# Evaluation of Session-based Recommendation Algorithms

MALTE LUDEWIG, TU Dortmund, Germany
DIETMAR JANNACH, AAU Klagenfurt, Austria

Recommender systems help users find relevant items of interest, for example on e-commerce or media streaming sites. Most academic research is concerned with approaches that personalize the recommendations according to long-term user profiles. In many real-world applications, however, such long-term profiles often do not exist and recommendations therefore have to be made solely based on the observed behavior of a user during an ongoing session. Given the high practical relevance of the problem, an increased interest in this problem can be observed in recent years, leading to a number of proposals for *session-based recommendation algorithms* that typically aim to predict the user's immediate next actions.

In this work, we present the results of an in-depth performance comparison of a number of such algorithms, using a variety of datasets and evaluation measures. Our comparison includes the most recent approaches based on recurrent neural networks like GRU4REC, factorized Markov model approaches such as FISM or FOSSIL, as well as simpler methods based, e.g., on nearest neighbor schemes. Our experiments reveal that algorithms of this latter class, despite their sometimes almost trivial nature, often perform equally well or significantly better than today's more complex approaches based on deep neural networks. Our results therefore suggest that there is substantial room for improvement regarding the development of more sophisticated session-based recommendation algorithms.[1]

CCS Concepts: • **Information systems** → **Recommender systems**; • **General and reference** → **Evaluation**;

Additional Key Words and Phrases: Session-based Recommendation; Sequential Recommendation; Deep Learning; Factorized Markov Models, Nearest Neighbors

## 1 INTRODUCTION

Many of today's online services use recommender systems to point their users or site visitors to additional items that might be of interest to them. In academic research, the majority of works is focusing on techniques that rely on long-term preference models to determine the items to be presented to the user. However, in many application domains of recommender systems, such long-term user models are often not available for a larger fraction of the users, e.g., because they are first-time visitors or because they are not logged in. Consequently, suitable recommendations have to be determined based on other types of information, usually the user's most recent interactions with the site or application. Recommendation techniques that rely solely on the user's actions in an ongoing session and which adapt their recommendations to the user's actions are called *session-based* recommendation approaches [Quadrana et al. 2018].

Amazon's "Customers who bought … also bought" recommendations can be considered an extreme case of such a session-based approach. In this case, the recommendations are seemingly only dependent on the item that is currently viewed by the user (and the purchasing patterns of the community). A number of other techniques were proposed in the research literature, which do not limit themselves to the very last action, but consider some or all user actions since the session started. Some of these techniques only consider which events happened; others, in contrast, in addition take the sequence of events into account in their algorithms. Besides the e-commerce

[1] A preliminary comparison of sequential recommendation algorithms was presented in our own previous work in [Jannach and Ludewig 2017; Kamehkhosh et al. 2017].

Authors' addresses: Malte Ludewig, TU Dortmund, Otto-Hahn-Straße 12, Dortmund, 44227, Germany, malte.ludewig@tu-dortmund.de; Dietmar Jannach, AAU Klagenfurt, Universitätsstraße 65-67, Klagenfurt am Wörthersee, 9020, Austria, dietmar.jannach@aau.at.

domain, a number of other application fields were in the focus in the literature, among them in particular music, web page navigation, or travel and tourism.

In academia, sequential recommendation problems are typically operationalized as the task of predicting the next user action. Experimental evaluations are usually based on larger, time-ordered logs of user actions, e.g., on the users' item viewing and purchase activities on an e-commerce shop or on their listening history on a music streaming site. From an algorithmic perspective, early approaches to predict the next user actions were based, for example, on sequential pattern mining techniques. Later on, different types of more sophisticated methods based on Markov models were proposed and successfully applied to the problem. Finally, in the most recent years, the use of deep learning approaches based on artificial neural networks was explored as another solution. Recurrent Neural Networks (RNN), which are capable of learning models from sequentially ordered data, are a "natural choice" for this problem, and significant advances regarding the prediction accuracy of such algorithms were reported in the recent literature [Devooght and Bersini 2017; Hidasi and Karatzoglou 2017; Hidasi et al. 2016a,b; Tan et al. 2016].

Despite the growing number of papers on the topic in recent years, no true "standard" benchmark data sets or evaluation protocols exist in the community. Therefore, it remains difficult to compare the various algorithmic proposals, in particular as often different baseline algorithms are used in the papers. And, for some of them it is also unclear if they are particularly strong. In our previous work [Jannach and Ludewig 2017; Kamehkhosh et al. 2017], we could, for example, demonstrate that a comparably simple k-nearest-neighbor method leads to similar or even better accuracy results than a modern deep learning approach.

To establish a common base for future research, we performed an in-depth performance comparison across multiple domains and datasets, which involved a number of comparably simple as well as more sophisticated algorithms from the recent literature. Our results show that computationally and conceptually simple methods often lead to predictions that are similarly accurate or even better than those of today's most recent techniques based on deep learning models. As a consequence, we argue that researchers should take these simpler methods as alternative baselines into account when developing novel session-based recommendation algorithms. Furthermore, our results suggest that there is still substantial room for improvement regarding the development of more sophisticated session-based recommendation algorithms.

This paper extends our previous works presented in [Jannach and Ludewig 2017; Kamehkhosh et al. 2017] in a number of ways. First, we made experiments for a larger number of datasets from different domains, using a richer set of performance measures. Second, we included recent *sequential* recommendation algorithms like FISM and FOSSIL [He and McAuley 2016; Kabbur et al. 2013] in the evaluation as well as the latest version of GRU4REC [Hidasi et al. 2016a]. Third, we designed a number of additional sequence-aware similarity measures for the previously proposed session-based nearest neighbor method, which in most cases lead to significant performance gains. Finally, we also propose a new method called *Session-based Matrix Factorization (SFM)*, which yields good results in some of the tested application domains.

The paper is organized as follows. Next, in Section 2, we discuss previous works and typical application areas of session-based recommendation approaches. In Section 3, we provide technical details about the algorithms that were compared in our work. Section 4 describes our evaluation setup and Section 5 the outcomes of our experiments. To foster reproducible research on the topic, we share the code of the used evaluation framework and the compared algorithms online.[2]

[2] https://www.dropbox.com/sh/dbzmtq4zhzbj5o9/AACldzQWbw-igKjcPTBI6ZPAa?dl=0

## 2 REVIEW OF SESSION-BASED RECOMMENDATION APPROACHES

Most of the approaches for session-based recommendation proposed in the literature implement some form of *sequence learning*, see also [Quadrana et al. 2018] for a recent survey on the more general class of sequence-aware recommenders. Early approaches were based on the identification of *frequent sequential patterns*, which can be used at recommendation time to predict a user's next action. These early approaches were applied, for example, in the context of predicting the online navigation behavior of users [Mobasher et al. 2002]. Later on, such pattern mining techniques were also used for next-item recommendation problems in e-commerce or the music domain [Bonnin and Jannach 2014; Hariri et al. 2012; Yap et al. 2012].

While frequent pattern techniques are easy to implement and lead to interpretable models, the mining process can be computationally demanding. At the same time, finding good algorithm parameters, in particular a suitable minimum support threshold, can be challenging. Finally, in some application domains it seems that using frequent item sequences does not lead to better recommendations than when using simpler item co-occurrence patterns [Bonnin and Jannach 2014]. In the context of this work, we investigate both sequential and co-occurrence patterns in their simplest forms as baselines.

In many newer works, more sophisticated sequence learning approaches were proposed that implement some form of *sequence modeling*. Such sequence modeling approaches are usually based on Markov Chain (MC) models [Garcin et al. 2013; He et al. 2009; Hosseinzadeh Aghdam et al. 2015; Mcfee and Lanckriet 2011], reinforcement learning (RL) and Markov Decision Processes (MDP) [Moling et al. 2012; Shani et al. 2005; Tavakol and Brefeld 2014], or Recurrent Neural Networks (RNN) [Du et al. 2016; Hidasi et al. 2016a,b; Liu et al. 2016; Soh et al. 2017; Song et al. 2016; Sordoni et al. 2015; Twardowski 2016; Yu et al. 2016; Zhang et al. 2014]. Again, the typical application scenarios of these methods include the e-commerce and the music recommendation domain.

An early approach based on an MDP model was proposed by [Shani et al. 2005]. It demonstrated the value of using sequential data in an e-commerce scenario, but also showed that models based on Markov Chains often cannot be directly applied due to data sparsity. Therefore, [Shani et al. 2005] proposed different heuristics to overcome the problem. An additional challenge when using this type of models is to decide how many preceding interactions should be considered when predicting the next one. Some authors therefore use a mixture of Variable-order Markov Models (VMMs) or *context-trees* to consider sequences of different lengths [Garcin et al. 2013; He et al. 2009]. Other works, for example by [Hosseinzadeh Aghdam et al. 2015], rely on Hidden Markov Models (HMMs) to overcome certain limitations of plain Markov Chain models. In [Moling et al. 2012; Shani et al. 2005], reinforcement learning was implemented based on MDPs, which made it possible to also consider the reward for the shop in the recommendation process. To deal with the problem of the explosion of the state space in such scenarios, [Tavakol and Brefeld 2014] proposed to model the state space based on the sequence of item attributes in order to predict the characteristics of the next item that the user will consider. In the context of the comparative analysis presented in this paper, we limit ourselves to a simple MC-based method as a baseline, in particular because some techniques like the one discussed by [Tavakol and Brefeld 2014] require the existence of knowledge about certain item attributes.

The most recent works on sequence modeling are based on RNNs. [Zhang et al. 2014], for example, used them for the prediction of user clicks in an advertisement scenario. [Hidasi et al. 2016a] were among the first to explore Gated Recurrent Units (GRUs) as a special form of RNNs for the prediction of the next user action in a session. Their method called GRU4REC was later on extended in different ways in [Hidasi and Karatzoglou 2017; Hidasi et al. 2016b] and [Quadrana et al.

2017]. While [Hidasi et al. 2016a] reported substantial performance improvements over an *item-based* k-nearest-neighbor (kNN) method when using their first version of GRU4REC, our previous work [Jannach and Ludewig 2017] showed that a *session-based* nearest neighbor method also leads to competitive accuracy results for the same problem setting. Since GRU4REC was substantially improved since its initial version, we include the latest version of the method proposed by [Hidasi and Karatzoglou 2017] in the performance comparison reported in this paper. Furthermore, given our observations regarding the often competitive performance of conceptually simpler methods we designed a number of variations of the basic session-based nearest neighborhood method from [Jannach and Ludewig 2017], which we also considered in the experiments.

Another family of sequence modeling approaches relies on *distributed item representations*, e.g., in the form of latent Markov embeddings [Chen et al. 2012, 2013; Feng et al. 2015; Wu et al. 2013] or distributional embeddings [Baeza-Yates et al. 2015; Djuric et al. 2014; Grbovic et al. 2015; Reddy et al. 2016; Tagami et al. 2015; Vasile et al. 2016; Zheleva et al. 2010]. Embeddings are dense, lower-dimensional representations that are derived from sequentially ordered data and encode transition probabilities based on the observations in the original data. They were applied, for example, in the domains of next-track music recommendation [Chen et al. 2012; Zheleva et al. 2010], recommendation of learning courses [Reddy et al. 2016], or next point-of-interest (POI) recommendation [Feng et al. 2015]. However, a general challenge when using item embeddings is that they can be computationally demanding and sometimes require substantial amounts of training data to be effective. In the context of our work, we experimented with item embeddings as an alternative representation of the user sessions. However, the usage of embeddings did not lead to an improvement in terms of the prediction accuracy for our problem settings, which is why we do not report the detailed outcomes of these experiments in this paper.

To overcome the limitations of pure sequence learning methods, a number of *hybrid methods* were proposed that, for instance, combine the advantages of matrix factorization techniques with sequence modeling approaches in the form of Factorized Markov Chains [Cheng et al. 2013; He et al. 2016; He and McAuley 2016; Lian et al. 2013; Rendle et al. 2010]. [Rendle et al. 2010] proposed the Factorized Personalized Markov Chain (FPMC) approach as an early method for next-item recommendations in e-commerce settings, where user interactions are represented as a three-dimensional tensor (user, current item, next-item). Later on, variations of FPMC were proposed and successfully applied for a variety of application problems, e.g., by [Kabbur et al. 2013] and [He and McAuley 2016]. Other hybrid techniques that, for example, use some form of clustering or Latent Dirichlet Allocation in combination with a sequential recommendation method were proposed, e.g., in [Hariri et al. 2012; Natarajan et al. 2013; Song et al. 2015], for the problems of next-track or next-app recommendation. In our experimental evaluation, we include both the FPMC method by [Rendle et al. 2010] as well as the recent variations and improvements described by [Kabbur et al. 2013] (FISM) and [He and McAuley 2016] (FOSSIL).

Besides pure *session-based* techniques, which solely consider a user's action of the ongoing session, there are also approaches that consider previous interactions of the same user in the recommendation process. Such techniques are called *session-aware* according to the terminology of [Quadrana et al. 2018]. Examples of such works include [Baeza-Yates et al. 2015; Billsus et al. 2000; Hariri et al. 2012; Jannach et al. 2017a, 2015a; Quadrana et al. 2017], and session-aware approaches were applied for various application domains like e-commerce, music, news, or next-app recommendation. Considering longer-term user preferences in these papers shows to be helpful to improve the recommendations in the current, ongoing session. In some cases, like in [Jannach et al. 2015a], it however turns out that the short-term user intents are much more important than the longer-term models. In the research presented herein, we therefore exclusively focus on

session-based recommendation scenarios. We however consider the combination of long-term and short-term models as an important area for future research.

## 3 DETAILS OF THE INVESTIGATED METHODS

Based on these discussion, we include the following four types of techniques in our comparison of session-based recommendation algorithms: simple heuristics as baseline methods, nearest-neighbor techniques, recurrent neural networks, and factorization-based methods. The main input to all methods is a training set of past user sessions, where each session consists of a set of sequentially ordered actions of a given type, e.g., an item view event in an online shop or a consumption event on a media streaming site. The models learned by the algorithms can then be used to predict the next event in a given user session in the test set. In our evaluations, we follow a pragmatic approach to determine user sessions—in case these are not provided in the datasets—and use user inactivity times to determine session borders. The details for each dataset are described later in this paper.

Regarding the choice of the algorithms, we focus on collaborative filtering methods based on implicit feedback signals, e.g., item view or music listening events. Depending on the specific application, content-based and hybrid algorithms can be designed that use additional meta-data or content features. Since these features are domain specific and such features are only available for very few of our datasets, we limit ourselves to methods that do not rely on such types of data in this paper.

### 3.1 Baseline Methods

We include the following baseline techniques in our comparison: a method that we call Simple Association Rules (AR), first-order Markov Chains (MC), and a method that we named Sequential Rules (SR). All baselines implement very simple prediction schemes, have a low computational complexity both for training and recommending, and only consider the very last item of a current user session to make the predictions. Furthermore, we include a prediction method based on Bayesian Personalized Ranking (BPR-MF) proposed by [Rendle et al. 2009] as an alternative baseline.

*3.1.1 Simple Association Rules (AR).* Simple Association Rules (AR) are a simplified version of the association rule mining technique [Agrawal et al. 1993] with a maximum rule size of two. The method is designed to capture the frequency of two co-occurring events, e.g., "Customers who bought … also bought". Algorithmically, the rules and their corresponding importance are "learned" by counting how often the items $i$ and $j$ occurred together in a session of any user.

Let a session $s$ be a chronologically ordered tuple of item click events $s = (s_1, s_2, s_3, \ldots, s_m)$ and $S_p$ the set of all past sessions. Given a user's current session $s$ with $s_{|s|}$ being the last item interaction in $s$, we can define the score for a recommendable item $i$ as follows, where the indicator function $1_{\text{EQ}}(a, b)$ is 1 in case $a$ and $b$ refer to the same item and 0 otherwise.

$$score_{\text{AR}}(i, s) = \frac{1}{\sum_{p \in S_p} \sum_{x=1}^{|p|} 1_{\text{EQ}}(s_{|s|}, p_x) \cdot (|p| - 1)} \sum_{p \in S_p} \sum_{x=1}^{|p|} \sum_{y=1}^{|p|} 1_{\text{EQ}}(s_{|s|}, p_x) \cdot 1_{\text{EQ}}(i, p_y) \tag{1}$$

In Equation 1, the sums at the right-hand side represent the counting scheme. The term at the left-hand side normalizes the score by the number of total rule occurrences originating from the current item $s_{|s|}$. A list of recommendations returned by the AR method then contains the items with the highest scores in descending order. No minimum support or confidence thresholds are applied. In our implementation, as shared online, we create the rules in one iteration over the training data and store them (sorted by weight) in nested maps to support fast lookups in the recommendation phase. With this data structure, top-n recommendations can be created almost instantaneously.

*3.1.2 Markov Chains (MC).* The MC baseline can be seen as a variant of AR with a focus on sequences in the data. Here, the rules are extracted from a first-order Markov Chain, see [Norris 1997], which describes the transition probability between two *subsequent* events in a session. In our baseline approach, we simply count how often users viewed item $q$ immediately after viewing item $p$. Technically, the score for an item $i$ given the current session $s$ with the last event $s_{|s|}$ can be defined as a simplified version of Equation 1:

$$score_{\text{MC}}(i,s) = \frac{1}{\sum_{p \in S_p} \sum_{x=1}^{|p|-1} 1_{\text{EQ}}(s_{|s|}, p_x)} \sum_{p \in S_p} \sum_{x=1}^{|p|-1} 1_{\text{EQ}}(s_{|s|}, p_x) \cdot 1_{\text{EQ}}(i, p_{x+1}) \tag{2}$$

where the function $1_{\text{EQ}}(a,b)$ again indicates whether $a$ and $b$ refer to the same item or not. Here, with the right-hand side of the formula, we count how often item $i$ appears immediately after $s_{|s|}$. The normalization term transforms the absolute count into a relative transition probability. In line with AR, in our implementation the rules and weights are recorded in nested maps in one single iteration over the training data to ensure short training times and to support the fast generation of the recommendations.

*3.1.3 Sequential Rules (SR).* Finally, the SR method as proposed in [Kamehkhosh et al. 2017] is a variation of MC or AR respectively. It also takes the order of actions into account, but in a less restrictive manner. In contrast to the MC method, we create a rule when an item $q$ appeared after an item $p$ in a session even when other events happened between $p$ and $q$.

When assigning weights to the rules, we consider the number of elements appearing between $p$ and $q$ in the session. Specifically, we use the weight function $w_{\text{SR}}(x) = 1/(x)$, where $x$ corresponds to the number of steps between the two items.[3] Given the current session $s$, the SR method calculates the score for the target item $i$ as follows:

$$score_{\text{SR}}(i,s) = \frac{1}{\sum_{p \in S_p} \sum_{x=2}^{|p|} 1_{\text{EQ}}(s_{|s|}, p_x) \cdot x} \sum_{p \in S_p} \sum_{x=2}^{|p|} \sum_{y=1}^{x-1} 1_{\text{EQ}}(s_{|s|}, p_y) \cdot 1_{\text{EQ}}(i, p_x) \cdot w_{\text{SR}}(x-y) \tag{3}$$

In contrast to Equation 1 for AR, the third inner sum only considers indices of previous item view events for each session $p$. In addition, the weighting function $w_{\text{SR}}(x)$ is added. Again, we normalize the absolute score by the total number of rule occurrences for the current item $s_{|s|}$. As for AR and MC, the algorithm was implemented using nested sorted maps, which can be created in a single iteration over the training data.

*3.1.4 Bayesian Personalized Ranking (BPR-MF).* To make our results comparable with previous research, we finally include a prediction method based on BPR-MF as a baseline in our experiments.[4] BPR-MF proposed by [Rendle et al. 2009] is a learning-to-rank method designed for implicit-feedback recommendation scenarios. The method is usually applied for matrix-completion problem formulations based on longer-term user-item interactions. In BPR-MF the matrix is factorized into two smaller matrices of latent user and item features ($W$ and $H$), optimizing the following criterion:

$$BPR_{OPT} = \sum_{(u,i,j) \in D_S} ln\, \sigma(r_{u,i} - r_{u,j}) - \lambda_\Theta ||\Theta||^2 \tag{4}$$

In the above formula, a ranking $r_{u,i}$ for user $u$ and item $i$ is approximated with the dot product of the corresponding rows in the matrices $W$ and $H$ ($r_{u,i} = \langle W_u, H_i \rangle$). The model parameters $\Theta = (W, H)$ are learned using stochastic gradient descent in multiple iterations over the dataset $D_S$, which consists of triplets of the form $(u,i,j)$, where $(u,i)$ is a positive feedback pair and $(u,j)$ is a sampled

[3] Other weighting functions, e.g., with a logarithmic decay, are possible as well. Using the linear function however led to the best results, on average, in our experiments.

[4] The method was proposed by Hidasi et al. in the context of the GRU4REC method.

negative example. The optimization criterion in Equation 4 aims to rank the positive sample $(u, i)$ higher than a non-observed sample $(u, j)$.

To apply the method for the session-based recommendation scenario—where there are no long-term user profiles—we attribute each session in the training set to a different user, i.e., each session corresponds to a user in the user-item interaction matrix. At prediction time, we use the average of the latent item vectors of the current session so far as the user vector.

Generally, BPR and other methods designed for the matrix-completion problems in their original form, i.e., without considering the short-term session context, do not lead to competitive results in session-based recommendation scenarios, as reported, e.g., in [Jannach et al. 2015a]. Therefore, we do not consider such algorithms, e.g., traditional matrix factorization techniques, as baselines in our experiments.

### 3.2 Nearest Neighbors

Despite their simplicity, nearest-neighbor-based approaches often perform surprisingly well as discussed, e.g., by [Verstrepen and Goethals 2014] and in our previous work [Jannach and Ludewig 2017; Kamehkhosh et al. 2017]. We, therefore, include different nearest neighbor schemes in our comparison. First, we consider a more traditional item-based variant, which was also employed as a baseline method by [Hidasi et al. 2016a]. Furthermore, we evaluate three variations of a more recent session-based nearest neighbor technique in our experiments.

*3.2.1 Item-based kNN (IKNN).* The IKNN method as used in [Hidasi et al. 2016a] only considers the last element in a given session and then returns those items as recommendations that are most similar to it in terms of their co-occurrence in other sessions. Technically, each item is encoded as a binary vector, where each element corresponds to a session and is set to "1" in case the item appeared in the session. The similarity of two items can then be determined, e.g., using the cosine similarity measure, and the number of neighbours $k$ is implicitly defined by the desired recommendation list length.

