, pp. 1–8.

Cremonesi, P. / Koren, Y. / Turrin, R. (2010): Performance of Recommender Algorithms on Top-N Recommendation Tasks. In: *Proceedings of the fourth ACM conference on Recommender systems*, pp. 39–46.

Dacrema, M. F. / Cantador, I. / Fernández-Tobías, I. / Berkovsky, S. / Cremonesi, P. (2012): Design and Evaluation of Cross-Domain Recommender Systems. In: *Recommender Systems Handbook*. Springer, pp. 485–516.

Davidson, J. / Liebald, B. / Liu, J. / Nandy, P. / Van Vleet, T. / Gargi, U. / Gupta, S. / He, Y. / Lambert, M. / Livingston, B., et al. (2010): The YouTube Video Recommendation System. In: *Proceedings of the fourth ACM conference on Recommender systems*, pp. 293–296.

Duricic, T. / Kowald, D. / Lacic, E. / Lex, E. (2023): Beyond-Accuracy: A Review on Diversity, Serendipity, and Fairness in Recommender Systems based on Graph Neural Networks. In: *Frontiers in Big Data*, vol. 6.

Fang, H. / Zhang, D. / Shu, Y. / Guo, G. (2020): Deep learning for sequential recommendation: Algorithms, influential factors, and evaluations. In: *ACM Transactions on Information Systems (TOIS)*, vol. 39, no. 1, pp. 1–42.

Ferrari Dacrema, M. / Cremonesi, P. / Jannach, D. (2019): Are We Really Making Much Progress? A Worrying Analysis of Recent Neural Recommendation Approaches. In: *Proceedings of the 13th ACM Conference on Recommender Systems*, pp. 101–109.

Frederickson, B. (2018): Fast Python Collaborative filtering for Implicit Datasets. In: *URL https://github. com/benfred/implicit*,

Funk, S. (2006): Netflix Update: Try This At Home. `https://sifter.org/simon/journal/20061211.html`. [Online; accessed 08-06-2024].

Goldberg, D. / Nichols, D. / Oki, B. M. / Terry, D. (1992): Using Collaborative Filtering to Weave an Information Tapestry. In: *Communications of the ACM*, vol. 35, no. 12, pp. 61–70.

Greentree, M. (2023): Why you should Download Music from your preferred Streaming Services. `https://www.subjectivesounds.com/tips/why-you-should-download-music-from-your-preferred-streaming-service`. [Online; accessed 30-05-2024].

Group, U. M. (2024): An Open Letter to the Artist and Songwriter Community Why We Must Call Time Out on TikTok. `https://www.universalmusic.com/an-open-letter-to-the-artist-and-songwriter-community-why-we-must-call-time-out-on-tiktok/`. [Online; accessed 30-05-2024].

Harper, F. M. / Konstan, J. A. (2015): The Movielens Datasets: History and context. In: *Acm transactions on interactive intelligent systems (tiis)*, vol. 5, no. 4, pp. 1–19.

He, X. / Liao, L. / Zhang, H. / Nie, L. / Hu, X. / Chua, T.-S. (2017): Neural Collaborative Filtering. In: *Proceedings of the 26th international conference on world wide web*, pp. 173–182.

Herlocker, J. L. / Konstan, J. A. / Borchers, A. / Riedl, J. (1999): An Algorithmic Framework for Performing Collaborative Filtering. In: *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 230–237.

Hu, Y. / Koren, Y. / Volinsky, C. (2008): Collaborative Filtering for Implicit Feedback Datasets. In: *2008 Eight IEEE International Conference on Data Mining*. IEEE, pp. 263–272.

Jannach, D. / Jugovac, M. (2019): Measuring the Business Value of Recommender Systems. In: *ACM Transactions on Management Information Systems (TMIS)*, vol. 10, no. 4, pp. 1–23.

Jannach, D. / Quadrana, M. / Cremonesi, P. (2022): Session-Based Recommender Systems. In: *Recommender Systems Handbook*. Springer, pp. 301–334.

Johnson, C. C. et al. (2014): Logistic Matrix Factorization for Implicit Feedback Data. In: *Advances in Neural Information Processing Systems*, vol. 27, no. 78, pp. 1–9.

Kaminskas, M. / Bridge, D. (2016): Diversity, Serendipity, Novelty, and Coverage: A Survey and Empirical Analysis of Beyond-Accuracy Objectives in Recommender Sys-

tems. In: *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 7, no. 1, pp. 1–42.

Koren, Y. (2008): Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering Model. In: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 426–434.

Koren, Y. / Bell, R. / Volinsky, C. (2009): Matrix Factorization Techniques for Recommender Systems. In: *Computer*, vol. 42, no. 8, pp. 30–37.