Conceptually, the method implements a certain form of a "Customers who bought … also bought" scheme like the AR baseline. The use of the cosine similarity metric however makes it less susceptible to popularity biases. Although item-to-item approaches are comparably simple, they are commonly used in practice and sometimes considered a strong baselines [Davidson et al. 2010; Linden et al. 2003]. In terms of the technical implementation, all similarity values can be pre-computed and sorted in the training process to ensure fast responses at recommendation time.[5]

*3.2.2 Session-based kNN (SKNN).* Instead of considering only the last event in the current session, the SKNN method compares the entire current session with the past sessions in the training data to determine the items to be recommended, see also [Bonnin and Jannach 2014; Hariri et al. 2012; Lerche et al. 2016]. Technically, given a session $s$, we first determine the $k$ most similar past sessions (neighbors) $N_s$ by applying a suitable session similarity measure, e.g., the Jaccard index or cosine similarity on binary vectors over the item space [Bonnin and Jannach 2014]. In our experiments, the binary cosine similarity measure led to the best results. As in [Jannach and Ludewig 2017], using $k = 500$ as the number of neighbors to consider led to good performance results for many datasets. Next, given the current session $s$, its neighbors $N_s$, and the chosen similarity function $sim(s_1, s_2)$ for two sessions $s_1$ and $s_2$, the recommendation score for each item $i$ can as defined by [Bonnin and Jannach 2014]:

$$score_{\text{SKNN}}(i, s) = \Sigma_{n \in N_s} sim(s, n) \cdot 1_n(i) \tag{5}$$

[5]We use the implementation published at https://github.com/hidasib/GRU4Rec.

Here, the indicator function $1_n(i)$ returns 1 if session $n$ contains item $i$ and 0 otherwise.

*Scalability Considerations.* Given a current session $s$, we cannot scan a potentially large set of past sessions for possible neighbors in an online recommendation scenario. Therefore, in our implementation of the algorithm, as described in [Jannach and Ludewig 2017] in more detail, we rely on pre-computed in-memory index data structures and on neighborhood sampling to enable fast recommendation responses. The index is used to quickly locate past sessions that contain a certain item, i.e., the index allows us to retrieve *possible* neighbor sessions that contain at least one element of the current session through fast lookup operations. On the other hand, sampling only a smaller fraction of all past sessions in our experiments as potential neighbors has shown to lead to comparably small accuracy compromises. In fact, in some domains like e-commerce, only looking for neighbors in the most recent sessions—thereby capturing recent trends in the community—proved to be very effective [Jannach et al. 2017b] and led to even better results than when all past sessions were taken into account.

Our nearest neighbor implementations, therefore, have an additional parameter $m$, which determines the size of the sample from which the neighbors of a target session are taken. In the experiments reported in [Jannach and Ludewig 2017], it was, for example, sufficient to consider only the 1,000 most recent sessions from several million existing ones.

*Sequence-Aware Extensions: V-SKNN, S-SKNN, and SF-SKNN.* The described SKNN method does not consider the order of the elements in a session when using the Jaccard index or cosine similarity as a distance measure. Since the order of the elements might, however, be relevant in some domains and since the user preferences might change within a single session depending on the already seen items, we propose three variations of the SKNN method.[6]

- Vector Multiplication Session-Based kNN (V-SKNN): The idea of this variant is to put more emphasis on the more recent events of a session when computing the similarities. Instead of encoding a session as a *binary* vector as described above, we use real-valued vectors to encode the current session. Only the very last element of the session obtains a value of "1"; the weights of the other elements are determined using a linear decay function that depends on the position of the element within the session, where elements appearing earlier in the session obtain a lower weight. As a result, when using the *dot product* as a similarity function between the current weight-encoded session and a binary-encoded past session, more emphasis is given to elements that appear later in the sessions.
- Sequential Session-based kNN (S-SKNN): This variant also puts more weight on elements that appear later in the session. This time, however, we achieve the effect with the following scoring function:

$$score_{\text{S-SKNN}}(i, s) = \Sigma_{n \in N_s} sim(s, n) \cdot w_n(s) \cdot 1_n(i) \tag{6}$$

  Here, the indicator function $1_n(i)$ is complemented with a weighting function $w_n(i, s)$, which takes the order of the events in the current session $s$ into account. The weight $w_n(i, s)$ increases when the more recent items of the current session $s$ also appeared in a neighboring session $n$. If an item $s_x$ is the most recent item of the current session $s$ that also appears in the neighbor session $n$, then the weight will be defined as $w_n(s) = x/|s|$, where the index $x$ indicates the

[6] We made additional experiments using other ways of encoding sequential information, e.g., by using embeddings of sessions and items with the popular *Word2Vec* and *Doc2Vec* approaches. However, none of these variations led to better accuracy results than the SKNN method in our experiments. We therefore omit these results from our later discussions.

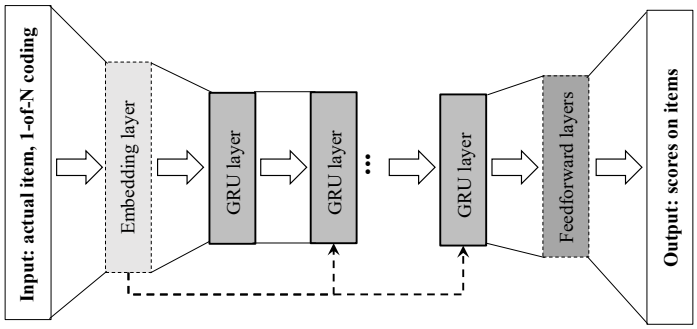



Fig. 1. Architecture of the GRU4REC neural network, adapted from [Hidasi et al. 2016a].

position of $s_x$ within the session.[7] If, for example, the second-to-last item of the current session with a length of 5 is the most recent item also included in the neighbor session $n$, the weight would be $w_n(i, s) = 4/5$. Items from this neighbor can, therefore, potentially obtain a higher score than, e.g., items from neighbor sessions that only include the third from last item of the current session, which are assigned a weight of $3/5$.

- Sequential Filter Session-based kNN (SF-SKNN): This method also uses a modified scoring function, but in a more restrictive way. The basic idea is that given the last event (and related item $s_{|s|}$) of the current session $s$, we only consider items for recommendation that appeared directly after $s_{|s|}$ in the training data at least once.

$$score_{\text{SF-SKNN}}(i, s) = \Sigma_{n \in N_s} sim(s, n) \cdot 1_n(s_{|s|}, i) \tag{7}$$

While the general scoring function is identical to the one of SKNN (Equation 5), we use a different implementation of the indicator function $1_n(s_{|s|}, i)$. Here, 1 is only returned if there exists any past session which contains the sequence $(s_{|s|}, i)$, given $s_{|s|}$ is the item currently viewed in the user's current session $s$. Though the sequence $(s_{|s|}, i)$ can be part of any past session, the item $i$ obviously still has to be a part of the neighbor session $n$ for the indicator function to return 1.

### 3.3 Neural Networks – GRU4REC

Approaches based on Recurrent Neural Networks (RNNs), as discussed in Section 2, represent the most recently explored family of techniques for session-based recommendation problems. Among these methods, GRU4REC is one of the latest deep learning approaches that was specifically designed for session-based recommendation scenarios [Hidasi and Karatzoglou 2017; Hidasi et al. 2016a].

GRU4REC models user sessions with the help of an RNN with Gated Recurrent Units [Cho et al. 2014] in order to predict the probability of the subsequent events (e.g., item clicks) given a session beginning. Figure 1 shows the general architecture of the network, in which the embedding, the feedforward, and additional GRU layers are optional. In fact, the authors of the method found that a single GRU layer of varying width led to the best performance in their experiments.

The input of the network is formed by a single item, which is one-hot encoded in a vector representing the entire item space, and the output is a vector of similar shape that should give a

[7] Note that the weighting function is designed to work independently from the similarity function. We rely on the binary session representation for the similarity calculation without considering the order of the items to ensure computational efficiency.

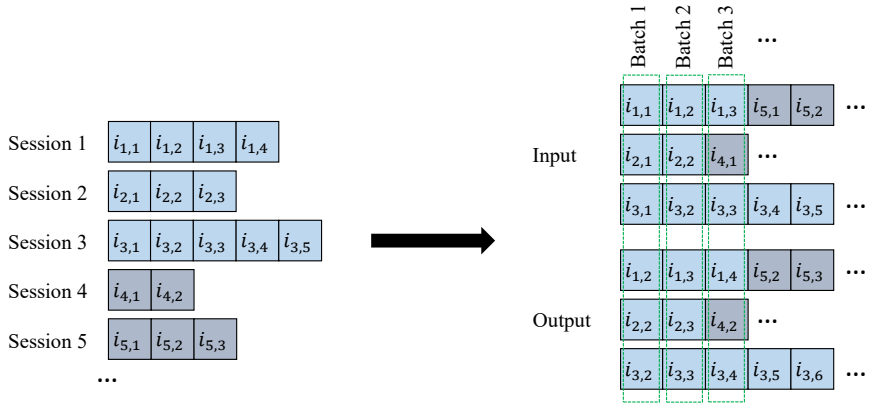



Fig. 2. Illustration of the session-parallel mini-batch scheme of GRU4REC, adapted from [Hidasi et al. 2016a].

ranking distribution for the subsequent item. Inbetween, the standard GRU layer keeps track of a hidden state that encodes the previously occurring items in the same session. Therefore, while training and predicting with the help of this network architecture, the items of a session have to be fed into the network in the correct order and the hidden state of the GRUs has to be reset after a session ends. In terms of the activation functions, the authors found *tanh* and the *sigmoid* function to work best for the GRU and the ranking layer, respectively.

While the usage of RNNs for session-based, or more generally, sequential prediction problems is a natural choice, the particular network architecture, the choice of the loss functions, and the use of session-parallel mini-batches to speed up the training phase are key innovative elements of the approach.

The model can be trained with stochastic gradient descent (SGD) using established optimizations like *ADAM*, *ADADELTA*, *RMSProp*, or *ADAGRAD* [Duchi et al. 2011; Kingma and Ba 2014; Zeiler 2012]. As common in practice when optimizing deep neural networks, Hidasi et al. train the network in batches. To ensure that the items or events are fed into the network in the correct order, they propose the *session-parallel mini-batch* training scheme, which is illustrated in Figure 2. In the training process, each part of a batch belongs to a specific session in the training data and the network records a separate hidden state for each position. Whenever a session at a position in the batch ends, the corresponding hidden state is reset and the next batch update includes the first event of a new session at that position.

As usual, a number of hyper-parameters can be tuned, including, the learning rate, the layer sizes, a momentum factor, and a drop-out factor to stabilize the network. The choice of the loss function is another key to the quality of the recommendations of GRU4REC. The following loss functions were designed or applied by the authors. In particular the latest function (*MAX*) proposed by [Hidasi and Karatzoglou 2017] led to a significant performance improvement over the previous ones.

- *BPR*: Bayesian Personalized Ranking (BPR), as discussed above, uses a pairwise ranking loss function for the task of creating top-n recommendations. In GRU4REC, a generalized version of this function is applied using the following formula:

$$L_s(\hat{r}_{s,i}, S_N) = -\frac{1}{|S_N|} \cdot \sum_{j \in S_N} log(\sigma(\hat{r}_{s,i} - \hat{r}_{s,j})) \tag{8}$$

In the loss function, the predicted rating $\hat{r}_{s,i}$ for the actual next item $i$ given the current session $s$ is compared to a set of negative samples $S_N$ with the goal of maximizing the difference between them. Here, the sigmoid and logarithm functions are applied to represent the proportion between the ranking of the negative and the positive example.

- *TOP1*: This loss function was introduced by the authors of GRU4REC and can be seen as a regularized approximation of the relative rank of a positive sample $\hat{r}_{s,i}$ and the negative samples $S_N$:

$$L_s(\hat{r}_{s,i}, S_N) = \frac{1}{|S_N|} \cdot \sum_{j \in S_N} \sigma(\hat{r}_{s,j} - \hat{r}_{s,i}) + \sigma(\hat{r}_{s,j}^2) \tag{9}$$

Here, the proportion is approximated with the sigmoid function, and the regularization term $\sigma(\hat{r}_{s,j}^2)$ is added so that the score of the negative samples is directed to zero.

- *MAX*: In continuation of their work, the authors proposed a generic extension to these two loss functions, where $L_s$ stands for a loss function like *BPR* or *TOP1* defined above:

$$L_{max}(\hat{r}_{s,i}, S_N) = L_s(\hat{r}_{s,i}, \{\max_{j \in S_N} \hat{r}_{s,j}\}) \tag{10}$$

Instead of using a sum of differences between the positive item's rating $\hat{r}_{s,i}$ and the negative samples $S_N$, only the highest rated negative sample $\max_{j \in S_N} \hat{r}_{s,j}$ from $S_N$ is used to calculate the loss. As this function has to be differentiable for SGD training, $\max_{j \in S_N}$ is approximated with the *softmax* function. The resulting functions $BPR_{max}$ and $TOP1_{max}$ showed superior performance when compared to the *BPR* and *TOP1* functions [Hidasi and Karatzoglou 2017].

In our experiments, we used the GRU4REC (v2.0) implementation that the authors shared online. The code is regularly maintained by the authors and includes the implementation of the GRU4REC method, the code of their baseline algorithms, as well as the code for the evaluation procedure proposed in [Hidasi et al. 2016a].

### 3.4 Factorization-based Methods

As described in Section 2, a number of (hybrid) factorization-based methods were proposed in recent years for *sequential* recommendation problems. We include three existing methods from the literature in our experiments, Factorized Personalized Markov Chains (FPMC) proposed by [Rendle et al. 2010], FISM by [Kabbur et al. 2013], and FOSSIL by [He and McAuley 2016]. Generally, these methods aim at predicting the next actions of users, but were not designed for session-based recommendation scenarios with anonymous users. We therefore describe for each method how we applied them to our problem setting. In addition, we propose a novel factorization method called Session-based Matrix Factorization (SMF), which relies on the $BPR_{max}$ and $TOP1_{max}$ loss functions as described above.

*3.4.1 Factorized Personalized Markov Chains (FPMC).* The FPMC method was designed for the specific problem of next-basket recommendation. The problem consists of predicting the contents of the next basket of a user, given his or her history of past shopping baskets. By limiting the basket size to one item and by considering the current session as the history of baskets, the method can be directly applied for session-based recommendation problems.

Technically, FPMC combines MC and traditional user-item matrix factorization in a three dimensional tensor factorization approach. As illustrated in Figure 3, the third dimension captures the transition probabilities from one item to another.

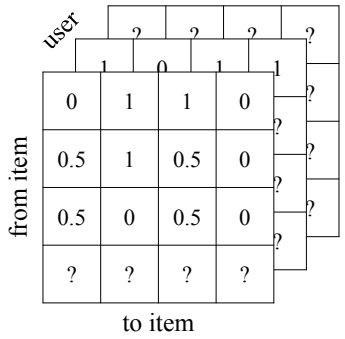



Fig. 3. Personalized transition cube, adapted from [Rendle et al. 2009].

Internally, a special form of the Canonical Tensor Decomposition is used to factor the cube into latent matrices, which can then be used to predict a ranking in the following way:

$$\hat{r}_{u,l,i} = \langle v_u^{U,I}, v_i^{I,U} \rangle + \langle v_i^{I,L}, v_l^{L,I} \rangle + \langle v_u^{U,L}, v_l^{L,U} \rangle \tag{11}$$

where $\hat{r}_{u,l,i}$ is a score for item $i$ with the preferences of user $u$ when he or she previously examined item $l$. The three-dimensional decomposition results in six latent matrices $v^{X,Y}$ representing the latent factors for dimension $X$ regarding dimension $Y$, e.g., $v^{U,L}$ are the user latent factors in terms of the previously examined item and $v^{I,L}$ the item latent factors regarding the previously examined item. Accordingly, $v_u^{U,L}$ for example represents the factors for a single user $u$ and $v_i^{I,L}$ the factors for item $i$, which are combined with the regular dot product ($\langle a, b \rangle$) to calculate the ranking $\hat{r}_{u,l,i}$. Those latent factors are learned using SGD with the pairwise ranking loss function BPR.

In our problem setting, where we have no long-term user histories, each session in the training data corresponds to a user. Once the model is trained, each new session therefore represents a user cold-start situation. To apply the model to our setting, we estimate the session latent vectors as the average of the latent factors of the individual items in the session. This approach was adopted also by [Hidasi et al. 2016a] to apply BPR-MF to session-based recommendation scenarios.

*3.4.2 Factored Item Similarity Models (FISM).* This method is based on an item-item factorization, which has the advantage of being directly applicable to our session-based cold-start scenario, where no explicit user representation can be learned. However, FISM does not incorporate sequential item-to-item transitions like FPMC does. Equation 12 shows the prediction function which [Kabbur et al. 2013] trained using SGD to predict ratings, e.g., for the movie domain.

$$\hat{r}_{u,i} = b_u + b_i + (n_u^+)^{-\alpha} \sum_{j \in R_u^+} p_j q_i^T \tag{12}$$

Technically, for user $u$ and item $i$, a score $\hat{r}_{u,i}$ is calculated as the sum of latent vector products $p_j q_i^T$ between item $i$ and the items $R_u^+$ already rated by the user $u$. In our scenario, $R_u^+$ corresponds to the previously inspected items in a session. The terms $b_u$ and $b_i$ are bias terms and $n_u^+$ specifies the number of ratings by user $u$, which is combined with a parameter $\alpha$ to normalize the sum of vector products to a certain degree. Instead of using the *RMSE* as an error metric, we use *BPR*'s pairwise loss function when optimizing the top-n recommendations for the given implicit feedback scenario.

*3.4.3 Factorized Sequential Prediction with Item Similarity Models (FOSSIL).* In this approach, FISM is combined with factorized Markov chains to incorporate sequential information into the model. The model can be described as shown in Equation 13 (from [He and McAuley 2016]):

$$\hat{r}_{u,l,i} = \underbrace{\sum_{j \in R_u^+ \setminus \{i\}} p_j q_i^T}_{\text{long-term preferences}} + \overbrace{(w + w_u)}^{\text{personalized weighting}} \cdot \underbrace{n_l m_i^T}_{\text{sequential dynamics}} \tag{13}$$

Again, $\hat{r}_{u,l,i}$ represents a rating for item $i$ given a user $u$ and his or her previously inspected item $l$. The first term represents the long-term user preferences and corresponds to the FISM model in Equation 12. Using a weighted sum with a global factor $w$ and a personalized factor $w_u$, the model is extended by a factorized Markov chain to capture the sequential dynamics. In the last term of Equation 13, a latent vector $n_l$ for item $l$ is multiplied with a latent vector $m_i$ for item $i$ to factor in the user-independent probability of item $l$ being followed by item $i$.

In our scenario, again, the sessions represent the users, $R_u$ corresponds to the current session and *BPR* is used as the loss function to rank suitable items over negative examples.

*3.4.4 Session-based Matrix Factorization (SMF).* Finally, SMF is a novel factorization-based model that we designed for the specific task of session-based recommendation. Similar to FOSSIL it combines factorized Markov chains with classic matrix factorization. In addition, our method considers the cold-start situation of session-based recommendation scenarios as follows.

In contrast to the traditional factorization-based prediction model $r_{u,i} = p_u q_i^T$, in the SMF method, we replace the latent user vector $p_u$ with a session preference vector $s_e$, which is computed as an embedding of the current session $s$:

$$s_e = M_{ST} \cdot s^T \tag{14}$$

Here, the session $s$ is as a binary vector similar to the representation in SKNN (see Section 3.2.2) and $M_{ST}$ is a transformation matrix of size $|I| \cdot |u_s|$, which reduces the size of the binary session vector (number of unique items $|I|$) to a specific latent vector size $|s_e|$.

Based on the embedded session representation $s_e$, the prediction function is defined as shown in Equation 15.

$$\hat{r}_{s,l,i} = w_i \cdot ( \underbrace{s_e q_i^T + b_{1,i}}_{\text{session preferences}} ) + (1 - w_i) \cdot ( \overbrace{n_l m_i^T + b_{2,i}}^{\text{sequential dynamics}} ) \tag{15}$$

The score $\hat{r}_{s,l,i}$ for a session $s$ with the most recent item $l$ and an item $i$ is computed as a weighted combination of session preferences and sequential dynamics. Here, the session preferences correspond to the long-term user preferences in the traditional matrix factorization model, i.e., the embedded session latent vector $s_e$ for the current session $s$ is multiplied with an item latent vector $q_i$ for item $i$ to compute a relevance score $i$ regarding $s$. The sequential dynamics are captured exactly as in Equation 13 for FOSSIL using latent representations for the currently inspected item $l$ and item $i$. Both partial scores are adjusted with a separate bias term $b_{x,i}$ and combined in a weighted sum with the factor $w_i$ dependent on item $i$.

To train this model, we incorporated some of the concepts from GRU4REC (see Section 3.3). Specifically, we adopted *ADAGRAD* for SGD-based optimization, and used $BPR_{max}$ and $TOP1_{max}$ as loss functions. Furthermore, we integrated two additional concepts (and corresponding hyper-parameters) in the training phase to avoid model over-fitting: a session drop-out factor and a skip-rate. For a drop-out factor of 0.1, for example, each positive entry of the binary session input

vector is set to 0 with a probability of 10%. The skip-rate, in contrast, describes how often not the immediate next item in the log data should be used as a positive sample in the training process, but the subsequent one. A skip rate of 0.1 therefore means that in 10 % of the cases the immediate next item is skipped.

## 4 EXPERIMENT SETUP

In this section, we describe the details of our algorithm comparison in terms of to the used evaluation protocol, the performance measures, and the evaluation datasets. All source code and pointers to the public datasets are provided online to ensure reproducibility of our research.[8]

### 4.1 Evaluation Protocol and Performance Measures

The general computational task in session-based recommendation problems is to generate a ranked list of objects that in some form "matches" a given session beginning. What represents a good match, depends on the specific application scenario. It could be a set of alternative shopping items in an e-commerce scenario or a continuation of given music listening session.

In offline evaluations for session-based recommendations, researchers often abstract from the underlying purpose of the system [Jannach and Adomavicius 2016], e.g., if the recommender should help discover something new or find alternatives to a currently inspected item. Instead, the recorded user sessions are typically considered as a "gold standard" for the evaluation. To measure the performance of an algorithm, researchers resort to assessing the capability of an algorithm to predict the withheld entries of a session.

Different approaches are found in the literature to withhold certain entries of a session. In some works, only the last element is hidden [Bonnin and Jannach 2014; Hariri et al. 2012], some propose to "reveal" the first $n$ elements of a session [Jannach et al. 2015a], while others, finally, evaluate their approaches by iteratively revealing one entry after the other [Hidasi et al. 2016b]. We employed the latter iterative revealing scheme in our experiments as it (i) conceptually includes both of the other techniques and (ii) reflects the user journey throughout a session in the best way.

*Selection of the Target Item and Accuracy Measures.* We measured prediction accuracy in two ways and correspondingly report the results in separate tables.

- First, to establish comparability with existing research, we use an evaluation scheme in which the task is to predict the *immediate next item* given the first $n$ elements. For each session, we iteratively increment $n$, measure the hit rate (HR) and the Mean Reciprocal Rank (MRR), and finally determine the average HR and MRR for all sessions for the different list lengths, as done by [Hidasi et al. 2016b].
- Second, instead of focusing only on the next item, we made a measurement where we considered *all subsequent* elements for the given session beginning, because all of them might be relevant to the user. In this scheme, we used the standard information retrieval measures precision and recall at defined list lengths. The number of given elements of the session is also iteratively incremented as in the previously described evaluation scheme.

*Sessionization strategies.* Different strategies exist in the literature to split the user activity logs into sessions. In some of the public datasets used in our evaluation, the activity logs were already split up into sessions, i.e., each log entry was assigned a unique session ID (*RSC15*, *Zalando*). For other datasets (*RETAILR*, *NOWPLAYING*, *30MUSIC*, *CLEF*), we used a common heuristic-based approach and considered a session as over after a defined user idle time, e.g., 30 minutes of user inactivity [Cooley et al. 1999]. For the TMALL dataset, where the timetamps for the recorded events

[8] https://www.dropbox.com/sh/dbzmtq4zhzbj5o9/AACldzQWbw-igKjcPTBI6ZPAa?dl=0

were only available at the granularity of a day, we considered all events of one day as belonging to one session. Finally, for the two playlists dataset (*AOTM*, *8TRACKS*), we considered all elements of a playlist to be part of a session.

*Training and Test Splits, Repeated Subsampling.* [Hidasi et al. 2016b] used one single training-test split. In the case of an e-commerce dataset, the data was split in a way that the sessions of all six months except those of the very last day of the entire dataset were placed in the training set. The last day was used for testing. We report the results of applying this evaluation scheme to ensure comparability, e.g., with respect to the results obtained for the e-commerce dataset that was used in their experiments.

Since such single-split setups have their limitations, we focus our discussion on the results that were obtained when applying a *sliding-window* protocol, where we split the data into 5 slices of equal size in days. For most e-commerce data, for example, we used the data of about one month for training and the subsequent data (e.g., of one day) for testing (see Section 4.2 for the dataset specific configurations). This allows us to make multiple measurements with different test sets. We then evaluate the performance for each of these data samples and report the average of the performance results for all slices. This latter protocol helps us reduce the danger that the observed outcomes are the results of one particular train-test configuration.[9]

For the playlist datasets *8TRACKS* and *AOTM* no timestamp information is available. For these datasets we therefore applied a standard cross-validation procedure, where elements are randomly assigned to the training and test sets. We did not use such a time-agnostic data splitting procedure for the e-commerce and news datasets for different reasons. First, as the results will show, there are strong temporal effects that should be considered in the recommendation process. Second, in these domains, the set of items is not static and in particular in the news domain new items appear constantly. Randomly splitting the sessions would then potentially result in the effect that future interactions with not-yet-existing items would be considered in the training phase.

*Additional Quality Factors.* Since accuracy is not the only relevant quality factor in practice, we made the following additional measurements, as was done by [Jannach and Ludewig 2017].

- *Coverage:* We report how many different items ever appear in the top-$k$ recommendations. This measure represents a form of catalog coverage, which is sometimes referred to as *aggregate diversity* [Adomavicius and Kwon 2012].
- *Popularity bias:* High accuracy values can, depending on the measurement method, correlate with the tendency of an algorithm to recommend mostly popular items [Jannach et al. 2015b]. To assess the popularity tendencies of the tested algorithms, we report the *average popularity score* for the elements of the top-$k$ recommendations of each algorithm. This average score is the mean of the *individual popularity scores* of each recommended item. We compute these scores based on the training set by counting how often each item appears in one of the training sessions and by then applying min-max normalization to obtain a score between 0 and 1.
- *Cold start:* Some methods might only be effective when a significant amount of training data is available. We, therefore, report the results of measurements where we artificially removed parts of the (older) training data to simulate such situations.

[9] To ensure that the smaller size of those splits does not negatively affect the performance of the model-based approaches, we tested the single-split configurations as well on all datasets. The obtained results are mostly in line with those obtained with the sliding-window protocol and shown in Appendix D.

- *Scalability:* Training modern machine learning methods can be computationally challenging, and obtaining good results may furthermore require extensive parameter tuning. We, therefore, report the times that the algorithms needed to train the models and to make predictions at runtime. In addition, we report the memory requirements of the algorithms.

By reporting quality factors *coverage* and *popularity bias* our aim is to emphasize that different recommendation strategies can lead to quite different recommendations, even if they are similar in terms of the prediction accuracy, see also [Jannach et al. 2015b]. Such multi-metric evaluation approaches should also help practitioners to better understand the potential side effects of the recommenders, e.g., reduced average sales diversity and additionally increased sales of top-sellers [Lee and Hosanagar 2014]. It remains however difficult to aggregate the individual performance factors into one single score, as the relative importance of the factors can depend not only on the application domain, but also on the specific business model of the provider.

*Parameter Optimization.* Some of the algorithms that we tested require extensive (hyper-)parameter tuning including SMF and GRU4REC. Thus, we systematically optimized the parameters for those algorithms for each dataset. Due to the computational complexity of the methods, we restricted the layer size for GRU4REC as well as the number of latent factors for SMF to 100 and used a randomized search method with 100 iterations for the remaining parameters as described by [Hidasi and Karatzoglou 2017]. In each iteration, the learning rate, the drop-out factor, the momentum, and the loss function were determined in a randomized process to find the maximum hit rate for a list length of 20. All optimizations were performed on special validation splits, which were created by splitting a training set into a validation training and test set. For the simpler S-KNN-based approaches, we used the same validation sets to manually adjust the number of neighbors and samples when applying cosine similarity as the distance measure (except for V-SKNN). The final parameters for each method and dataset are provided in Appendix A.

### 4.2 Datasets

We made measurements for datasets from three different domains: e-commerce, music, and news.

*E-Commerce Datasets.* We used the following four e-commerce datasets.

- *RSC15.* This is one of the datasets that was used in [Hidasi et al. 2016a] and their later works. It was published in the context of the ACM RecSys 2015 Challenge and contains recorded click sequences (item views, purchases) for a period of six months. We use the label *RSC15-S* to denote the dataset and measurement where only one single train-test split is used. For RSC15, each split consists of 30 days of training and 1 day of test data.
- *TMALL.* This dataset was published in the context of the TMall competition and contains interaction logs of the tmall.com website for one year. For TMALL, each split consists of 90 days of training and 1 day of test data.
- *RETAILR.* The e-commerce personalization company *retailrocket* published this dataset covering six month of user browsing activities, also in the context of a competition. For RETAILR, each split consists of 25 days of training and 2 days of test data.
- *ZALANDO.* The final dataset is non-public and was shared with us by the fashion retailer Zalando. It contains user logs of their shopping platform for a period of one year. In our evaluation, we only considered the item *view* events as was done for the other e-commerce datasets. For ZALANDO, each split consists of 90 days of training and 1 day of test data.

Table 1 shows an overview of the characteristics of the e-commerce datasets. Except for the *RSC15-S* dataset, which we include to make our evaluation comparable with previous works [Hidasi

Table 1. Characteristics of the e-commerce datasets. The values are averaged over all five non-overlapping splits for each dataset, except for *RSC15-S*, where we only use one train-test split.

| Dataset | RSC15-S | RSC15 | TMALL | RETAILR | ZALANDO |
|---|---|---|---|---|---|
| Actions | 31.71M | 5.43M | 13.42M | 212,182 | 4.54M |
| Sessions | 7.98M | 1.38M | 1.77M | 59,962 | 365,126 |
| Items | 37,483 | 28,582 | 425,348 | 31,968 | 189,328 |
| Timespan in Days | 182 | 31 | 91 | 27 | 91 |
| Actions per Session | 3.97 | 3.95 | 7.56 | 3.54 | 12.43 |
| Unique Items per Session | 3.17 | 3.17 | 5.56 | 2.56 | 8.39 |
| Actions per Day | 174,222 | 175,063 | 149,096 | 7,858 | 50,410 |
| Sessions per Day | 43,854 | 44,358 | 19,719 | 2,220 | 4056 |

and Karatzoglou 2017; Jannach and Ludewig 2017], we report the average values after creating five data splits as described above.

*Media Datasets: Music and News.* As in [Jannach and Ludewig 2017], we use the music domain as an alternative area to evaluate session-based recommendation algorithms, because music is commonly consumed within listening sessions in sequential order. We use the same datasets that were used in [Jannach and Ludewig 2017], which consist of two sets of *listening logs* and two datasets of user-created *playlists*. In addition, we made measurements using a dataset from the news recommendation domain.

We in particular consider the news domain because it has certain distinct characteristics [Karimi et al. 2018]. First, constantly new items become available for recommendation [Das et al. 2007; Liu et al. 2010]. At the same time, items can also quickly become outdated. Second, previous research indicates that short-term popularity trends can be important for the success of a recommender [Ludmann 2017]. The experiments based on this dataset should therefore provide an indicator if the general insights obtained from other domains generalize to a domain with very specific characteristics.

- *8TRACKS and AOTM:* These dataset include playlists created by music enthusiasts. The *AOTM* dataset was collected from the Art-of-the-Mix platform and is publicly available [McFee and Lanckriet 2012]. The non-public *8TRACKS)* dataset was shared with us by the 8tracks.com music platform. For all music datasets, each split consists of 90 days of training and 5 days of test data.
- *30MUSIC and NOWPLAYING:* The *30MUSIC* dataset contains listening histories of the last.fm music platform and was published by [Turrin et al. 2015]. The *NOWPLAYING* dataset was created from music-related tweets, where users posted which tracks they were currently listening [Zangerle et al. 2014].
- *CLEF:* The dataset was made available to participants of the 2017 CLEF NewsREEL challenge.[10] It consists of a stream of user actions (e.g., article reads) and article publication events, which were collected by the company *plista* for several publishers. In our evaluation we only considered the article read events. We used the data of the publisher with the largest amount of recorded interactions (the popular German sports news portal Sport1[11]). For CLEF, each split consists of 5 days of training and 1 days of test data.

The statistics for the datasets from the media (music and news) domain are given in Table 2.

[10] http://www.clef-newsreel.org/
[11] https://www.sport1.de/

Table 2. Characteristics of the music and news datasets. The values are again averaged over all five non-overlapping splits.

| | 8TRACKS | 30MUSIC | AOTM | NOWPLAYING | CLEF |
|---|---|---|---|---|---|
| Actions | 1.50M | 638,933 | 306,830 | 271,177 | 5.54M |
| Sessions | 132,453 | 37,333 | 21,888 | 27,005 | 1.64M |
| Items | 376,422 | 210,633 | 91,166 | 75,169 | 742 |
| Timespan in Days | 95 | 95 | 95 | 95 | 6 |
| Actions per Session | 11.32 | 17.11 | 14.02 | 10.04 | 3.37 |
| Items per Session | 11.31 | 14.47 | 14.01 | 9.38 | 3.17 |
| Actions per Day | 16,663 | 7,099 | 3,409 | 3,013 | 923,414 |
| Sessions per Day | 1,472 | 415 | 243 | 300 | 274,074 |

## 5 RESULTS

### 5.1 E-Commerce Datasets

Table 3 shows the MRR and Hit Rate results at a recommendation list length 20 for the four tested e-commerce datasets. In addition, we report the results when applying the standard measures precision and recall when considering *all* hidden elements in the rest of the session as described above (see Table 4). Finally, we also report coverage and popularity statistics for each algorithm.

*5.1.1 Accuracy Measures.* The results when the task is to predict the *immediate next* element in a session (as done in [Hidasi and Karatzoglou 2017; Jannach and Ludewig 2017]) are shown in Tables 3a to 3e. The following observations in terms of the hit rate and the MRR can be made.[12]

- The lowest accuracy values are almost consistently achieved across all datasets by the family of Factorized Markov Chain approaches (FISM, FPMC and FOSSIL) and the session-aware BPR-MF variant. BPR-MF in fact often exhibits the best performance among these methods even though it was not designed for sequential recommendation problems. In two cases, however, the session-based BPR-MF variant led to very competitive results when the measurement was taken at a recommendation list length of 1, although at the potential price of a high popularity bias and low coverage. Apart from this phenomenon, our results indicate that the methods that were designed under the assumption of longer-term and richer user profiles are often not particularly well suited for the specifics of session-based recommendation problems.
- The simple pairwise association methods (AR and SR) mostly occupy the middle places in our comparison. In most cases, it is preferable to consider the available sequentiality information (SR). Only for the *TMALL* dataset, where the transactions of an entire day are considered as a session[13], and for *RETAILR*, the sequence-agnostic AR method is slightly better in terms of the hit rate. In terms of the overall ranking, the trivial SR method is, to some surprise, among the *top-performing* methods for two of the datasets in terms of the MRR, with good results also for the hit rate. The MC method finally, is usually placed somewhere in the middle of the ranking. Similar to the SR method, it is very strong in terms of the MRR for two of the datasets.
- The performance of the newly-proposed SMF method is very strong for the *RSC15* and *RSC15-S* dataset and in the middle ranges for the other datasets. The SMF method consistently outperforms the factorization-based methods from the literature, apparently due to the embedding of the current user session.

[12]We provide additional results that were obtained for measurements taken at multiple list lengths in Appendix B.
[13]In the dataset, timestamps are only available at the granularity of days.

Table 3. Hit rate (HR), Mean reciprocal rank (MRR), catalog coverage (COV), and the average popularity (POP) for a list length of 20 obtained for the e-commerce datasets. The table rows are ordered by MRR@20. The best values are highlighted in each column and, in case of accuracy measures, marked with a star when the difference w.r.t. to the second-best performing method was statistically significant.

(a) RSC15

| Metrics | MRR@20 | HR@20 | COV@20 | POP@20 |
|---|---|---|---|---|
| GRU4REC | **0.308** | ***0.683** | 0.504 | 0.054 |
| SR | 0.304 | 0.653 | 0.668 | 0.072 |
| SMF | 0.302 | 0.666 | 0.565 | 0.055 |
| MC | 0.300 | 0.642 | 0.645 | 0.070 |
| AR | 0.289 | 0.636 | 0.630 | 0.093 |
| V-SKNN | 0.283 | 0.653 | 0.619 | 0.079 |
| S-SKNN | 0.272 | 0.602 | 0.655 | 0.072 |
| SF-SKNN | 0.270 | 0.589 | 0.619 | 0.066 |
| S-KNN | 0.266 | 0.621 | 0.634 | 0.073 |
| IKNN | 0.208 | 0.486 | 0.755 | 0.041 |
| FPMC | 0.201 | 0.363 | **0.975** | 0.055 |
| BPR-MF | 0.176 | 0.235 | 0.911 | 0.088 |
| FISM | 0.115 | 0.162 | 0.974 | **0.008** |
| FOSSIL | 0.062 | 0.190 | 0.917 | 0.048 |

(b) TMALL

| Metrics | MRR@20 | HR@20 | COV@20 | POP@20 |
|---|---|---|---|---|
| S-SKNN | ***0.185** | 0.387 | 0.467 | 0.025 |
| S-KNN | 0.182 | ***0.404** | 0.381 | 0.026 |
| V-SKNN | 0.179 | 0.373 | 0.464 | 0.024 |
| BPR-MF | 0.159 | 0.204 | 0.723 | 0.057 |
| SF-SKNN | 0.136 | 0.216 | 0.436 | 0.018 |
| GRU4REC | 0.129 | 0.277 | 0.151 | 0.035 |
| AR | 0.129 | 0.262 | 0.509 | 0.021 |
| SR | 0.128 | 0.242 | 0.569 | 0.021 |
| SMF | 0.121 | 0.261 | 0.261 | 0.036 |
| MC | 0.116 | 0.200 | 0.498 | 0.019 |
| FPMC | 0.101 | 0.119 | **0.880** | 0.005 |
| IKNN | 0.051 | 0.150 | 0.728 | 0.007 |
| FISM | 0.024 | 0.037 | 0.752 | **0.003** |
| FOSSIL | 0.001 | 0.004 | 0.598 | 0.016 |

(c) RETAILR

| Metrics | MRR@20 | HR@20 | COV@20 | POP@20 |
|---|---|---|---|---|
| S-SKNN | ***0.345** | ***0.591** | 0.596 | 0.056 |
| V-SKNN | 0.338 | 0.573 | 0.575 | 0.060 |
| S-KNN | 0.337 | 0.583 | 0.566 | 0.058 |
| BPR-MF | 0.303 | 0.357 | 0.824 | 0.060 |
| FPMC | 0.273 | 0.320 | **0.929** | 0.022 |
| SF-SKNN | 0.260 | 0.358 | 0.403 | 0.035 |
| SR | 0.245 | 0.419 | 0.524 | 0.042 |
| GRU4REC | 0.243 | 0.480 | 0.602 | 0.060 |
| AR | 0.241 | 0.439 | 0.544 | 0.053 |
| MC | 0.230 | 0.359 | 0.411 | 0.035 |
| SMF | 0.225 | 0.459 | 0.449 | 0.085 |
| IKNN | 0.107 | 0.240 | 0.584 | 0.033 |
| FISM | 0.075 | 0.132 | 0.848 | **0.018** |
| FOSSIL | 0.022 | 0.058 | 0.753 | 0.127 |

(d) ZALANDO

| Metrics | MRR@20 | HR@20 | COV@20 | POP@20 |
|---|---|---|---|---|
| SR | **0.304** | 0.483 | 0.586 | 0.061 |
| MC | 0.303 | 0.455 | 0.513 | 0.060 |
| IKNN | 0.275 | 0.405 | 0.714 | 0.037 |
| GRU4REC | 0.267 | 0.468 | 0.304 | 0.101 |
| SMF | 0.267 | 0.447 | 0.362 | 0.107 |
| AR | 0.258 | 0.467 | 0.467 | 0.089 |
| SF-SKNN | 0.249 | 0.438 | 0.432 | 0.057 |
| V-SKNN | 0.233 | ***0.521** | 0.432 | 0.096 |
| S-SKNN | 0.219 | 0.499 | 0.435 | 0.087 |
| S-KNN | 0.172 | 0.456 | 0.309 | 0.093 |
| BPR-MF | 0.104 | 0.162 | 0.609 | 0.058 |
| FPMC | 0.051 | 0.075 | **0.812** | 0.021 |
| FISM | 0.004 | 0.011 | 0.624 | **0.020** |
| FOSSIL | 0.002 | 0.005 | 0.671 | 0.034 |

(e) RSC15-S

| Dataset | RSC15 | |
|---|---|---|
| Metric | MRR@20 | HR@20 |
| GRU4REC | **0.312** | **0.719** |
| SMF | 0.309 | 0.713 |
| SR | 0.308 | 0.690 |
| V-SKNN | 0.274 | 0.675 |
| SKNN | 0.250 | 0.641 |

Table 4. Precision (P@20) and Recall (R@20) for the e-commerce datasets. The rows are ordered by the P@20 values for the *TMALL* data set, which led to a relatively consistent ranking of the algorithms.

| Dataset | RSC15 | | TMALL | | ROCKET | | ZALANDO | |
|---|---|---|---|---|---|---|---|---|
| Metric | P@20 | R@20 | P@20 | R@20 | P@20 | R@20 | P@20 | R@20 |
| SKNN | 0.086 | 0.464 | **0.095** | **0.312** | **0.056** | **0.478** | 0.074 | 0.202 |
| V-SKNN | **0.092** | 0.494 | 0.088 | 0.291 | 0.055 | 0.462 | **0.076** | **0.207** |
| SMF | 0.092 | **0.501** | 0.068 | 0.230 | 0.047 | 0.397 | 0.062 | 0.175 |
| GRU4REC | 0.085 | 0.470 | 0.068 | 0.233 | 0.046 | 0.400 | 0.065 | 0.181 |
| SR | 0.089 | 0.488 | 0.052 | 0.193 | 0.038 | 0.342 | 0.060 | 0.174 |

- GRU4REC is consistently among the top five algorithms in this comparison in terms of the hit rate and exhibits competitive performance results also with respect to the MRR. The method is outperforming all other methods on the *RSC15(-S)* datasets in terms of the hitrate and is competitive w.r.t. the MRR, where the differences between the top-performing methods are tiny. On the other datasets, the accuracy results of GRU4REC are, however, often significantly lower than those of the best-performing methods.[14]
- For each of the datasets, one of the proposed neighborhood-based methods was usually the winner in terms of the hit rate and the MRR (except for *RSC15(-S)* and the MRR on *ZALANDO*). Using one of the variants that considers sequentiality information is usually favorable, except for the case of the *TMALL* dataset. The most consistent performance of the neighborhood-based methods is achieved with the V-SKNN method which uses a specific sequence-aware similarity measure that gives more weight to the most recent interactions. Generally, the results suggest that there is even room for further improvement in the context of the neighborhood-based methods. In the experiments reported in this work, we could, for example, observe that using a slightly different similarity measure already led to substantial performance improvements for some of the datasets.