Koren, Y. / Rendle, S. / Bell, R. (2021): Advances in Collaborative Filtering. In: *Recommender systems handbook*, pp. 91–142.

Li, Y. / Liu, K. / Satapathy, R. / Wang, S. / Cambria, E. (2024): Recent Developments in Recommender Systems: A Survey. In: *IEEE Computational Intelligence Magazine*, vol. 19, no. 2, pp. 78–95.

Liu, T.-Y. et al. (2009): Learning to Rank for Information Retrieval. In: *Foundations and Trends® in Information Retrieval*, vol. 3, no. 3, pp. 225–331.

McFee, B. / Bertin-Mahieux, T. / Ellis, D. P. / Lanckriet, G. R. (2012): The Million Song Dataset Challenge. In: *Proceedings of the 21st International Conference on World Wide Web*, pp. 909–916.

Mnih, A. / Salakhutdinov, R. R. (2007): Probabilistic Matrix Factorization. In: *Advances in neural information processing systems*, vol. 20.

Nikolakopoulos, A. N. / Ning, X. / Desrosiers, C. / Karypis, G. (2021): Trust Your Neighbors: A Comprehensive Survey of Neighborhood-Based Methods for Recommender Systems. In: *Recommender systems handbook*, pp. 39–89.

Paterek, A. (2007): Improving Regularized Singular Value Decomposition for Collaborative Filtering. In: *Proceedings of KDD cup and workshop*. Volume 2007. 2007, pp. 5–8.

Perera, D. / Rajaratne, M. / Arunathilake, S. / Karunanayaka, K. / Liyanage, B. (2020): A Critical Analysis of Music Recommendation Systems and New Perspectives. In: *Human Interaction, Emerging Technologies and Future Applications II: Pro-*

ceedings of the 2nd International Conference on Human Interaction and Emerging Technologies: Future Applications (IHIET–AI 2020), April 23-25, 2020, Lausanne, Switzerland. Springer, pp. 82–87.

Rendle, S. (2021): Item Recommendation from Implicit Feedback. In: *Recommender Systems Handbook*. Springer, pp. 143–171.

Rendle, S. / Freudenthaler, C. / Gantner, Z. / Schmidt-Thieme, L. (2012): BPR: Bayesian Personalized Ranking from Implicit Feedback. In: *arXiv preprint arXiv:1205.2618*,

Rendle, S. / Krichene, W. / Zhang, L. / Anderson, J. (2020): Neural Collaborative Filtering vs. Matrix Factorization Revisited. In: *Proceedings of the 14th ACM Conference on Recommender Systems*, pp. 240–248.

Rendle, S. / Krichene, W. / Zhang, L. / Koren, Y. (2021): Revisiting the Performance of iALS on Item Recommendation Benchmarks. CoRR abs/2110.14037 (2021). In: *arXiv preprint arXiv:2110.14037*,

Ricci, F. / Rokach, L. / Shapira, B. (2021): Recommender Systems: Techniques, Applications, and Challenges. In: *Recommender systems handbook*, pp. 1–35.

Sarwar, B. / Karypis, G. / Konstan, J. / Riedl, J. T. (2000): Application of Dimensionality Reduction in Recommender System – A Case Study. In.

Schedl, M. / Knees, P. / McFee, B. / Bogdanov, D. (2021): Music Recommendation Systems: Techniques, Use Cases, and Challenges. In: *Recommender Systems Handbook*. Springer, pp. 927–971.

Shani, G. / Gunawardana, A. (2011): Evaluating Recommendation Systems. In: *Recommender systems handbook*, pp. 257–297.

TikTok (2024): TikTok Music. `https://music.tiktok.com/`. [Online; accessed 30-05-2024].

Volkovs, M. / Yu, G. W. (2015): Effective Latent Models for Binary Feedback in Recommender Systems. In: *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pp. 313–322.

Wu, S. / Sun, F. / Zhang, W. / Xie, X. / Cui, B. (2022): Graph Neural Networks in Recommender Systems: A Survey. In: *ACM Computing Surveys*, vol. 55, no. 5, pp. 1–37.

Zangerle, E. / Bauer, C. (2022): Evaluating Recommender Systems: Survey and Framework. In: *ACM computing surveys*, vol. 55, no. 8, pp. 1–38.

Statement of authorship

I hereby certify that I have authored this document independently and without using other than the indicated assistance. The thoughts taken directly or indirectly from external sources are marked as such. The work has not been submitted in the same or a similar form to any other examination authority and has not yet been published. I am aware that an untruthful statement will have legal consequences.

I agree that the work will be examined for contained plagiarism with the help of a plagiarism detection service.

Dresden, on 7th July 2024