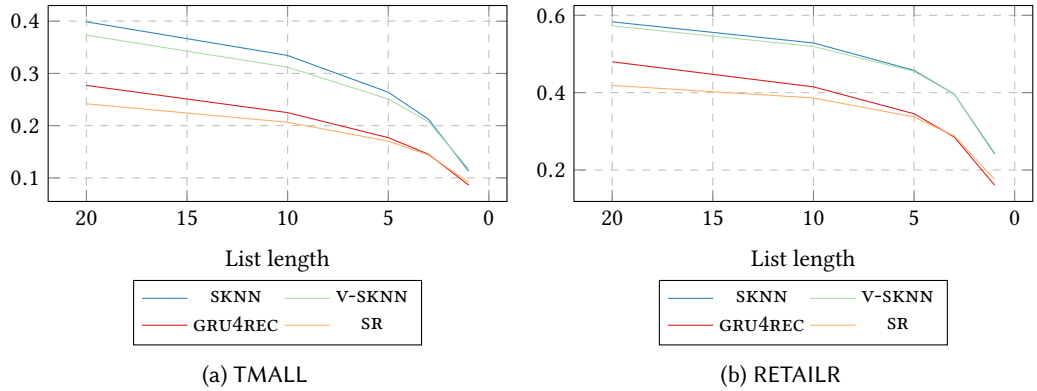
*Precision and Recall for the Remaining Session.* The ranking of the *best-performing algorithms* when evaluating *all* subsequent elements of a session (not only the immediate next click) and measuring precision and recall is given in Table 4. We report the detailed results for all algorithms for all measurements in the appendix.

The obtained results are mostly in line with the previously reported observations. The best performance is achieved by the neighborhood-based methods, with V-SKNN working very well across all datasets. Differently from the previous measurement, GRU4REC shows a lower performance for the *RSC-15* dataset than the other methods. This is probably due to the fact that GRU4REC is optimized to predict the *immediate* next action. Generally, which type of accuracy measurement—focusing on the prediction of the immediate next element or considering the prediction of any item that is relevant in the session as a success—is more appropriate, depends on the application domain. Our results show that the kNN-based methods are successful in both forms, i.e., they are often good at predicting the next element while, at the same time, they many times include *more* items that are relevant for the given session than, e.g., GRU4REC.

*Impact of Different List Lengths.* To see if the recommendation list length at which the measurement is taken has an influence on the algorithm ranking, we varied the length from 20 to 1. Figure 4a and Figure 4b show how the best algorithms perform for the *TMALL* and *RETAILR* datasets when different list lengths are used in the evaluation. The results show that the ranking of the algorithms can in fact be affected by the change of the list length.

Specifically, the differences between the nearest-neighbor methods and the GRU4REC and SR methods becomes gradually smaller for shorter list lengths. This is not too surprising because both GRU4REC and SR focus on the prediction of the immediate next action and often lead to better performance values in terms of the MRR. Since the particular evaluation protocol here also only focuses on the correct prediction of the next item, the effect might however be overemphasized due to the specific measurement method. An interesting observation is that at list length 1, BPR-MF and to some extent the FPMC method lead to the best results for some e-commerce datasets. In the case of BPR-MF, this however comes at the price of a high popularity tendency of the algorithm and a comparably low coverage (see Table 15 in Appendix B).

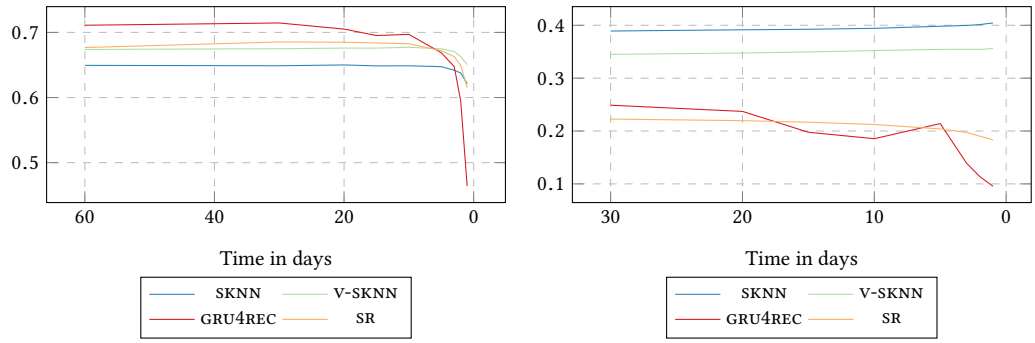
[14]We applied the Wilcoxon signed-rank test ($\alpha = 0.05$) to determine the significance of differences between the two best performing approaches for each dataset.




(a) TMALL (b) RETAILR

Fig. 4. Hit rate (HR) for two of the e-commerce datasets when reducing the recommendation list length from 20 to 1.

*5.1.2 Cold-Start and Sparsity Effects.* Previous experiments on the *RSC15* dataset revealed that discarding major parts of the older data has no strong impact on the prediction accuracy, at least in the e-commerce domain [Jannach and Ludewig 2017]. We therefore made additional experiments to analyze the effects in more depth. Figure 5a and Figure 5b show the results of this simulation for two of the e-commerce datasets.




(a) RSC15-S (b) TMALL

Fig. 5. HR@20 for two e-commerce datasets when artificially reducing the size of the training set from 60 days to 1 day.

The results for the *RSC15-S* (single-split) dataset (Figure 5a) are in line with what was previously reported in [Jannach and Ludewig 2017]. In the e-commerce domain, the user behavior seems to be strongly influenced by recent sales trends, an effect that was also reported in [Jannach et al. 2017b]. Discarding most of the historical data has almost no influence on the resulting hit rates. This behavior is similar for all compared algorithms. Only in the extreme case when only the data of the last few days is considered, the performance of the algorithms degrades. A similar observation can be made for the *TMALL* dataset. Generally, the observations also explain why the recency-based neighborhood sampling approach implemented in the kNN methods does not have a strong negative effect on the accuracy. In fact, focusing on the most recent sessions when looking

for similar neighbors has shown to have a positive effect in [Jannach and Ludewig 2017], when compared to a random neighborhood selection scheme.

*Considering other Types of Events.* In the reported experiments, the models were trained with past item view events and we also predicted the next view event for a given session beginning. This choice was made to make our work comparable with previous research. Additional types of events (e.g., "add-to-wishlist", "add-to-cart") can easily incorporated as positive preference signals into the investigated algorithms. How to weight the different types of signals nd how to interpret signals like "remove-from-wishlist" is an area for future research.

Depending on the domain, also different types of events might be in the focus as well in the prediction phase. In the experiments reported here, we predict item views. In our previous research on the topic [Jannach and Ludewig 2017], we also made experiments in which we focused on the prediction of purchase events. In these experiments, the ranking of the algorithms was similar for the item prediction and the purchase prediction tasks. However, some previous research suggests that view-based collaborative filtering algorithms lead to sometimes quite different recommendations than purchase-based ones and also differ in their effectiveness [Lee and Hosanagar 2014]. In general, the choice of the prediction target should therefore be made with the goal of the recommender in mind, e.g., increase user attention and click-through-rates vs. increasing sales, see, e.g., [Jannach and Hegelich 2009].

*5.1.3 Coverage and Popularity Bias.* The results listed in Table 3a to Table 3d show that in terms of the coverage (or: aggregate diversity), the factorization-based methods consistently lead to the highest values, i.e., they place the largest number of different items into the top-n lists of the users. GRU4REC represents in all datasets, except *RETAILR*, the other extreme and seems to focus its recommendations on a comparably narrow range of items. In particular in the case of the *TMALL* dataset, the coverage of the item space of GRU4REC is as low as 0.15, i.e., the top-20 recommendations for all given sessions in the test set cover only 15 % of the available items. To what extent low coverage is undesired, again depends on the specific application domain.

Not many consistent patterns can be identified with regard to the popularity biases of the different algorithms. BPR-MF, as was previously discussed by [Jannach et al. 2015b], has a comparably strong tendency to focus on generally popular items. Our newly proposed SMF method exhibits a similar tendency across all datasets. The FPMC method usually represents the other end of the spectrum. The tendency of the many of the other algorithms to recommend popular items seems to strongly depend on the dataset characteristics. According to our previous work [Jannach and Ludewig 2017], the basic SKNN method tends to recommend slightly more popular items than GRU4REC. In this new series of measurements, this is, however, not consistently the case across the datasets.

### 5.2 Media Datasets

Table 5, Table 6, and Table 7 show the results for the music and news domains, respectively.

*Accuracy.* The accuracy results generally exhibit similar patterns as the results obtained for the e-commerce datasets. For these datasets, however, the winning strategy more strongly depends on the chosen measure. When the MRR is used as a performance measure, often the trivial baselines SR or AR lead to the best results. In terms of the hitrate, in contrast, usually one of the nearest neighbor methods again performs best.

With respect to the MRR measure also GRU4REC exhibited very competitive performance, except for the *8TRACKS* and *AOTM* datasets, where the highest MRR values were achieved with the AR and the SKNN method. Looking at the playlist datasets (*8TRACKS* and *AOTM*), the comparably good results of the sequence-agnostic AR and SKNN strategy indicate that the ordering of the tracks is

Table 5. Hit rate (HR), Mean reciprocal rank (MRR), catalog coverage (COV), and the average popularity (POP) for a list length of 20 tested on the music datasets. The tables show the top ten algorithms ordered by MRR@20. The best results are highlighted and significant differences are marked with a star.

(a) NOWPLAYING

| Metrics | MRR@20 | HR@20 | COV@20 | POP@20 |
|---|---|---|---|---|
| SR | **0.105** | 0.203 | 0.466 | **0.025** |
| GRU4REC | 0.102 | 0.197 | 0.433 | 0.052 |
| MC | 0.097 | 0.158 | 0.294 | 0.028 |
| SF-SKNN | 0.095 | 0.165 | 0.277 | 0.031 |
| SMF | 0.088 | 0.183 | 0.242 | 0.092 |
| V-SKNN | 0.078 | 0.255 | 0.428 | 0.064 |
| S-SKNN | 0.078 | *__0.262__ | 0.415 | 0.062 |
| AR | 0.071 | 0.208 | 0.453 | 0.051 |
| S-KNN | 0.069 | 0.243 | 0.301 | 0.069 |
| IKNN | 0.057 | 0.182 | **0.580** | 0.029 |

(b) 8TRACKS

| Metrics | MRR@20 | HR@20 | COV@20 | POP@20 |
|---|---|---|---|---|
| AR | *__0.0071__ | 0.0255 | 0.4529 | 0.0912 |
| SMF | 0.0064 | 0.0230 | 0.1527 | 0.0864 |
| SR | 0.0063 | 0.0170 | 0.4967 | 0.0531 |
| SF-SKNN | 0.0063 | 0.0118 | 0.3049 | 0.0362 |
| V-SKNN | 0.0057 | 0.0352 | 0.4080 | 0.1194 |
| S-KNN | 0.0053 | *__0.0375__ | 0.2430 | 0.1079 |
| IKNN | 0.0050 | 0.0176 | **0.6956** | **0.0245** |
| GRU4REC | 0.0050 | 0.0189 | 0.0692 | 0.1222 |
| S-SKNN | 0.0047 | 0.0293 | 0.4509 | 0.0806 |
| MC | 0.0046 | 0.0098 | 0.3496 | 0.0320 |

(c) 30MUSIC

| Metrics | MRR@20 | HR@20 | COV@20 | POP@20 |
|---|---|---|---|---|
| SR | *__0.238__ | 0.332 | 0.389 | 0.023 |
| MC | 0.232 | 0.284 | 0.204 | **0.021** |
| GRU4REC | 0.226 | 0.326 | 0.345 | 0.056 |
| SF-SKNN | 0.208 | 0.286 | 0.185 | 0.022 |
| SMF | 0.178 | 0.284 | 0.151 | 0.105 |
| V-SKNN | 0.110 | 0.382 | 0.317 | 0.054 |
| IKNN | 0.109 | 0.297 | **0.460** | 0.023 |
| S-SKNN | 0.108 | *__0.386__ | 0.293 | 0.052 |
| AR | 0.096 | 0.309 | 0.352 | 0.039 |
| S-KNN | 0.090 | 0.344 | 0.191 | 0.057 |

(d) AOTM

| Metrics | MRR@20 | HR@20 | COV@20 | POP@20 |
|---|---|---|---|---|
| SMF | **0.0111** | 0.0297 | 0.2456 | 0.1998 |
| SF-SKNN | 0.0110 | 0.0144 | 0.3558 | 0.0508 |
| SR | 0.0076 | 0.0195 | 0.5864 | 0.0533 |
| GRU4REC | 0.0071 | 0.0156 | 0.4652 | 0.1151 |
| MC | 0.0063 | 0.0132 | 0.3803 | 0.0497 |
| AR | 0.0059 | 0.0233 | 0.5531 | 0.1049 |
| V-SKNN | 0.0054 | 0.0377 | 0.5362 | 0.1397 |
| S-SKNN | 0.0054 | 0.0397 | 0.5356 | 0.1289 |
| S-KNN | 0.0053 | *__0.0429__ | 0.2802 | 0.1677 |
| IKNN | 0.0049 | 0.0186 | **0.7879** | **0.0472** |

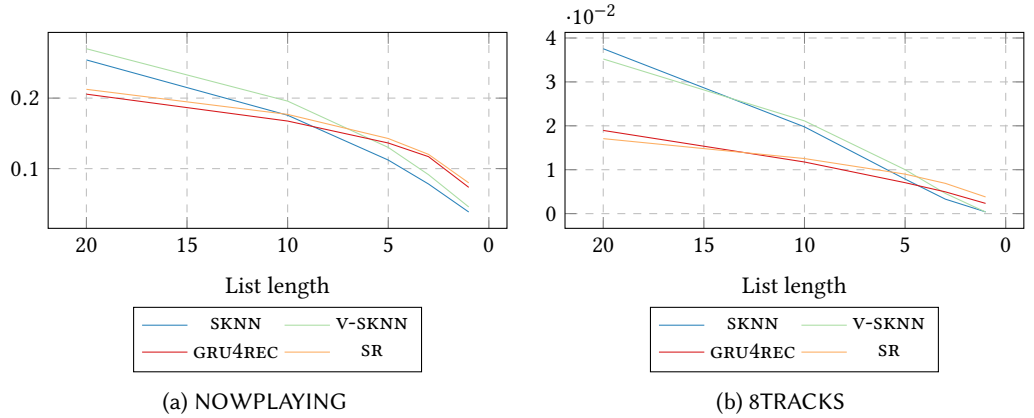
Table 6. Precision (P@20) and Recall (R@20) for the music datasets. The results are ordered by P@20 for *8TRACKS*, which represents the largest music dataset in our evaluation.

| Dataset | LFM | | 8TRACKS | | 30MUSIC | | AOTM | |
|---|---|---|---|---|---|---|---|---|
| Metric | P@20 | R@20 | P@20 | R@20 | P@20 | R@20 | P@20 | R@20 |
| V-SKNN | **0.0717** | **0.1909** | **0.0122** | 0.0308 | **0.1094** | **0.2321** | 0.0133 | 0.0361 |
| SKNN | 0.0680 | 0.1824 | 0.0117 | **0.0313** | 0.1035 | 0.2140 | **0.0155** | **0.0440** |
| SMF | 0.0499 | 0.1453 | 0.0086 | 0.0218 | 0.0746 | 0.1655 | 0.0084 | 0.0259 |
| SR | 0.0501 | 0.1465 | 0.0055 | 0.0140 | 0.0878 | 0.2010 | 0.0053 | 0.0146 |
| GRU4REC | 0.0272 | 0.0810 | 0.0037 | 0.0095 | 0.0404 | 0.0988 | 0.0010 | 0.0027 |

Table 7. Hit rate (HR), Mean reciprocal rank (MRR), Precision (P), Recall (R), item coverage (COV), and average popularity (POP) results for a list length of 20 on the *CLEF* dataset (ordered by MRR@20).

| Metrics | MRR@20 | HR@20 | COV@20 | POP@20 | P@20 | R@20 |
|---|---|---|---|---|---|---|
| SMF | **0.234** | 0.706 | 0.650 | 0.083 | 0.062 | 0.527 |
| V-SKNN | 0.224 | 0.776 | 0.621 | **0.083** | 0.068 | 0.584 |
| SR | 0.223 | 0.672 | **0.655** | 0.093 | 0.058 | 0.502 |
| GRU4REC | 0.220 | 0.568 | 0.174 | 0.094 | **0.072** | **0.626** |
| S-KNN | 0.219 | **0.778** | 0.613 | 0.084 | 0.066 | 0.577 |

not too important for the playlist creators. Among the neighborhood-based methods, V-SKNN was again consistently among the top-performing methods. When looking at the standard precision and recall measurements for the five best-performing approaches in Table 6, we can see that V-SKNN is




(a) NOWPLAYING (b) 8TRACKS

Fig. 6. Hit rate (HR) for two of the music datasets when reducing the result list length from 20 to 1.

the winning strategy across all datasets and that GRU4REC is again less effective for this particular measurement.

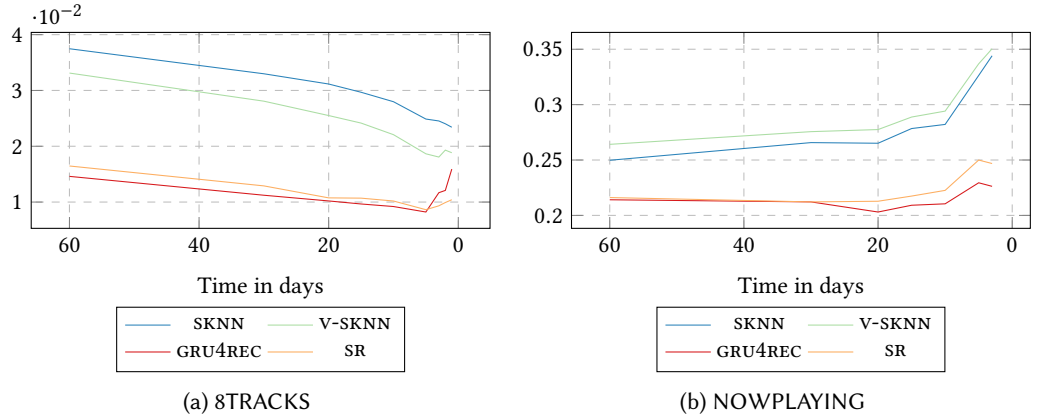
Finally, looking at the *news* domain, the average results shown in Table 7 in general confirm the trends observed for the other datasets. The V-SKNN method is top-performing on almost all measures. GRU4REC also works comparably well on this dataset, especially on the precision and recall measures. Again, however, we can also observe a comparably low level of coverage and a comparably high tendency to recommend popular items. In contrast to all other domains and datasets however, when looking at the results of the individual splits for the *CLEF* dataset, we could observe that those are subject to large fluctuations. Depending on the day that was chosen for testing, the ranking of the algorithms in terms of the accuracy measures changes drastically, which we could not observe for any other dataset.

As mentioned in Section 4, we conducted additional single split experiments to ensure that the reduced amount of training data in the sliding window protocol does not affect the performance of the model-based approaches. The single-split results in Appendix D reveal GRU4REC as the best-performing approach for this particular dataset, which was also the case for two of the individual splits. Thus, even though such large fluctuations did only occur in the news domain, this is an indicator that applying a single-split evaluation protocol can easily lead to "random" and misleading results.

The effects when considering different list lengths for two of the datasets is shown in Figure 6a (*NOWPLAYING*) and Figure 6b (*8TRACKS*). In contrast to the e-commerce datasets, the relative ranking of the algorithms even changes when the list lengths become shorter. For both datasets, the GRU4REC method and the very simple AR and SR methods, respectively, are better in terms of the hit rate when it comes to very short list lengths. Considering the good results for the MRR for these methods (Table 5a and Table 5b), this was expected. Again, the good performance of certain methods can be explained by the fact that these methods are optimized to predict the immediate next item of a given session.

*Cold-Start and Sparsity Effects.* An interesting effect can be observed when older data is discarded to simulate sparsity effects. Figure 7a and Figure 7b show the results for the *8TRACKS* and *NOWPLAYING* datasets, respectively.[15] While for the *8TRACKS* playlist dataset the accuracy values more

[15]The other media datasets did not exhibit any notable particularities.




(a) 8TRACKS (b) NOWPLAYING

Fig. 7. HR@20 for two music datasets when incrementally reducing the size of the training set to 1 day.

or less consistently decrease when older data is discarded, we can observe an *increase* in accuracy for the *NOWPLAYING* dataset. Remember that this dataset is based on the analysis of user posts on Twitter about their current listening behavior. Obtaining the highest accuracy values when only considering the very last days means that this dataset is strongly dominated by short-term popularity trends and that the recommendation of older, non-trending tracks is detrimental to the accuracy results.

*Coverage and Popularity Bias.* In terms of coverage (see Table 5), the findings for datasets from the media domain are also mostly in line with those for the e-commerce datasets. The ranking of the algorithms varies largely across the datasets. The differences are, however, often less pronounced. Regarding the popularity tendency of the algorithms, methods that are based on pairwise sequences (SR and MC) in most cases lead to the recommendation of lesser known items, while nearest-neighbor-based techniques quite often focus on the recommendation of comparably popular objects.

### 5.3 Computational Complexity & Memory Usage

The methods included in our comparison vary largely in terms of the computational complexity and their memory requirements. Since neighborhood-based methods do not scale well when applied in a naive manner, we used implementation variants that rely on neighborhood sampling and specific in-memory data structures. The comparison of SKNN method and GRU4REC in [Jannach and Ludewig 2017] showed that, with such an implementation, recommendations can be quickly computed at prediction time with nearest neighbor methods, even though the prediction performance of model-based techniques like GRU4REC could not be achieved.

To enable comparability with previous research [Jannach and Ludewig 2017], we report the running times and memory demands for the single-split *RSC15-S* dataset, which is also the largest one in terms of the recorded user actions. Additionally, we include the *8TRACKS* dataset, which is rather small compared to *RSC15-S* in terms of the number of events, but has the largest product catalog of all datasets. Table 8 shows the times required for training the model (if applicable), the time needed to compute a recommendation at prediction time, and the memory requirements for the internal data structures. The reported results were obtained when using an Intel Core i7 4790K processor with 32GB of DDR3-1600 memory and a Nvidia GeForce GTX 960 graphics card with 2GB of memory. The following observations can be made.

Table 8. Overview of Computation Times and Memory Requirements for the *RSC15-S* dataset and the first split of the *8TRACKS* dataset, ordered in terms of required training times for the *RSC15-S* dataset.

| Dataset | RSC15-S | | | 8TRACKS | | |
|---|---|---|---|---|---|---|
| Technique/Metric | Train. (min) | Pred. (ms) | Mem. (MB) | Train. | Pred. | Mem. |
| MC | 0.77 | 3.34 | 38 | 0.05 | 14.71 | 144 |
| S-KNN | 1.24 | 33.05 | 6051 | 0.04 | 52.96 | 353 |
| S-SKNN | 1.26 | 30.26 | 6051 | 0.05 | 51.01 | 353 |
| V-SKNN | 1.30 | 32.67 | 6051 | 0.04 | 52.43 | 353 |
| SF-SKNN | 1.72 | 29.82 | 6254 | 0.06 | 18.82 | 493 |
| SR | 2.41 | 3.14 | 54 | 0.18 | 15.88 | 284 |
| AR | 3.00 | 3.36 | 40 | 0.28 | 15.79 | 257 |
| FISM | 356.84 | 8.40 | 4937 | 35.07 | 60.72 | 387 |
| GRU4REC (GPU) | 385.35 | 7.43 | 59 | 19.08 | 58.08 | 588 |
| BPR-MF | 392.60 | 8.37 | 8009 | 42.69 | 65.15 | 771 |
| SMF (GPU) | 446.66 | 14.02 | 1640 | 77.50 | 43.36 | 1805 |
| FPMC | 469.39 | 9.08 | 6786 | 60.92 | 71.58 | 1302 |
| FOSSIL | 499.19 | 10.56 | 4987 | 50.99 | 64.91 | 582 |

*Running Times.* The simple methods in our comparative evaluation need from less than one to about three minutes of "training" (e.g., co-occurrence counting or in-memory data structure setup) for the *RSC15-S* dataset. The factorization-based methods and the deep learning based method, on the other hand, need about 6 to 8 hours to learn a model for the single data split. Note that while the deep learning method GRU4REC and the factorization-based approach SMF do not take the longest absolute time in this comparison, they are the only method for which the computations are done on the GPU. Running GRU4REC, for example, on a CPU tripled the computation times according to the measurements in [Jannach and Ludewig 2017].

Looking at the times needed to compute a single recommendation list, given a session beginning, we can observe that the simple rule-based methods AR, MC, and SR are among the fastest ones with prediction times at about 3 ms for the *RSC15-S* dataset. The factorization-based methods and GRU4REC are also very efficient, with prediction times mostly below 10 ms on average. The nearest-neighbor methods are slower for this task as they have to consider the neighbors in the prediction process. Since the neighbors can be determined through fast lookup operations, the overall prediction time even for the more elaborate S-SKNN and V-SKNN similarity schemes never exceeds 33 ms for creating a recommendation list.

Looking at the *8TRACKS* dataset with its large number of items, we can, however, see that the prediction times for many algorithms, including GRU4REC and several of the factorization-based ones significantly increase, while the prediction time for the neighborhood models only doubles. In the end, making the neighborhood-based computations is at least as fast as computing the predictions based on the offline-trained models. Overall, due to the used in-memory data structures and through the neighborhood sampling approach, such neighborhood models are also suited under the narrow time constraints of real-time recommendations. Differently from other methods, newly arriving interaction signals can be easily included in the underlying model without re-training [Jugovac et al. 2018].

*Memory Requirements.* In terms of the memory requirements, the rule-based methods AR, MC, and SR that basically record item co-occurrences of size two require the least memory, i.e., below 100 megabytes. Also the memory demands of GRU4REC are very low in this comparison, and GRU4REC occupies only about 60 MB of memory on the graphics card for the *RSC15-S* dataset. The factorization-based methods and the neighborhood methods, in contrast, have substantially higher memory requirements. The lookup data structures of the neighborhood-based methods, for

example, in our implementation occupy about 6 GB of memory. When additional recency-based sampling is applied, which according to the analyses above does not hurt accuracy, these demands could, however, be substantially lowered.

For some algorithms, the memory requirements largely depend on the characteristics of the datasets. Looking at the numbers for the *8TRACKS* dataset, which covers over 300,000 different items (in contrast to the about 30,000 of the *RSC15* dataset), we see that in particular the memory demand of GRU4REC substantially increases with the number of items. As a result, GRU4REC's network even needs more memory than neighborhood-based methods for this dataset. Given these observations it seems promising to implement additional data sampling strategies within the more complex methods—as we did for the nearest neighbor methods—to decrease their computational demands.

## 6 CONCLUSION AND FUTURE DIRECTIONS

### 6.1 Summary of Main Insights

Being able to predict the user's short-term interest in an online session is a highly relevant problem in practice, which has raised increased interest also in the academic field in recent years. Even though a number of different algorithmic approaches were proposed over the years, no standard benchmark datasets and baseline algorithms exist today. In this work, we have compared a number of very recent and computationally complex algorithms for session-based recommendation with more light-weight approaches based, e.g., on session neighborhoods. The experimental analyses on a number of different datasets show that in many cases one of the simpler methods is able to outperform even the most recent methods based on recurrent neural networks in terms of the prediction accuracy. At the same time, the computational demands of these methods can be kept comparably low when using in-memory cache data structures and data sampling.

Overall, the results, therefore, indicate that additional research is required with respect to the development of sophisticated models that are more flexible in terms of how much sequential information is contained in the training data. This is in particular the case as several improvements for the nearest-neighbor methods can be imagined as well. In this work, we could for example observe that already using a different similarity measure, as done in the V-SKNN method, can lead to substantial performance improvements for different datasets. As a side result, we noticed that using the latent feature vectors of the items of the current session for sequential factorization-based methods does not lead to high accuracy values and that such methods are usually not strong baselines when comparing session-based algorithms.

Currently, constantly new deep learning-based algorithms for session-based recommendation are proposed, e.g., [Liu et al. 2018] and [Li et al. 2018], which, for example, report improvements over GRU4REC. We performed an initial evaluation of the STAMP method proposed in [Liu et al. 2018]. Our first results indicate that STAMP does not outperform the trivial SR technique in terms of the MRR on the *Diginetica* dataset that was used for the evaluation in [Liu et al. 2018]. The STAMP method, however, seems to be advantageous in terms of the hit rate for this particular dataset.

Generally, it is of course surprising that a recent and popular RNN-based method is not substantially better than longer-existing nearest neighbor approaches. We believe that this might be a result of the fact that for the specific task of session-based recommendation no "standard" existed so far with respect to baseline techniques and evaluation protocols. With this work, our aim is to contribute better baselines to benchmark session-based algorithms in the future. A limitation of our work in some sense is that we could not identify *one* best baseline method across all settings and datasets. While we would identify at least one very-well performing simpler method for each

dataset, the relative performance of the algorithms seems depend on a number of factors, which are not yet fully understood.

Nevertheless, as the simple baseline approaches AR, SR, and S-KNN are computationally cheap and easy to test, their results obtained for a given dataset can potentially be used as an indicator for the general characteristics that a more complex model should aim to implement. If it, for example, turns out that SR is the best performing baseline method, GRU4REC, an extension to GRU4REC or a different sequential model might probably be a good choice. In contrast, a good performance of S-KNN indicates that a more sophisticated model should not necessarily focus too much on the order of the items in a session.

### 6.2 Future Directions

From an algorithmic perspective, we believe that future complex models should consider more than the last event in a session when making the next-item prediction. Even in GRU4REC, the previous items of a session are only considered implicitly through the hidden states in the prediction process. Our neighborhood models are in most cases much better when they consider all events in a session, albeit with a focus on the most recent interactions. In that context and in particular for longer sessions, it might also be helpful to detect interest changes that happen *within* an individual session. This could, for example be achieved by considering *semantic* information (e.g., meta-data or content features) about the items of the session, as was done, for example, in [Hariri et al. 2012] or [Hidasi et al. 2016b]. Recent advances in the area of deep learning might be particularly helpful in this context to extract such content features, e.g., from text, images, or videos, and to use this information in *hybrid* approaches. Furthermore, the work in this paper focused on item-view events and more research is required to understand how to leverage other types of user actions like "add-to-wishlist" or "add-to-cart" in the learning and prediction process. With that information at hand, also other types of prediction problems can be addressed, e.g., whether or not a session will lead to a purchase or if there is a high probability that the user will abandon the session.

Going beyond the current session, more research also seems required in the area of *session-aware* recommendation and the consideration of previous sessions of the current user. Open questions in this area are, for example, how to model general long-term user preferences (e.g., towards certain brands in e-commerce), how to detect user-individual preference drifts, or how to identify a subset of past sessions that are good predictors for the current one. This latter aspect was for example explored in [Lerche et al. 2016] in the context of using recommendations as reminders. In addition, more elaborate strategies than static weighting schemes can be envisioned when combining short-term and long-term models. The importance weights, could for example be determined based on the length of the current session or the specific items that were considered.

Besides the consideration of signals at the individual user level, future research might also explore the incorporation of additional contextual factors or short-term trends in the community as a whole, when predicting the relevance of individual items. Recent works [Jannach and Ludewig 2017; Jannach et al. 2017b; Tan et al. 2016] for example showed that considering short-term popularity trends and recency effects can lead to significant performance improvements in the e-commerce domain. Item recency (freshness) also plays a particular role in other domains such as music and news recommendation, and more work is required to understand how to integrate these aspects in today's recommendation algorithms.

Finally, since the relative performance of the different algorithms tested in our work sometimes varies across different datasets, more research is required to understand in which situations certain algorithms are better suited than others. These insights can then be further used to inform the design of hybrid recommendation approaches, which have shown to lead to the highest recommendation accuracy for session-based recommendation also in [Jannach and Ludewig 2017]. Generally, many

factors can influence the performance of a certain recommendation algorithm. [Adomavicius and Zhang 2012] have, for example, made a number of important analyses aiming to relate dataset characteristics, e.g., rating distributions and dataset sizes, with prediction accuracy. In the context of session-based recommendation problems, additional factors may have an influence, for example, the existence and strength of the sequential patterns that can be found in the data. Furthermore, often domain-specific aspects like item freshness and general item popularity might play important roles and should be further explored in future research.

## REFERENCES


Gediminas Adomavicius and YoungOk Kwon. 2012. Improving Aggregate Recommendation Diversity Using Ranking-Based Techniques. *IEEE Trans. on Knowl. and Data Eng.* 24, 5 (May 2012), 896–911.

Gediminas Adomavicius and Jingjing Zhang. 2012. Impact of Data Characteristics on Recommender Systems Performance. *ACM Trans. Manage. Inf. Syst.* 3, 1 (2012), 3:1–3:17.

Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. 1993. Mining Association Rules Between Sets of Items in Large Databases. In *SIGMOD '93*. 207–216.

Ricardo Baeza-Yates, Di Jiang, Fabrizio Silvestri, and Beverly Harrison. 2015. Predicting The Next App That You Are Going To Use. In *WSDM '15*. 285–294.

Daniel Billsus, Michael J. Pazzani, and James Chen. 2000. A Learning Agent for Wireless News Access. In *IUI '00*. 33–36.

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

Shuo Chen, Josh L. Moore, Douglas Turnbull, and Thorsten Joachims. 2012. Playlist Prediction via Metric Embedding. In *KDD '12*. 714–722.

Shuo Chen, Jiexun Xu, and Thorsten Joachims. 2013. Multi-space Probabilistic Sequence Modeling. In *KDD '13*. 865–873.

Chen Cheng, Haiqin Yang, Michael R. Lyu, and Irwin King. 2013. Where You Like to Go Next: Successive Point-of-interest Recommendation. In *IJCAI '13*. 2605–2611.

Kyunghyun Cho, Bart van Merriënboer, Çağlar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *EMNLP '14*. 1724–1734.

Robert Cooley, Bamshad Mobasher, and Jaideep Srivastava. 1999. Data Preparation for Mining World Wide Web Browsing Patterns. *Knowledge and Information Systems* 1, 1 (1999), 5–32.

Abhinandan S. Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. 2007. Google News Personalization: Scalable Online Collaborative Filtering. In *Proceedings of the 16th International Conference on World Wide Web (WWW '07)*. 271–280.

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

Robin Devooght and Hugues Bersini. 2017. Long and Short-Term Recommendations with Recurrent Neural Networks. In *UMAP '17*. 13–21.

Nemanja Djuric, Vladan Radosavljevic, Mihajlo Grbovic, and Narayan Bhamidipati. 2014. Hidden Conditional Random Fields with Deep User Embeddings for Ad Targeting. In *ICDM '14*. 779–784.

Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. 2016. Recurrent Marked Temporal Point Processes: Embedding Event History to Vector. In *KDD '16*. 1555–1564.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *J. Mach. Learn. Res.* 12 (July 2011), 2121–2159.

Shanshan Feng, Xutao Li, Yifeng Zeng, Gao Cong, Yeow Meng Chee, and Quan Yuan. 2015. Personalized Ranking Metric Embedding for Next New POI Recommendation. In *IJCAI '15*. 2069–2075.

Florent Garcin, Christos Dimitrakakis, and Boi Faltings. 2013. Personalized News Recommendation with Context Trees. In *RecSys '13*. 105–112.

Mihajlo Grbovic, Vladan Radosavljevic, Nemanja Djuric, Narayan Bhamidipati, Jaikit Savla, Varun Bhagwan, and Doug Sharp. 2015. E-commerce in Your Inbox: Product Recommendations at Scale. In *KDD '15*. 1809–1818.

Negar Hariri, Bamshad Mobasher, and Robin Burke. 2012. Context-aware music recommendation based on latent topic sequential patterns. In *RecSys '12*. 131–131.

Jing He, Xin Li, Lejian Liao, Dandan Song, and William Cheung. 2016. Inferring a Personalized Next Point-of-Interest Recommendation Model with Latent Behavior Patterns. In *AAAI '16*.

Qi He, Daxin Jiang, Zhen Liao, Steven C. H. Hoi, Kuiyu Chang, Ee-Peng Lim, and Hang Li. 2009. Web Query Recommendation via Sequential Query Prediction. In *ICDE '09*. 1443–1454.

Ruining He and Julian McAuley. 2016. Fusing Similarity Models with Markov Chains for Sparse Sequential Recommendation. *CoRR* abs/1609.09152 (2016). https://arxiv.org/abs/1609.09152

Balázs Hidasi and Alexandros Karatzoglou. 2017. Recurrent Neural Networks with Top-k Gains for Session-based Recommendations. *CoRR* abs/1706.03847 (2017). arXiv:1706.03847 http://arxiv.org/abs/1706.03847

Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016a. Session-based Recommendations with Recurrent Neural Networks. In *ICLR '16*.

Balázs Hidasi, Massimo Quadrana, Alexandros Karatzoglou, and Domonkos Tikk. 2016b. Parallel Recurrent Neural Network Architectures for Feature-rich Session-based Recommendations. In *RecSys '16*. 241–248.

Mehdi Hosseinzadeh Aghdam, Negar Hariri, Bamshad Mobasher, and Robin Burke. 2015. Adapting Recommendations to Contextual Changes Using Hierarchical Hidden Markov Models. In *RecSys '15*. 241–244.

Dietmar Jannach and Gediminas Adomavicius. 2016. Recommendations with a Purpose. In *RecSys '16*. 7–10.

Dietmar Jannach and Kolja Hegelich. 2009. A Case Study on the Effectiveness of Recommendations in the Mobile Internet. In *RecSys '09*. 205–208.

Dietmar Jannach, Iman Kamehkhosh, and Lukas Lerche. 2017a. Leveraging Multi-Dimensional User Models for Personalized Next-Track Music Recommendation. In *ACM SAC 2017*.

Dietmar Jannach, Lukas Lerche, and Michael Jugovac. 2015a. Adaptation and Evaluation of Recommendations for Short-term Shopping Goals. In *RecSys '15*. 211–218.

Dietmar Jannach, Lukas Lerche, Iman Kamehkhosh, and Michael Jugovac. 2015b. What recommenders recommend: an analysis of recommendation biases and possible countermeasures. *User Modeling and User-Adapted Interaction* 25, 5 (2015), 427–491.

Dietmar Jannach and Malte Ludewig. 2017. When Recurrent Neural Networks meet the Neighborhood for Session-Based Recommendation. In *RecSys '17*. 306–310.

Dietmar Jannach, Malte Ludewig, and Lukas Lerche. 2017b. Session-based Item Recommendation in E-Commerce: On Short-Term Intents, Reminders, Trends, and Discounts. *User-Modeling and User-Adapted Interaction* 27, 3–5 (2017), 351–392.

Michael Jugovac, Dietmar Jannach, and Mozhgan Karimi. 2018. StreamingRec: A Framework for Benchmarking Stream-based News Recommenders. In *RecSys 2018*.

Santosh Kabbur, Xia Ning, and George Karypis. 2013. FISM: Factored Item Similarity Models for top-N Recommender Systems. In *KDD '13*. 659–667.

Iman Kamehkhosh, Dietmar Jannach, and Malte Ludewig. 2017. A Comparison of Frequent Pattern Techniques and a Deep Learning Method for Session-Based Recommendation. In *TempRec Workshop at ACM RecSys '17*. Como, Italy.

Mozhgan Karimi, Dietmar Jannach, and Michael Jugovac. 2018. News Recommender Systems - Survey and Roads Ahead. *Information Processing and Management* (2018).

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014). arXiv:1412.6980 http://arxiv.org/abs/1412.6980

Dokyun Lee and Kartik Hosanagar. 2014. Impact of Recommender Systems on Sales Volume and Diversity. In *ICIS 2014*.

Lukas Lerche, Dietmar Jannach, and Malte Ludewig. 2016. On the Value of Reminders Within E-Commerce Recommendations. In *UMAP '16*. 27–35.

Zhi Li, Hongke Zhao, Qi Liu, Zhenya Huang, Tao Mei, and Enhong Chen. 2018. Learning from History and Present: Next-item Recommendation via Discriminatively Exploiting User Behaviors. In *KDD 2018*.

Defu Lian, Vincent W. Zheng, and Xing Xie. 2013. Collaborative Filtering Meets Next Check-in Location Prediction. In *WWW '13*. 231–232.

Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon.Com Recommendations: Item-to-Item Collaborative Filtering. *IEEE Internet Computing* 7, 1 (Jan. 2003), 76–80.

Jiahui Liu, Peter Dolan, and Elin Rønby Pedersen. 2010. Personalized News Recommendation Based on Click Behavior. In *IUI '10*. 31–40.

Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: Short-Term Attention/Memory Priority Model for Session-based Recommendation. In *KDD 2018*.

Yanchi Liu, Chuanren Liu, Bin Liu, Meng Qu, and Hui Xiong. 2016. Unified Point-of-Interest Recommendation with Temporal Interval Assessment. In *KDD '16*. 1015–1024.

Cornelius A. Ludmann. 2017. Recommending News Articles in the CLEF News Recommendation Evaluation Lab with the Data Stream Management System Odysseus. In *Working Notes of CLEF 2017 - Conference and Labs of the Evaluation*.

Brian Mcfee and Gert Lanckriet. 2011. The Natural Language of Playlists. In *ISMIR '11*. 537–541.

Brian McFee and Gert R. G. Lanckriet. 2012. Hypergraph Models of Playlist Dialects. In *ISMIR '12*. 343–348.

Bamshad Mobasher, Honghua Dai, Tao Luo, and Miki Nakagawa. 2002. Using Sequential and Non-Sequential Patterns in Predictive Web Usage Mining Tasks. In *ICDM '02*. 669–672.

Omar Moling, Linas Baltrunas, and Francesco Ricci. 2012. Optimal Radio Channel Recommendations with Explicit and Implicit Feedback. In *RecSys '12*. 75–82.

Nagarajan Natarajan, Donghyuk Shin, and Inderjit S. Dhillon. 2013. Which App Will You Use Next?: Collaborative Filtering with Interactional Context. In *RecSys '13*. 201–208.

J.R. Norris. 1997. *Markov Chains*. Cambridge University Press, Cambridge.

Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. 2018. Sequence-Aware Recommender Systems. *Comput. Surveys* 54 (2018), 1–36. Issue 1.

Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. 2017. Personalizing Session-based Recommendations with Hierarchical Recurrent Neural Networks. In *RecSys '17*.

Siddharth Reddy, Igor Labutov, and Thorsten Joachims. 2016. Learning Student and Content Embeddings for Personalized Lesson Sequence Recommendation. In *ACM Learning @ Scale '16*. 93–96.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI '09*. 452–461.

Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing Personalized Markov Chains for Next-basket Recommendation. In *WWW '10*. 811–820.

Guy Shani, David Heckerman, and Ronen I. Brafman. 2005. An MDP-Based Recommender System. *J. Mach. Learn. Res.* 6 (2005), 1265–1295.

Harold Soh, Scott Sanner, Madeleine White, and Greg Jamieson. 2017. Deep Sequential Recommendation for Personalized Adaptive User Interfaces. In *IUI '17*. 589–593.

Qiang Song, Jian Cheng, Ting Yuan, and Hanqing Lu. 2015. Personalized Recommendation Meets Your Next Favorite. In *CIKM '15*. 1775–1778.

Yang Song, Ali Mamdouh Elkahky, and Xiaodong He. 2016. Multi-Rate Deep Learning for Temporal Recommendation. In *SIGIR '16*. 909–912.

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015. A Hierarchical Recurrent Encoder-Decoder for Generative Context-Aware Query Suggestion. In *CIKM '15*. 553–562.

Yukihiro Tagami, Hayato Kobayashi, Shingo Ono, and Akira Tajima. 2015. Modeling User Activities on the Web Using Paragraph Vector. In *WWW '15*. 125–126.

Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved Recurrent Neural Networks for Session-based Recommendations. In *DLRS '16 Workshop at ACM RecSys*. 17–22.

Maryam Tavakol and Ulf Brefeld. 2014. Factored MDPs for Detecting Topics of User Sessions. In *RecSys '14*. 33–40.

Roberto Turrin, Massimo Quadrana, Andrea Condorelli, Roberto Pagano, and Paolo Cremonesi. 2015. 30Music Listening and Playlists Dataset. In *Poster Proceedings of RecSys '15*.

Bartlomiej Twardowski. 2016. Modelling Contextual Information in Session-Aware Recommender Systems with Neural Networks. In *RecSys '16*. 273–276.

Flavian Vasile, Elena Smirnova, and Alexis Conneau. 2016. Meta-Prod2Vec: Product Embeddings Using Side-Information for Recommendation. In *RecSys '16*. 225–232.

Koen Verstrepen and Bart Goethals. 2014. Unifying Nearest Neighbors Collaborative Filtering. In *RecSys '14*. 177–184.

Xiang Wu, Qi Liu, Enhong Chen, Liang He, Jingsong Lv, Can Cao, and Guoping Hu. 2013. Personalized Next-song Recommendation in Online Karaokes. In *RecSys '13*. 137–140.

Ghim-Eng Yap, Xiao-Li Li, and Philip S. Yu. 2012. Effective Next-items Recommendation via Personalized Sequential Pattern Mining. In *DASFAA '12, Volume Part II*. 48–64.

Feng Yu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. A Dynamic Recurrent Model for Next Basket Recommendation. In *SIGIR '16*. 729–732.

Eva Zangerle, Martin Pichl, Wolfgang Gassler, and Günther Specht. 2014. #Nowplaying Music Dataset: Extracting Listening Behavior from Twitter. In *WISMM '14 Workshop at MM '14*. 21–26.

Matthew D. Zeiler. 2012. ADADELTA: An Adaptive Learning Rate Method. *CoRR* abs/1212.5701 (2012). arXiv:1212.5701 http://arxiv.org/abs/1212.5701

Yuyu Zhang, Hanjun Dai, Chang Xu, Jun Feng, Taifeng Wang, Jiang Bian, Bin Wang, and Tie-Yan Liu. 2014. Sequential Click Prediction for Sponsored Search with Recurrent Neural Networks. In *AAAI '14*. 1369–1375.

Elena Zheleva, John Guiver, Eduarda Mendes Rodrigues, and Nataša Milić-Frayling. 2010. Statistical Models of Music-listening Sessions in Social Media. In *WWW '10*. 1019–1028.

## AUTHOR BIOGRAPHIES

*Malte Ludewig* is a PhD candidate in Computer Science at TU Dortmund, Germany, from where he also received his MSc degree. His research interests lie in the field of recommender systems—with a focus on session-based recommendations—and personalization in e-commerce environments in general.

*Dietmar Jannach* is a Professor of Computer Science at AAU Klagenfurt, Austria, and head of the department's information systems research group. Dr. Jannach has worked on different areas of artificial intelligence, including recommender systems, model-based diagnosis, and knowledge-based systems. He is the leading author of a textbook on recommender systems and has authored more than hundred research papers, focusing on the application of artificial intelligence technology to practical problems.

## A PARAMETER CONFIGURATIONS

Table 9. Parameters for algorithm GRU4REC for all datasets.

| Dataset | layer_size | objective | lr | momentum | drop_out |
|---|---|---|---|---|---|
| RSC15 | 100 | $BPR_{max}$ | 0.20 | 0.5 | 0.0 |
| TMALL | 100 | $TOP1_{max}$ | 0.05 | 0.0 | 0.3 |
| RETAILROCKET | 100 | $TOP1_{max}$ | 0.15 | 0.3 | 0.0 |
| 8TRACKS | 100 | $TOP1_{max}$ | 0.10 | 0.0 | 0.7 |
| AOTM | 100 | $BPR_{max}$ | 0.10 | 0.3 | 0.1 |
| NOWPLAYING | 100 | $BPR_{max}$ | 0.10 | 0.5 | 0.1 |
| 30MUSIC | 100 | $TOP1_{max}$ | 0.10 | 0.1 | 0.1 |
| ZALANDO | 100 | $BPR_{max}$ | 0.20 | 0.1 | 0.1 |
| CLEF | 100 | $TOP1_{max}$ | 0.20 | 0.2 | 0.5 |
| LASTFM | 100 | $BPR_{max}$ | 0.15 | 0.4 | 0.3 |

Table 10. Parameters used for the SMF algorithm for all datasets.

| Dataset | layer_size | objective | lr | momentum | drop_out | skip |
|---|---|---|---|---|---|---|
| RSC15 | 100 | $TOP1_{max}$ | 0.085 | 0.2 | 0.30 | 0.00 |
| TMALL | 100 | $BPR_{max}$ | 0.015 | 0.6 | 0.00 | 0.00 |
| RETAILROCKET | 100 | $BPR_{max}$ | 0.045 | 0.1 | 0.40 | 0.20 |
| 8TRACKS | 100 | $TOP1_{max}$ | 0.010 | 0.5 | 0.30 | 0.35 |
| AOTM | 100 | $BPR_{max}$ | 0.090 | 0.8 | 0.40 | 0.20 |
| NOWPLAYING | 100 | $TOP1_{max}$ | 0.055 | 0.2 | 0.40 | 0.20 |
| 30MUSIC | 100 | $TOP1_{max}$ | 0.100 | 0.1 | 0.20 | 0.20 |
| ZALANDO | 100 | $TOP1_{max}$ | 0.030 | 0.3 | 0.25 | 0.00 |
| CLEF | 100 | $BPR_{max}$ | 0.050 | 0.0 | 0.40 | 0.25 |
| LASTFM | 100 | $TOP1_{max}$ | 0.015 | 0.3 | 0.15 | 0.45 |

Table 11. Parameters used for the V-SKNN algorithm for all datasets.

| Dataset | K | samples |
|---|---|---|
| RSC15 | 200 | 2000 |
| TMALL | 200 | 2000 |
| RETAILROCKET | 200 | 2000 |
| ZALANDO | 200 | 2000 |
| 8TRACKS | 200 | 2000 |
| AOTM | 200 | 2000 |
| NOWPLAYING | 100 | 1000 |
| 30MUSIC | 100 | 1000 |
| CLEF | 100 | 1000 |
| LASTFM | 200 | 2000 |

Table 12. Parameters used for the SKNN, S-SKNN, and SF-SKNN algorithm for all datasets.

| Dataset | K | samples |
|---|---|---|
| RSC15 | 500 | 1000 |
| TMALL | 100 | 500 |
| RETAILROCKET | 100 | 500 |
| ZALANDO | 100 | 500 |
| 8TRACKS | 100 | 500 |
| AOTM | 100 | 500 |
| NOWPLAYING | 100 | 500 |
| 30MUSIC | 100 | 500 |
| CLEF | 100 | 500 |
| LASTFM | 100 | 500 |

## B FULL RESULT TABLES

Table 13. Hit rate (HR), Mean reciprocal rank (MRR), item coverage (COV), and average popularity (POP) results for a list length of 20, 10, 5, 3, and 1 on the RSC15 dataset (sorted by MRR@20).

| Algorithm | MRR@20 | HR@20 | COV@20 | POP@20 | MRR@10 | HR@10 | COV@10 | POP@10 |
|---|---|---|---|---|---|---|---|---|
| GRU4REC | 0.308 | 0.683 | 0.504 | 0.054 | 0.301 | 0.591 | 0.431 | 0.058 |
| SR | 0.304 | 0.653 | 0.668 | 0.072 | 0.298 | 0.569 | 0.592 | 0.073 |
| SMF | 0.302 | 0.666 | 0.565 | 0.055 | 0.295 | 0.575 | 0.486 | 0.058 |
| MC | 0.300 | 0.642 | 0.645 | 0.070 | 0.295 | 0.562 | 0.584 | 0.071 |
| AR | 0.289 | 0.636 | 0.630 | 0.093 | 0.283 | 0.550 | 0.548 | 0.091 |
| V-SKNN | 0.283 | 0.653 | 0.619 | 0.079 | 0.277 | 0.563 | 0.534 | 0.081 |
| S-SKNN | 0.272 | 0.602 | 0.655 | 0.072 | 0.267 | 0.531 | 0.543 | 0.077 |
| SF-SKNN | 0.270 | 0.589 | 0.619 | 0.066 | 0.266 | 0.524 | 0.545 | 0.074 |
| S-KNN | 0.266 | 0.621 | 0.634 | 0.073 | 0.259 | 0.526 | 0.520 | 0.078 |
| IKNN | 0.208 | 0.486 | 0.755 | 0.041 | 0.203 | 0.408 | 0.671 | 0.046 |
| FPMC | 0.201 | 0.363 | 0.975 | 0.055 | 0.198 | 0.311 | 0.908 | 0.056 |
| BPR-MF | 0.176 | 0.235 | 0.911 | 0.088 | 0.175 | 0.223 | 0.793 | 0.079 |
| FISM | 0.115 | 0.162 | 0.974 | 0.008 | 0.114 | 0.149 | 0.917 | 0.012 |
| FOSSIL | 0.062 | 0.190 | 0.917 | 0.048 | 0.058 | 0.135 | 0.806 | 0.047 |
| Algorithm | MRR@5 | HR@5 | COV@5 | POP@5 | MRR@3 | HR@3 | MRR@1 | HR@1 |
| GRU4REC | 0.285 | 0.470 | 0.355 | 0.062 | 0.263 | 0.371 | 0.180 | 0.180 |
| SR | 0.283 | 0.457 | 0.503 | 0.075 | 0.262 | 0.365 | 0.182 | 0.182 |
| SMF | 0.280 | 0.459 | 0.406 | 0.061 | 0.258 | 0.364 | 0.177 | 0.177 |
| MC | 0.280 | 0.452 | 0.506 | 0.073 | 0.259 | 0.362 | 0.181 | 0.181 |
| AR | 0.268 | 0.438 | 0.455 | 0.092 | 0.248 | 0.348 | 0.171 | 0.171 |
| V-SKNN | 0.261 | 0.446 | 0.451 | 0.084 | 0.239 | 0.351 | 0.154 | 0.154 |
| S-SKNN | 0.252 | 0.424 | 0.437 | 0.082 | 0.231 | 0.333 | 0.153 | 0.153 |
| SF-SKNN | 0.252 | 0.421 | 0.457 | 0.081 | 0.232 | 0.332 | 0.154 | 0.154 |
| S-KNN | 0.244 | 0.411 | 0.417 | 0.084 | 0.224 | 0.322 | 0.149 | 0.149 |
| IKNN | 0.190 | 0.315 | 0.566 | 0.050 | 0.174 | 0.244 | 0.121 | 0.121 |
| FPMC | 0.191 | 0.261 | 0.774 | 0.058 | 0.183 | 0.226 | 0.148 | 0.148 |
| BPR-MF | 0.174 | 0.214 | 0.630 | 0.070 | 0.172 | 0.205 | 0.144 | 0.144 |
| FISM | 0.112 | 0.135 | 0.810 | 0.019 | 0.110 | 0.126 | 0.096 | 0.096 |
| FOSSIL | 0.052 | 0.092 | 0.653 | 0.046 | 0.047 | 0.067 | 0.031 | 0.031 |

Table 14. Hit rate (HR), Mean reciprocal rank (MRR), item coverage (COV), and average popularity (POP) results for a list length of 20, 10, 5, 3, and 1 on the RSC15-S dataset (sorted by MRR@20).

| Algorithm | MRR@20 | HR@20 | COV@20 | POP@20 | MRR@10 | HR@10 | COV@10 | POP@10 |
|---|---|---|---|---|---|---|---|---|
| SMF | 0.309 | 0.713 | 0.512 | 0.052 | 0.301 | 0.606 | 0.414 | 0.054 |
| GRU4REC | 0.308 | 0.719 | 0.350 | 0.033 | 0.301 | 0.609 | 0.277 | 0.034 |
| SR | 0.308 | 0.690 | 0.512 | 0.038 | 0.301 | 0.591 | 0.407 | 0.038 |
| MC | 0.296 | 0.667 | 0.518 | 0.039 | 0.289 | 0.567 | 0.413 | 0.039 |
| AR | 0.281 | 0.655 | 0.473 | 0.045 | 0.273 | 0.543 | 0.374 | 0.043 |
| V-SKNN | 0.274 | 0.675 | 0.427 | 0.037 | 0.266 | 0.562 | 0.328 | 0.039 |
| S-SKNN | 0.266 | 0.667 | 0.417 | 0.035 | 0.258 | 0.548 | 0.309 | 0.038 |
| SF-SKNN | 0.260 | 0.670 | 0.446 | 0.037 | 0.251 | 0.545 | 0.339 | 0.039 |
| S-KNN | 0.250 | 0.641 | 0.398 | 0.036 | 0.242 | 0.521 | 0.293 | 0.038 |
| Algorithm | MRR@5 | HR@5 | COV@5 | POP@5 | MRR@3 | HR@3 | MRR@1 | HR@1 |
| SMF | 0.284 | 0.476 | 0.320 | 0.056 | 0.259 | 0.368 | 0.177 | 0.177 |
| GRU4REC | 0.283 | 0.473 | 0.207 | 0.037 | 0.259 | 0.369 | 0.175 | 0.175 |
| SR | 0.284 | 0.468 | 0.308 | 0.040 | 0.262 | 0.371 | 0.179 | 0.179 |
| MC | 0.273 | 0.444 | 0.314 | 0.041 | 0.252 | 0.354 | 0.174 | 0.174 |
| AR | 0.257 | 0.422 | 0.283 | 0.046 | 0.236 | 0.333 | 0.163 | 0.163 |
| V-SKNN | 0.248 | 0.429 | 0.242 | 0.042 | 0.225 | 0.329 | 0.145 | 0.145 |
| S-SKNN | 0.240 | 0.414 | 0.221 | 0.041 | 0.218 | 0.318 | 0.142 | 0.142 |
| SF-SKNN | 0.233 | 0.406 | 0.246 | 0.042 | 0.210 | 0.307 | 0.137 | 0.137 |
| S-KNN | 0.224 | 0.390 | 0.207 | 0.041 | 0.203 | 0.297 | 0.133 | 0.133 |

Table 15. Hit rate (HR), Mean reciprocal rank (MRR), item coverage (COV), and average popularity (POP) results for a list length of 20, 10, 5, 3, and 1 on the TMALL dataset (sorted by MRR@20).

| Algorithm | MRR@20 | HR@20 | COV@20 | POP@20 | MRR@10 | HR@10 | COV@10 | POP@10 |
|---|---|---|---|---|---|---|---|---|
| S-SKNN | 0.185 | 0.387 | 0.467 | 0.025 | 0.181 | 0.330 | 0.309 | 0.027 |
| S-KNN | 0.182 | 0.404 | 0.381 | 0.026 | 0.177 | 0.334 | 0.249 | 0.029 |
| V-SKNN | 0.179 | 0.373 | 0.464 | 0.024 | 0.175 | 0.312 | 0.320 | 0.026 |
| BPR-MF | 0.159 | 0.204 | 0.723 | 0.057 | 0.159 | 0.197 | 0.534 | 0.076 |
| SF-SKNN | 0.136 | 0.216 | 0.436 | 0.018 | 0.135 | 0.203 | 0.338 | 0.022 |
| GRU4REC | 0.129 | 0.277 | 0.151 | 0.035 | 0.125 | 0.225 | 0.109 | 0.039 |
| AR | 0.129 | 0.262 | 0.509 | 0.021 | 0.126 | 0.217 | 0.358 | 0.024 |
| SR | 0.128 | 0.242 | 0.569 | 0.021 | 0.125 | 0.206 | 0.421 | 0.023 |
| SMF | 0.121 | 0.261 | 0.261 | 0.036 | 0.118 | 0.213 | 0.193 | 0.039 |
| MC | 0.116 | 0.200 | 0.498 | 0.019 | 0.114 | 0.178 | 0.391 | 0.022 |
| FPMC | 0.101 | 0.119 | 0.880 | 0.005 | 0.100 | 0.114 | 0.730 | 0.007 |
| IKNN | 0.051 | 0.150 | 0.728 | 0.007 | 0.048 | 0.112 | 0.575 | 0.008 |
| FISM | 0.024 | 0.037 | 0.752 | 0.003 | 0.023 | 0.032 | 0.586 | 0.003 |
| FOSSIL | 0.001 | 0.004 | 0.598 | 0.016 | 0.001 | 0.003 | 0.457 | 0.021 |
| Algorithm | MRR@5 | HR@5 | COV@5 | POP@5 | MRR@3 | HR@3 | MRR@1 | HR@1 |
| S-SKNN | 0.173 | 0.267 | 0.196 | 0.031 | 0.161 | 0.217 | 0.119 | 0.119 |
| S-KNN | 0.168 | 0.264 | 0.161 | 0.032 | 0.156 | 0.212 | 0.113 | 0.113 |
| V-SKNN | 0.167 | 0.251 | 0.218 | 0.029 | 0.157 | 0.207 | 0.118 | 0.118 |
| BPR-MF | 0.157 | 0.189 | 0.343 | 0.097 | 0.156 | 0.181 | 0.134 | 0.134 |
| SF-SKNN | 0.132 | 0.182 | 0.243 | 0.026 | 0.127 | 0.160 | 0.101 | 0.101 |
| GRU4REC | 0.119 | 0.177 | 0.078 | 0.043 | 0.112 | 0.145 | 0.086 | 0.086 |
| AR | 0.120 | 0.175 | 0.235 | 0.027 | 0.113 | 0.145 | 0.089 | 0.089 |
| SR | 0.120 | 0.170 | 0.286 | 0.026 | 0.114 | 0.144 | 0.091 | 0.091 |
| SMF | 0.112 | 0.168 | 0.140 | 0.041 | 0.105 | 0.138 | 0.079 | 0.079 |
| MC | 0.111 | 0.151 | 0.284 | 0.025 | 0.106 | 0.131 | 0.086 | 0.086 |
| FPMC | 0.100 | 0.109 | 0.540 | 0.010 | 0.099 | 0.105 | 0.093 | 0.093 |
| IKNN | 0.044 | 0.079 | 0.403 | 0.009 | 0.039 | 0.058 | 0.025 | 0.025 |
| FISM | 0.023 | 0.028 | 0.419 | 0.004 | 0.022 | 0.026 | 0.019 | 0.019 |
| FOSSIL | 0.001 | 0.002 | 0.310 | 0.028 | 0.001 | 0.002 | 0.001 | 0.001 |

Table 16. Hit rate (HR), Mean reciprocal rank (MRR), item coverage (COV), and average popularity (POP) results for a list length of 20, 10, 5, 3, and 1 on the RETAILROCKET dataset (sorted by MRR@20).

| Algorithm | MRR@20 | HR@20 | COV@20 | POP@20 | MRR@10 | HR@10 | COV@10 | POP@10 |
|---|---|---|---|---|---|---|---|---|
| S-SKNN | 0.345 | 0.591 | 0.596 | 0.056 | 0.341 | 0.537 | 0.480 | 0.066 |
| V-SKNN | 0.338 | 0.573 | 0.575 | 0.060 | 0.334 | 0.519 | 0.474 | 0.069 |
| S-KNN | 0.337 | 0.583 | 0.566 | 0.058 | 0.333 | 0.528 | 0.445 | 0.068 |
| BPR-MF | 0.303 | 0.357 | 0.824 | 0.060 | 0.303 | 0.352 | 0.627 | 0.072 |
| FPMC | 0.273 | 0.320 | 0.929 | 0.022 | 0.272 | 0.309 | 0.777 | 0.026 |
| SF-SKNN | 0.260 | 0.358 | 0.403 | 0.035 | 0.259 | 0.350 | 0.373 | 0.046 |
| SR | 0.245 | 0.419 | 0.524 | 0.042 | 0.243 | 0.386 | 0.458 | 0.050 |
| GRU4REC | 0.243 | 0.480 | 0.602 | 0.060 | 0.238 | 0.415 | 0.478 | 0.066 |
| AR | 0.241 | 0.439 | 0.544 | 0.053 | 0.238 | 0.390 | 0.449 | 0.061 |
| MC | 0.230 | 0.359 | 0.411 | 0.035 | 0.228 | 0.343 | 0.383 | 0.045 |
| SMF | 0.225 | 0.459 | 0.449 | 0.085 | 0.221 | 0.393 | 0.360 | 0.092 |
| IKNN | 0.107 | 0.240 | 0.584 | 0.033 | 0.105 | 0.202 | 0.505 | 0.038 |
| FISM | 0.075 | 0.132 | 0.848 | 0.018 | 0.074 | 0.112 | 0.672 | 0.019 |
| FOSSIL | 0.022 | 0.058 | 0.753 | 0.127 | 0.020 | 0.043 | 0.560 | 0.150 |
| Algorithm | MRR@5 | HR@5 | COV@5 | POP@5 | MRR@3 | HR@3 | MRR@1 | HR@1 |
| S-SKNN | 0.332 | 0.470 | 0.344 | 0.076 | 0.318 | 0.406 | 0.247 | 0.247 |
| V-SKNN | 0.326 | 0.455 | 0.348 | 0.078 | 0.312 | 0.396 | 0.245 | 0.245 |
| S-KNN | 0.324 | 0.458 | 0.316 | 0.080 | 0.310 | 0.396 | 0.242 | 0.242 |
| BPR-MF | 0.302 | 0.345 | 0.417 | 0.083 | 0.300 | 0.337 | 0.267 | 0.267 |
| FPMC | 0.271 | 0.298 | 0.560 | 0.032 | 0.269 | 0.289 | 0.251 | 0.251 |
| SF-SKNN | 0.257 | 0.331 | 0.311 | 0.058 | 0.250 | 0.302 | 0.208 | 0.208 |
| SR | 0.236 | 0.337 | 0.354 | 0.059 | 0.225 | 0.288 | 0.176 | 0.176 |
| GRU4REC | 0.229 | 0.345 | 0.350 | 0.072 | 0.215 | 0.285 | 0.161 | 0.161 |
| AR | 0.230 | 0.331 | 0.333 | 0.071 | 0.218 | 0.280 | 0.170 | 0.170 |
| MC | 0.224 | 0.308 | 0.322 | 0.056 | 0.215 | 0.270 | 0.171 | 0.171 |
| SMF | 0.211 | 0.322 | 0.270 | 0.099 | 0.198 | 0.264 | 0.148 | 0.148 |
| IKNN | 0.099 | 0.159 | 0.388 | 0.042 | 0.091 | 0.127 | 0.065 | 0.065 |
| FISM | 0.071 | 0.094 | 0.474 | 0.023 | 0.069 | 0.083 | 0.058 | 0.058 |
| FOSSIL | 0.019 | 0.032 | 0.377 | 0.171 | 0.017 | 0.024 | 0.012 | 0.012 |

Table 17. Hit rate (HR), Mean reciprocal rank (MRR), item coverage (COV), and average popularity (POP) results for a list length of 20, 10, 5, 3, and 1 on the ZALANDO dataset (sorted by MRR@20).

| Algorithm | MRR@20 | HR@20 | COV@20 | POP@20 | MRR@10 | HR@10 | COV@10 | POP@10 |
|---|---|---|---|---|---|---|---|---|
| SR | 0.304 | 0.483 | 0.586 | 0.061 | 0.302 | 0.462 | 0.433 | 0.066 |
| MC | 0.303 | 0.455 | 0.513 | 0.060 | 0.302 | 0.441 | 0.412 | 0.066 |
| IKNN | 0.275 | 0.405 | 0.714 | 0.037 | 0.273 | 0.385 | 0.532 | 0.041 |
| GRU4REC | 0.267 | 0.468 | 0.304 | 0.101 | 0.265 | 0.433 | 0.239 | 0.103 |
| SMF | 0.267 | 0.447 | 0.362 | 0.107 | 0.265 | 0.418 | 0.282 | 0.108 |
| AR | 0.258 | 0.467 | 0.467 | 0.089 | 0.256 | 0.435 | 0.337 | 0.090 |
| SF-SKNN | 0.249 | 0.438 | 0.432 | 0.057 | 0.249 | 0.430 | 0.348 | 0.068 |
| V-SKNN | 0.233 | 0.521 | 0.432 | 0.096 | 0.230 | 0.482 | 0.296 | 0.096 |
| S-SKNN | 0.219 | 0.499 | 0.435 | 0.087 | 0.216 | 0.456 | 0.280 | 0.092 |
| S-KNN | 0.172 | 0.456 | 0.309 | 0.093 | 0.167 | 0.380 | 0.201 | 0.097 |
| BPR-MF | 0.104 | 0.162 | 0.609 | 0.058 | 0.103 | 0.152 | 0.415 | 0.069 |
| FPMC | 0.051 | 0.075 | 0.812 | 0.021 | 0.050 | 0.067 | 0.629 | 0.022 |
| FISM | 0.004 | 0.011 | 0.624 | 0.020 | 0.004 | 0.008 | 0.444 | 0.020 |
| FOSSIL | 0.002 | 0.005 | 0.671 | 0.034 | 0.002 | 0.004 | 0.493 | 0.036 |
| Algorithm | MRR@5 | HR@5 | COV@5 | POP@5 | MRR@3 | HR@3 | MRR@1 | HR@1 |
| SR | 0.298 | 0.429 | 0.290 | 0.069 | 0.287 | 0.382 | 0.211 | 0.211 |
| MC | 0.298 | 0.415 | 0.292 | 0.069 | 0.289 | 0.377 | 0.218 | 0.218 |
| IKNN | 0.270 | 0.362 | 0.349 | 0.047 | 0.264 | 0.335 | 0.205 | 0.205 |
| GRU4REC | 0.259 | 0.389 | 0.182 | 0.100 | 0.247 | 0.337 | 0.177 | 0.177 |
| SMF | 0.259 | 0.380 | 0.210 | 0.104 | 0.249 | 0.333 | 0.183 | 0.183 |
| AR | 0.250 | 0.393 | 0.233 | 0.088 | 0.237 | 0.338 | 0.159 | 0.159 |
| SF-SKNN | 0.245 | 0.403 | 0.249 | 0.074 | 0.232 | 0.348 | 0.142 | 0.142 |
| V-SKNN | 0.222 | 0.422 | 0.197 | 0.092 | 0.205 | 0.346 | 0.095 | 0.095 |
| S-SKNN | 0.207 | 0.388 | 0.174 | 0.095 | 0.189 | 0.311 | 0.095 | 0.095 |
| S-KNN | 0.154 | 0.290 | 0.125 | 0.103 | 0.137 | 0.216 | 0.079 | 0.079 |
| BPR-MF | 0.102 | 0.141 | 0.247 | 0.083 | 0.099 | 0.130 | 0.073 | 0.073 |
| FPMC | 0.049 | 0.061 | 0.434 | 0.025 | 0.048 | 0.056 | 0.042 | 0.042 |
| FISM | 0.004 | 0.006 | 0.290 | 0.021 | 0.004 | 0.005 | 0.003 | 0.003 |
| FOSSIL | 0.002 | 0.003 | 0.333 | 0.037 | 0.002 | 0.002 | 0.001 | 0.001 |

Table 18. Hit rate (HR), Mean reciprocal rank (MRR), item coverage (COV), and average popularity (POP) results for a list length of 20, 10, 5, 3, and 1 on the 8TRACKS dataset (sorted by MRR@20).

| Algorithm | MRR@20 | HR@20 | COV@20 | POP@20 | MRR@10 | HR@10 | COV@10 | POP@10 |
|---|---|---|---|---|---|---|---|---|
| AR | 0.0071 | 0.0255 | 0.4530 | 0.0912 | 0.0066 | 0.0173 | 0.3178 | 0.1075 |
| SMF | 0.0064 | 0.0231 | 0.1528 | 0.0864 | 0.0058 | 0.0148 | 0.1076 | 0.0916 |
| SR | 0.0064 | 0.0171 | 0.4967 | 0.0531 | 0.0061 | 0.0125 | 0.3645 | 0.0636 |
| SF-SKNN | 0.0064 | 0.0119 | 0.3049 | 0.0363 | 0.0063 | 0.0102 | 0.2439 | 0.0517 |
| V-SKNN | 0.0057 | 0.0352 | 0.4081 | 0.1194 | 0.0048 | 0.0211 | 0.2523 | 0.1353 |
| S-KNN | 0.0053 | 0.0376 | 0.2431 | 0.1080 | 0.0041 | 0.0198 | 0.1544 | 0.1153 |
| IKNN | 0.0051 | 0.0177 | 0.6956 | 0.0245 | 0.0047 | 0.0124 | 0.5101 | 0.0267 |
| GRU4REC | 0.0050 | 0.0189 | 0.0693 | 0.1223 | 0.0045 | 0.0118 | 0.0512 | 0.1326 |
| S-SKNN | 0.0048 | 0.0293 | 0.4509 | 0.0807 | 0.0040 | 0.0182 | 0.2741 | 0.0961 |
| MC | 0.0046 | 0.0099 | 0.3496 | 0.0321 | 0.0045 | 0.0079 | 0.2756 | 0.0402 |
| BPR-MF | 0.0002 | 0.0004 | 0.6138 | 0.0088 | 0.0002 | 0.0003 | 0.4253 | 0.0131 |
| FOSSIL | 0.0001 | 0.0002 | 0.6703 | 0.0084 | 0.0001 | 0.0002 | 0.4941 | 0.0121 |
| FPMC | 0.0000 | 0.0001 | 0.7608 | 0.0031 | 0.0000 | 0.0001 | 0.5729 | 0.0035 |
| FISM | 0.0000 | 0.0001 | 0.6210 | 0.0027 | 0.0000 | 0.0000 | 0.4460 | 0.0028 |
| Algorithm | MRR@5 | HR@5 | COV@5 | POP@5 | MRR@3 | HR@3 | MRR@1 | HR@1 |
| AR | 0.0057 | 0.0108 | 0.1998 | 0.1251 | 0.0049 | 0.0073 | 0.0032 | 0.0032 |
| SMF | 0.0051 | 0.0092 | 0.0753 | 0.0958 | 0.0045 | 0.0064 | 0.0031 | 0.0031 |
| SR | 0.0056 | 0.0090 | 0.2395 | 0.0744 | 0.0051 | 0.0069 | 0.0038 | 0.0038 |
| SF-SKNN | 0.0060 | 0.0083 | 0.1794 | 0.0690 | 0.0057 | 0.0071 | 0.0047 | 0.0047 |
| V-SKNN | 0.0034 | 0.0102 | 0.1496 | 0.1364 | 0.0022 | 0.0048 | 0.0004 | 0.0004 |
| S-KNN | 0.0026 | 0.0079 | 0.0975 | 0.1065 | 0.0016 | 0.0033 | 0.0004 | 0.0004 |
| IKNN | 0.0041 | 0.0078 | 0.3293 | 0.0286 | 0.0035 | 0.0053 | 0.0022 | 0.0022 |
| GRU4REC | 0.0039 | 0.0070 | 0.0385 | 0.1414 | 0.0034 | 0.0050 | 0.0023 | 0.0023 |
| S-SKNN | 0.0026 | 0.0081 | 0.1404 | 0.1028 | 0.0016 | 0.0034 | 0.0004 | 0.0004 |
| MC | 0.0043 | 0.0062 | 0.2022 | 0.0493 | 0.0040 | 0.0051 | 0.0032 | 0.0032 |
| BPR-MF | 0.0002 | 0.0003 | 0.2576 | 0.0203 | 0.0002 | 0.0002 | 0.0001 | 0.0001 |
| FOSSIL | 0.0001 | 0.0001 | 0.3338 | 0.0178 | 0.0001 | 0.0001 | 0.0001 | 0.0001 |
| FPMC | 0.0000 | 0.0001 | 0.3879 | 0.0043 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| FISM | 0.0000 | 0.0000 | 0.2952 | 0.0030 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |

Table 19. Hit rate (HR), Mean reciprocal rank (MRR), item coverage (COV), and average popularity (POP) results for a list length of 20, 10, 5, 3, and 1 on the AOTM dataset (sorted by MRR@20).

| Algorithm | MRR@20 | HR@20 | COV@20 | POP@20 | MRR@10 | HR@10 | COV@10 | POP@10 |
|---|---|---|---|---|---|---|---|---|
| SMF | 0.0111 | 0.0298 | 0.2457 | 0.1998 | 0.0105 | 0.0205 | 0.1795 | 0.2085 |
| SF-SKNN | 0.0111 | 0.0145 | 0.3559 | 0.0508 | 0.0110 | 0.0139 | 0.3022 | 0.0686 |
| SR | 0.0077 | 0.0195 | 0.5864 | 0.0533 | 0.0073 | 0.0149 | 0.4481 | 0.0599 |
| GRU4REC | 0.0072 | 0.0157 | 0.4653 | 0.1151 | 0.0070 | 0.0125 | 0.3550 | 0.1131 |
| MC | 0.0063 | 0.0133 | 0.3803 | 0.0498 | 0.0062 | 0.0112 | 0.3198 | 0.0602 |
| AR | 0.0059 | 0.0233 | 0.5532 | 0.1049 | 0.0053 | 0.0146 | 0.4003 | 0.1178 |
| V-SKNN | 0.0055 | 0.0378 | 0.5363 | 0.1397 | 0.0043 | 0.0208 | 0.3357 | 0.1550 |
| S-SKNN | 0.0055 | 0.0397 | 0.5357 | 0.1289 | 0.0042 | 0.0209 | 0.3228 | 0.1475 |
| S-KNN | 0.0054 | 0.0429 | 0.2802 | 0.1678 | 0.0038 | 0.0200 | 0.1785 | 0.1666 |
| IKNN | 0.0049 | 0.0187 | 0.7880 | 0.0473 | 0.0045 | 0.0122 | 0.5777 | 0.0481 |
| FOSSIL | 0.0007 | 0.0027 | 0.5529 | 0.0978 | 0.0006 | 0.0017 | 0.3717 | 0.1139 |
| BPR-MF | 0.0005 | 0.0018 | 0.5659 | 0.0968 | 0.0005 | 0.0012 | 0.3550 | 0.1180 |
| FPMC | 0.0003 | 0.0007 | 0.7851 | 0.0264 | 0.0003 | 0.0006 | 0.5867 | 0.0289 |
| FISM | 0.0001 | 0.0004 | 0.6172 | 0.0272 | 0.0001 | 0.0002 | 0.4296 | 0.0288 |
| Algorithm | MRR@5 | HR@5 | COV@5 | POP@5 | MRR@3 | HR@3 | MRR@1 | HR@1 |
| SMF | 0.0097 | 0.0149 | 0.1265 | 0.2136 | 0.0091 | 0.0118 | 0.0070 | 0.0070 |
| SF-SKNN | 0.0109 | 0.0130 | 0.2306 | 0.0875 | 0.0107 | 0.0121 | 0.0096 | 0.0096 |
| SR | 0.0068 | 0.0107 | 0.3015 | 0.0626 | 0.0062 | 0.0081 | 0.0047 | 0.0047 |
| GRU4REC | 0.0067 | 0.0102 | 0.2432 | 0.1141 | 0.0063 | 0.0085 | 0.0045 | 0.0045 |
| MC | 0.0059 | 0.0089 | 0.2449 | 0.0681 | 0.0055 | 0.0072 | 0.0042 | 0.0042 |
| AR | 0.0046 | 0.0089 | 0.2543 | 0.1318 | 0.0039 | 0.0059 | 0.0024 | 0.0024 |
| V-SKNN | 0.0027 | 0.0085 | 0.1927 | 0.1592 | 0.0016 | 0.0034 | 0.0004 | 0.0004 |
| S-SKNN | 0.0025 | 0.0077 | 0.1718 | 0.1558 | 0.0014 | 0.0028 | 0.0005 | 0.0005 |
| S-KNN | 0.0021 | 0.0063 | 0.1108 | 0.1549 | 0.0011 | 0.0022 | 0.0005 | 0.0005 |
| IKNN | 0.0038 | 0.0073 | 0.3591 | 0.0500 | 0.0033 | 0.0051 | 0.0020 | 0.0020 |
| FOSSIL | 0.0005 | 0.0010 | 0.2311 | 0.1308 | 0.0005 | 0.0007 | 0.0003 | 0.0003 |
| BPR-MF | 0.0004 | 0.0007 | 0.1900 | 0.1403 | 0.0004 | 0.0005 | 0.0003 | 0.0003 |
| FPMC | 0.0003 | 0.0004 | 0.3884 | 0.0325 | 0.0003 | 0.0003 | 0.0003 | 0.0003 |
| FISM | 0.0001 | 0.0001 | 0.2743 | 0.0311 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |

Table 20. Hit rate (HR), Mean reciprocal rank (MRR), item coverage (COV), and average popularity (POP) results for a list length of 20, 10, 5, 3, and 1 on the 30MUSIC dataset (sorted by MRR@20).

| Algorithm | MRR@20 | HR@20 | COV@20 | POP@20 | MRR@10 | HR@10 | COV@10 | POP@10 |
|---|---|---|---|---|---|---|---|---|
| SR | 0.2377 | 0.3323 | 0.3893 | 0.0232 | 0.2363 | 0.3120 | 0.2913 | 0.0273 |
| MC | 0.2318 | 0.2844 | 0.2038 | 0.0205 | 0.2314 | 0.2780 | 0.1804 | 0.0265 |
| GRU4REC | 0.2264 | 0.3257 | 0.3447 | 0.0556 | 0.2249 | 0.3042 | 0.2423 | 0.0567 |
| SF-SKNN | 0.2079 | 0.2856 | 0.1854 | 0.0219 | 0.2078 | 0.2834 | 0.1634 | 0.0302 |
| SMF | 0.1777 | 0.2843 | 0.1508 | 0.1048 | 0.1756 | 0.2547 | 0.1117 | 0.1062 |
| V-SKNN | 0.1099 | 0.3819 | 0.3170 | 0.0538 | 0.1040 | 0.3002 | 0.1944 | 0.0573 |
| IKNN | 0.1086 | 0.2971 | 0.4596 | 0.0226 | 0.1053 | 0.2501 | 0.3122 | 0.0249 |
| S-SKNN | 0.1077 | 0.3856 | 0.2931 | 0.0515 | 0.1014 | 0.2975 | 0.1759 | 0.0569 |
| AR | 0.0960 | 0.3088 | 0.3524 | 0.0394 | 0.0911 | 0.2395 | 0.2375 | 0.0433 |
| S-KNN | 0.0898 | 0.3443 | 0.1912 | 0.0574 | 0.0832 | 0.2501 | 0.1155 | 0.0637 |
| BPR-MF | 0.0427 | 0.0580 | 0.4521 | 0.0281 | 0.0425 | 0.0548 | 0.2792 | 0.0381 |
| FPMC | 0.0293 | 0.0359 | 0.6544 | 0.0078 | 0.0291 | 0.0334 | 0.4556 | 0.0085 |
| FISM | 0.0029 | 0.0047 | 0.4676 | 0.0084 | 0.0028 | 0.0038 | 0.3052 | 0.0089 |
| FOSSIL | 0.0029 | 0.0100 | 0.3347 | 0.0297 | 0.0027 | 0.0073 | 0.1919 | 0.0436 |
| Algorithm | MRR@5 | HR@5 | COV@5 | POP@5 | MRR@3 | HR@3 | MRR@1 | HR@1 |
| SR | 0.2326 | 0.2845 | 0.1920 | 0.0294 | 0.2270 | 0.2601 | 0.2005 | 0.2005 |
| MC | 0.2298 | 0.2663 | 0.1467 | 0.0309 | 0.2270 | 0.2544 | 0.2043 | 0.2043 |
| GRU4REC | 0.2215 | 0.2796 | 0.1629 | 0.0557 | 0.2162 | 0.2564 | 0.1835 | 0.1835 |
| SF-SKNN | 0.2062 | 0.2726 | 0.1318 | 0.0371 | 0.2009 | 0.2499 | 0.1614 | 0.1614 |
| SMF | 0.1712 | 0.2223 | 0.0817 | 0.1057 | 0.1655 | 0.1970 | 0.1405 | 0.1405 |
| V-SKNN | 0.0882 | 0.1813 | 0.1165 | 0.0612 | 0.0727 | 0.1125 | 0.0442 | 0.0442 |
| IKNN | 0.0956 | 0.1788 | 0.1961 | 0.0248 | 0.0837 | 0.1265 | 0.0523 | 0.0523 |
| S-SKNN | 0.0851 | 0.1753 | 0.1043 | 0.0629 | 0.0701 | 0.1086 | 0.0428 | 0.0428 |
| AR | 0.0803 | 0.1580 | 0.1466 | 0.0476 | 0.0686 | 0.1063 | 0.0413 | 0.0413 |
| S-KNN | 0.0689 | 0.1424 | 0.0691 | 0.0714 | 0.0566 | 0.0877 | 0.0344 | 0.0344 |
| BPR-MF | 0.0421 | 0.0515 | 0.1523 | 0.0518 | 0.0414 | 0.0483 | 0.0356 | 0.0356 |
| FPMC | 0.0289 | 0.0317 | 0.2872 | 0.0096 | 0.0286 | 0.0303 | 0.0272 | 0.0272 |
| FISM | 0.0028 | 0.0034 | 0.1859 | 0.0095 | 0.0027 | 0.0030 | 0.0025 | 0.0025 |
| FOSSIL | 0.0023 | 0.0048 | 0.0914 | 0.0634 | 0.0019 | 0.0031 | 0.0011 | 0.0011 |

Table 21. Hit rate (HR), Mean reciprocal rank (MRR), item coverage (COV), and average popularity (POP) results for a list length of 20, 10, 5, 3, and 1 on the NOWPLAYING dataset (sorted by MRR@20).

| Algorithm | MRR@20 | HR@20 | COV@20 | POP@20 | MRR@10 | HR@10 | COV@10 | POP@10 |
|---|---|---|---|---|---|---|---|---|
| SR | 0.1053 | 0.2033 | 0.4656 | 0.0247 | 0.1031 | 0.1712 | 0.3605 | 0.0284 |
| GRU4REC | 0.1018 | 0.1970 | 0.4331 | 0.0516 | 0.0995 | 0.1632 | 0.3261 | 0.0529 |
| MC | 0.0971 | 0.1582 | 0.2936 | 0.0284 | 0.0960 | 0.1417 | 0.2547 | 0.0347 |
| SF-SKNN | 0.0954 | 0.1647 | 0.2773 | 0.0311 | 0.0945 | 0.1524 | 0.2369 | 0.0414 |
| SMF | 0.0882 | 0.1825 | 0.2417 | 0.0916 | 0.0859 | 0.1484 | 0.1847 | 0.0960 |
| V-SKNN | 0.0785 | 0.2552 | 0.4283 | 0.0639 | 0.0737 | 0.1861 | 0.2904 | 0.0719 |
| S-SKNN | 0.0777 | 0.2622 | 0.4149 | 0.0624 | 0.0725 | 0.1880 | 0.2727 | 0.0699 |
| AR | 0.0710 | 0.2076 | 0.4531 | 0.0511 | 0.0672 | 0.1518 | 0.3261 | 0.0584 |
| S-KNN | 0.0689 | 0.2429 | 0.3007 | 0.0690 | 0.0637 | 0.1676 | 0.1962 | 0.0759 |
| IKNN | 0.0569 | 0.1822 | 0.5799 | 0.0294 | 0.0534 | 0.1321 | 0.4313 | 0.0308 |
| BPR-MF | 0.0392 | 0.0621 | 0.5903 | 0.0672 | 0.0387 | 0.0547 | 0.3764 | 0.0843 |
| FPMC | 0.0331 | 0.0470 | 0.7865 | 0.0154 | 0.0327 | 0.0418 | 0.5833 | 0.0190 |
| FOSSIL | 0.0136 | 0.0432 | 0.5950 | 0.0336 | 0.0127 | 0.0302 | 0.4035 | 0.0389 |
| FISM | 0.0108 | 0.0183 | 0.6451 | 0.0110 | 0.0105 | 0.0145 | 0.4551 | 0.0123 |
| Algorithm | MRR@5 | HR@5 | COV@5 | POP@5 | MRR@3 | HR@3 | MRR@1 | HR@1 |
| SR | 0.0988 | 0.1395 | 0.2490 | 0.0305 | 0.0938 | 0.1173 | 0.0760 | 0.0760 |
| GRU4REC | 0.0958 | 0.1352 | 0.2282 | 0.0540 | 0.0913 | 0.1154 | 0.0726 | 0.0726 |
| MC | 0.0935 | 0.1236 | 0.2024 | 0.0409 | 0.0904 | 0.1099 | 0.0751 | 0.0751 |
| SF-SKNN | 0.0921 | 0.1344 | 0.1856 | 0.0518 | 0.0876 | 0.1149 | 0.0665 | 0.0665 |
| SMF | 0.0818 | 0.1181 | 0.1358 | 0.0985 | 0.0774 | 0.0985 | 0.0614 | 0.0614 |
| V-SKNN | 0.0651 | 0.1213 | 0.1819 | 0.0786 | 0.0562 | 0.0819 | 0.0381 | 0.0381 |
| S-SKNN | 0.0632 | 0.1181 | 0.1657 | 0.0771 | 0.0544 | 0.0790 | 0.0371 | 0.0371 |
| AR | 0.0611 | 0.1060 | 0.2114 | 0.0672 | 0.0543 | 0.0763 | 0.0379 | 0.0379 |
| S-KNN | 0.0556 | 0.1048 | 0.1226 | 0.0825 | 0.0476 | 0.0695 | 0.0318 | 0.0318 |
| IKNN | 0.0477 | 0.0884 | 0.2869 | 0.0317 | 0.0416 | 0.0615 | 0.0265 | 0.0265 |
| BPR-MF | 0.0378 | 0.0482 | 0.2043 | 0.0995 | 0.0367 | 0.0433 | 0.0314 | 0.0314 |
| FPMC | 0.0321 | 0.0371 | 0.3807 | 0.0238 | 0.0316 | 0.0346 | 0.0293 | 0.0293 |
| FOSSIL | 0.0113 | 0.0195 | 0.2502 | 0.0439 | 0.0102 | 0.0148 | 0.0069 | 0.0069 |
| FISM | 0.0102 | 0.0122 | 0.2934 | 0.0143 | 0.0100 | 0.0111 | 0.0092 | 0.0092 |

Table 22. Hit rate (HR), Mean reciprocal rank (MRR), item coverage (COV), and average popularity (POP) results for a list length of 20, 10, 5, 3, and 1 on the CLEF dataset (sorted by MRR@20).

| Algorithm | MRR@20 | HR@20 | COV@20 | POP@20 | MRR@10 | HR@10 | COV@10 | POP@10 |
|---|---|---|---|---|---|---|---|---|
| SMF | 0.234 | 0.706 | 0.650 | 0.083 | 0.222 | 0.529 | 0.582 | 0.097 |
| MC | 0.225 | 0.687 | 0.732 | 0.095 | 0.213 | 0.514 | 0.705 | 0.123 |
| V-SKNN | 0.224 | 0.776 | 0.621 | 0.082 | 0.211 | 0.596 | 0.566 | 0.113 |
| SR | 0.223 | 0.672 | 0.655 | 0.093 | 0.212 | 0.513 | 0.608 | 0.123 |
| GRU4REC | 0.220 | 0.568 | 0.174 | 0.094 | 0.212 | 0.462 | 0.129 | 0.118 |
| S-KNN | 0.219 | 0.778 | 0.613 | 0.084 | 0.205 | 0.588 | 0.545 | 0.122 |
| AR | 0.216 | 0.666 | 0.724 | 0.100 | 0.204 | 0.490 | 0.656 | 0.148 |
| IKNN | 0.188 | 0.596 | 0.746 | 0.059 | 0.177 | 0.436 | 0.722 | 0.047 |
| FPMC | 0.171 | 0.598 | 0.847 | 0.082 | 0.159 | 0.414 | 0.721 | 0.093 |
| FOSSIL | 0.166 | 0.571 | 0.963 | 0.079 | 0.155 | 0.417 | 0.864 | 0.093 |
| FISM | 0.129 | 0.403 | 0.997 | 0.080 | 0.122 | 0.297 | 0.963 | 0.108 |
| Algorithm | MRR@5 | HR@5 | COV@5 | POP@5 | MRR@3 | HR@3 | MRR@1 | HR@1 |
| SMF | 0.198 | 0.354 | 0.511 | 0.109 | 0.176 | 0.255 | 0.117 | 0.117 |
| MC | 0.190 | 0.339 | 0.653 | 0.144 | 0.170 | 0.249 | 0.113 | 0.113 |
| V-SKNN | 0.185 | 0.404 | 0.495 | 0.154 | 0.154 | 0.264 | 0.076 | 0.076 |
| SR | 0.189 | 0.337 | 0.542 | 0.146 | 0.169 | 0.246 | 0.111 | 0.111 |
| GRU4REC | 0.195 | 0.331 | 0.101 | 0.138 | 0.177 | 0.252 | 0.118 | 0.118 |
| S-KNN | 0.179 | 0.394 | 0.476 | 0.164 | 0.148 | 0.254 | 0.070 | 0.070 |
| AR | 0.183 | 0.339 | 0.559 | 0.215 | 0.161 | 0.242 | 0.102 | 0.102 |
| IKNN | 0.159 | 0.300 | 0.669 | 0.046 | 0.138 | 0.209 | 0.084 | 0.084 |
| FPMC | 0.138 | 0.258 | 0.601 | 0.108 | 0.120 | 0.178 | 0.078 | 0.078 |
| FOSSIL | 0.136 | 0.268 | 0.703 | 0.101 | 0.119 | 0.195 | 0.064 | 0.064 |
| FISM | 0.109 | 0.201 | 0.857 | 0.140 | 0.096 | 0.142 | 0.064 | 0.064 |

## C ADDITIONAL RESULTS FOR PRECISION AND RECALL

Table 23. Precision (P) and Recall (R) results for a list length of 20, 10, 5, and 3 on the TMALL dataset (sorted by P@20).

| Algorithm | P@20 | R@20 | P@10 | R@10 | P@5 | R@5 | P@3 | R@3 |
|---|---|---|---|---|---|---|---|---|
| S-KNN | 0.095 | 0.312 | 0.141 | 0.257 | 0.196 | 0.199 | 0.235 | 0.156 |
| S-SKNN | 0.094 | 0.263 | 0.139 | 0.215 | 0.191 | 0.165 | 0.229 | 0.129 |
| V-SKNN | 0.091 | 0.291 | 0.131 | 0.239 | 0.186 | 0.187 | 0.230 | 0.150 |
| SMF | 0.068 | 0.230 | 0.099 | 0.184 | 0.139 | 0.141 | 0.172 | 0.113 |
| GRU4REC | 0.068 | 0.233 | 0.098 | 0.187 | 0.137 | 0.143 | 0.170 | 0.115 |
| AR | 0.057 | 0.173 | 0.082 | 0.138 | 0.115 | 0.106 | 0.143 | 0.085 |
| SR | 0.052 | 0.193 | 0.081 | 0.162 | 0.121 | 0.131 | 0.158 | 0.109 |
| IKNN | 0.043 | 0.112 | 0.059 | 0.082 | 0.077 | 0.057 | 0.091 | 0.042 |
| SF-SKNN | 0.041 | 0.136 | 0.072 | 0.125 | 0.116 | 0.108 | 0.154 | 0.092 |
| MC | 0.036 | 0.124 | 0.058 | 0.107 | 0.090 | 0.089 | 0.119 | 0.075 |
| BPR-MF | 0.027 | 0.113 | 0.050 | 0.108 | 0.092 | 0.102 | 0.142 | 0.097 |
| FPMC | 0.015 | 0.078 | 0.028 | 0.073 | 0.050 | 0.068 | 0.077 | 0.064 |
| FISM | 0.009 | 0.046 | 0.015 | 0.042 | 0.027 | 0.038 | 0.040 | 0.035 |
| FOSSIL | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |

Table 24. Precision (P) and Recall (R) results for a list length of 20, 10, 5, and 3 on the RETAILROCKET dataset (sorted by P@20).

| Algorithm | P@20 | R@20 | P@10 | R@10 | P@5 | R@5 | P@3 | R@3 |
|---|---|---|---|---|---|---|---|---|
| S-SKNN | 0.057 | 0.480 | 0.096 | 0.434 | 0.156 | 0.376 | 0.214 | 0.326 |
| S-KNN | 0.056 | 0.478 | 0.095 | 0.433 | 0.152 | 0.373 | 0.208 | 0.323 |
| V-SKNN | 0.055 | 0.462 | 0.093 | 0.417 | 0.152 | 0.364 | 0.209 | 0.316 |
| SMF | 0.047 | 0.397 | 0.074 | 0.335 | 0.113 | 0.271 | 0.148 | 0.219 |
| GRU4REC | 0.046 | 0.400 | 0.073 | 0.339 | 0.111 | 0.271 | 0.143 | 0.215 |
| AR | 0.041 | 0.360 | 0.068 | 0.318 | 0.109 | 0.268 | 0.147 | 0.225 |
| SR | 0.038 | 0.342 | 0.067 | 0.313 | 0.110 | 0.269 | 0.149 | 0.227 |
| MC | 0.030 | 0.284 | 0.056 | 0.271 | 0.097 | 0.242 | 0.136 | 0.210 |
| SF-SKNN | 0.030 | 0.285 | 0.057 | 0.280 | 0.105 | 0.263 | 0.156 | 0.240 |
| BPR-MF | 0.029 | 0.286 | 0.056 | 0.282 | 0.107 | 0.277 | 0.172 | 0.272 |
| IKNN | 0.026 | 0.199 | 0.042 | 0.167 | 0.061 | 0.129 | 0.077 | 0.103 |
| FPMC | 0.023 | 0.253 | 0.043 | 0.246 | 0.083 | 0.239 | 0.133 | 0.233 |
| FOSSIL | 0.006 | 0.057 | 0.009 | 0.045 | 0.012 | 0.032 | 0.016 | 0.025 |
| FISM | 0.005 | 0.059 | 0.008 | 0.045 | 0.011 | 0.034 | 0.014 | 0.027 |

Table 25. Precision (P) and Recall (R) results for a list length of 20, 10, 5, and 3 on the ZALANDO dataset (sorted by P@20).

| Algorithm | P@20 | R@20 | P@10 | R@10 | P@5 | R@5 | P@3 | R@3 |
|---|---|---|---|---|---|---|---|---|
| V-SKNN | 0.076 | 0.219 | 0.129 | 0.195 | 0.210 | 0.168 | 0.282 | 0.141 |
| S-SKNN | 0.075 | 0.217 | 0.127 | 0.192 | 0.198 | 0.159 | 0.252 | 0.127 |
| S-KNN | 0.074 | 0.202 | 0.115 | 0.169 | 0.164 | 0.130 | 0.196 | 0.098 |
| GRU4REC | 0.065 | 0.181 | 0.109 | 0.161 | 0.181 | 0.141 | 0.247 | 0.120 |
| SMF | 0.062 | 0.175 | 0.103 | 0.154 | 0.169 | 0.133 | 0.228 | 0.112 |
| AR | 0.060 | 0.179 | 0.105 | 0.161 | 0.178 | 0.141 | 0.241 | 0.118 |
| SR | 0.060 | 0.174 | 0.106 | 0.161 | 0.185 | 0.146 | 0.260 | 0.127 |
| MC | 0.054 | 0.167 | 0.100 | 0.157 | 0.176 | 0.142 | 0.246 | 0.123 |
| SF-SKNN | 0.053 | 0.165 | 0.101 | 0.160 | 0.182 | 0.147 | 0.254 | 0.129 |
| IKNN | 0.045 | 0.134 | 0.080 | 0.122 | 0.137 | 0.109 | 0.196 | 0.096 |
| BPR-MF | 0.026 | 0.089 | 0.048 | 0.084 | 0.088 | 0.078 | 0.133 | 0.072 |
| FPMC | 0.016 | 0.060 | 0.029 | 0.055 | 0.051 | 0.051 | 0.078 | 0.047 |
| FOSSIL | 0.009 | 0.026 | 0.013 | 0.020 | 0.018 | 0.015 | 0.023 | 0.011 |
| FISM | 0.007 | 0.028 | 0.013 | 0.024 | 0.021 | 0.021 | 0.031 | 0.018 |

Table 26. Precision (P) and Recall (R) results for a list length of 20, 10, 5, and 3 on the 8TRACKS dataset (sorted by P@20).

| Algorithm | P@20 | R@20 | P@10 | R@10 | P@5 | R@5 | P@3 | R@3 |
|---|---|---|---|---|---|---|---|---|
| V-SKNN | 0.0122 | 0.0308 | 0.0138 | 0.0184 | 0.0125 | 0.0084 | 0.0109 | 0.0047 |
| S-KNN | 0.0117 | 0.0313 | 0.0119 | 0.0171 | 0.0090 | 0.0067 | 0.0059 | 0.0026 |
| S-SKNN | 0.0100 | 0.0266 | 0.0114 | 0.0162 | 0.0097 | 0.0070 | 0.0065 | 0.0029 |
| AR | 0.0087 | 0.0219 | 0.0112 | 0.0147 | 0.0136 | 0.0090 | 0.0155 | 0.0062 |
| SMF | 0.0086 | 0.0218 | 0.0104 | 0.0135 | 0.0118 | 0.0078 | 0.0126 | 0.0052 |
| IKNN | 0.0060 | 0.0149 | 0.0077 | 0.0102 | 0.0092 | 0.0063 | 0.0103 | 0.0043 |
| SR | 0.0055 | 0.0140 | 0.0077 | 0.0099 | 0.0098 | 0.0067 | 0.0111 | 0.0046 |
| GRU4REC | 0.0037 | 0.0095 | 0.0046 | 0.0061 | 0.0056 | 0.0039 | 0.0064 | 0.0027 |
| SF-SKNN | 0.0032 | 0.0081 | 0.0052 | 0.0067 | 0.0079 | 0.0052 | 0.0104 | 0.0042 |
| MC | 0.0025 | 0.0064 | 0.0036 | 0.0048 | 0.0051 | 0.0035 | 0.0065 | 0.0028 |
| FOSSIL | 0.0021 | 0.0043 | 0.0024 | 0.0025 | 0.0027 | 0.0013 | 0.0029 | 0.0009 |
| BPR-MF | 0.0018 | 0.0037 | 0.0021 | 0.0020 | 0.0022 | 0.0011 | 0.0019 | 0.0006 |
| FPMC | 0.0005 | 0.0012 | 0.0008 | 0.0009 | 0.0012 | 0.0007 | 0.0016 | 0.0006 |
| FISM | 0.0004 | 0.0008 | 0.0004 | 0.0005 | 0.0004 | 0.0002 | 0.0005 | 0.0002 |

Table 27. Precision (P) and Recall (R) results for a list length of 20, 10, 5, and 3 on the AOTM dataset (sorted by P@20).

| Algorithm | P@20 | R@20 | P@10 | R@10 | P@5 | R@5 | P@3 | R@3 |
|---|---|---|---|---|---|---|---|---|
| S-KNN | 0.0155 | 0.0440 | 0.0157 | 0.0214 | 0.0099 | 0.0058 | 0.0056 | 0.0023 |
| V-SKNN | 0.0133 | 0.0361 | 0.0145 | 0.0196 | 0.0118 | 0.0078 | 0.0106 | 0.0056 |
| S-SKNN | 0.0125 | 0.0353 | 0.0122 | 0.0178 | 0.0084 | 0.0065 | 0.0054 | 0.0027 |
| SMF | 0.0084 | 0.0259 | 0.0105 | 0.0163 | 0.0130 | 0.0104 | 0.0143 | 0.0070 |
| AR | 0.0066 | 0.0183 | 0.0082 | 0.0119 | 0.0095 | 0.0068 | 0.0107 | 0.0049 |
| IKNN | 0.0056 | 0.0155 | 0.0069 | 0.0102 | 0.0082 | 0.0062 | 0.0090 | 0.0043 |
| SR | 0.0053 | 0.0146 | 0.0070 | 0.0098 | 0.0082 | 0.0061 | 0.0092 | 0.0041 |
| SF-SKNN | 0.0024 | 0.0074 | 0.0043 | 0.0068 | 0.0075 | 0.0062 | 0.0108 | 0.0055 |
| MC | 0.0022 | 0.0069 | 0.0034 | 0.0056 | 0.0049 | 0.0043 | 0.0060 | 0.0033 |
| FOSSIL | 0.0012 | 0.0036 | 0.0013 | 0.0023 | 0.0015 | 0.0013 | 0.0017 | 0.0008 |
| GRU4REC | 0.0010 | 0.0027 | 0.0011 | 0.0015 | 0.0014 | 0.0008 | 0.0015 | 0.0005 |
| BPR-MF | 0.0004 | 0.0018 | 0.0005 | 0.0012 | 0.0005 | 0.0007 | 0.0005 | 0.0005 |
| FISM | 0.0004 | 0.0013 | 0.0005 | 0.0007 | 0.0005 | 0.0004 | 0.0004 | 0.0002 |
| FPMC | 0.0002 | 0.0007 | 0.0003 | 0.0006 | 0.0004 | 0.0004 | 0.0005 | 0.0004 |

Table 28. Precision (P) and Recall (R) results for a list length of 20, 10, 5, and 3 on the 30MUSIC dataset (sorted by P@20).

| Algorithm | P@20 | R@20 | P@10 | R@10 | P@5 | R@5 | P@3 | R@3 |
|---|---|---|---|---|---|---|---|---|
| V-SKNN | 0.1117 | 0.2438 | 0.1585 | 0.1948 | 0.1947 | 0.1371 | 0.2239 | 0.1044 |
| S-SKNN | 0.1110 | 0.2353 | 0.1439 | 0.1672 | 0.1431 | 0.0832 | 0.1344 | 0.0458 |
| S-KNN | 0.1035 | 0.2140 | 0.1295 | 0.1462 | 0.1283 | 0.0722 | 0.1216 | 0.0402 |
| IKNN | 0.0935 | 0.2023 | 0.1336 | 0.1611 | 0.1529 | 0.1015 | 0.1585 | 0.0651 |
| AR | 0.0914 | 0.1923 | 0.1244 | 0.1435 | 0.1354 | 0.0825 | 0.1386 | 0.0514 |
| SR | 0.0878 | 0.2010 | 0.1393 | 0.1750 | 0.1884 | 0.1353 | 0.2204 | 0.1042 |
| SMF | 0.0746 | 0.1655 | 0.1025 | 0.1272 | 0.1290 | 0.0876 | 0.1451 | 0.0626 |
| GRU4REC | 0.0404 | 0.0988 | 0.0627 | 0.0856 | 0.0932 | 0.0715 | 0.1236 | 0.0611 |
| SF-SKNN | 0.0319 | 0.0865 | 0.0591 | 0.0852 | 0.1027 | 0.0793 | 0.1447 | 0.0702 |
| MC | 0.0313 | 0.0852 | 0.0553 | 0.0811 | 0.0928 | 0.0743 | 0.1333 | 0.0679 |
| BPR-MF | 0.0172 | 0.0340 | 0.0290 | 0.0292 | 0.0442 | 0.0227 | 0.0569 | 0.0185 |
| FOSSIL | 0.0123 | 0.0188 | 0.0134 | 0.0117 | 0.0134 | 0.0055 | 0.0099 | 0.0027 |
| FPMC | 0.0046 | 0.0146 | 0.0079 | 0.0126 | 0.0137 | 0.0110 | 0.0205 | 0.0100 |
| FISM | 0.0015 | 0.0036 | 0.0019 | 0.0022 | 0.0025 | 0.0014 | 0.0025 | 0.0009 |

Table 29. Precision (P) and Recall (R) results for a list length of 20, 10, 5, and 3 on the NOWPLAYING dataset (sorted by P@20).

| Algorithm | P@20 | R@20 | P@10 | R@10 | P@5 | R@5 | P@3 | R@3 |
|---|---|---|---|---|---|---|---|---|
| S-SKNN | 0.0726 | 0.1944 | 0.0890 | 0.1296 | 0.0950 | 0.0694 | 0.0935 | 0.0405 |
| V-SKNN | 0.0718 | 0.1909 | 0.0900 | 0.1303 | 0.1048 | 0.0873 | 0.1169 | 0.0633 |
| S-KNN | 0.0680 | 0.1824 | 0.0841 | 0.1186 | 0.0868 | 0.0622 | 0.0890 | 0.0362 |
| AR | 0.0554 | 0.1551 | 0.0724 | 0.1086 | 0.0876 | 0.0705 | 0.0968 | 0.0491 |
| SR | 0.0501 | 0.1465 | 0.0717 | 0.1132 | 0.0945 | 0.0826 | 0.1080 | 0.0614 |
| SMF | 0.0499 | 0.1453 | 0.0668 | 0.1043 | 0.0840 | 0.0709 | 0.0966 | 0.0525 |
| IKNN | 0.0492 | 0.1385 | 0.0639 | 0.0974 | 0.0755 | 0.0608 | 0.0809 | 0.0405 |
| SF-SKNN | 0.0280 | 0.0903 | 0.0495 | 0.0816 | 0.0761 | 0.0660 | 0.0992 | 0.0539 |
| GRU4REC | 0.0272 | 0.0810 | 0.0383 | 0.0601 | 0.0523 | 0.0444 | 0.0636 | 0.0343 |
| MC | 0.0250 | 0.0845 | 0.0415 | 0.0724 | 0.0625 | 0.0573 | 0.0810 | 0.0473 |
| FOSSIL | 0.0169 | 0.0412 | 0.0229 | 0.0291 | 0.0308 | 0.0204 | 0.0359 | 0.0149 |
| BPR-MF | 0.0156 | 0.0393 | 0.0231 | 0.0328 | 0.0358 | 0.0275 | 0.0492 | 0.0240 |
| FPMC | 0.0061 | 0.0244 | 0.0102 | 0.0211 | 0.0171 | 0.0181 | 0.0248 | 0.0161 |
| FISM | 0.0023 | 0.0077 | 0.0033 | 0.0064 | 0.0051 | 0.0053 | 0.0072 | 0.0044 |

Table 30. Precision (P) and Recall (R) results for a list length of 20, 10, 5, and 3 on the CLEF dataset (sorted by P@20).

| Algorithm | P@20 | R@20 | P@10 | R@10 | P@5 | R@5 | P@3 | R@3 |
|---|---|---|---|---|---|---|---|---|
| GRU4REC | 0.072 | 0.626 | 0.100 | 0.454 | 0.128 | 0.298 | 0.144 | 0.204 |
| V-SKNN | 0.069 | 0.593 | 0.089 | 0.413 | 0.108 | 0.262 | 0.119 | 0.180 |
| S-SKNN | 0.066 | 0.579 | 0.086 | 0.404 | 0.097 | 0.244 | 0.101 | 0.154 |
| S-KNN | 0.066 | 0.577 | 0.085 | 0.399 | 0.096 | 0.241 | 0.099 | 0.152 |
| SF-SKNN | 0.064 | 0.565 | 0.082 | 0.390 | 0.095 | 0.241 | 0.098 | 0.151 |
| SMF | 0.062 | 0.527 | 0.084 | 0.377 | 0.107 | 0.246 | 0.120 | 0.167 |
| FPMC | 0.060 | 0.515 | 0.078 | 0.347 | 0.093 | 0.216 | 0.110 | 0.154 |
| MC | 0.059 | 0.510 | 0.081 | 0.368 | 0.107 | 0.251 | 0.119 | 0.170 |
| FOSSIL | 0.059 | 0.504 | 0.075 | 0.334 | 0.090 | 0.205 | 0.103 | 0.143 |
| AR | 0.058 | 0.506 | 0.080 | 0.364 | 0.101 | 0.239 | 0.115 | 0.170 |
| SR | 0.058 | 0.502 | 0.081 | 0.366 | 0.108 | 0.251 | 0.115 | 0.162 |
| FISM | 0.058 | 0.506 | 0.077 | 0.357 | 0.095 | 0.227 | 0.105 | 0.156 |
| IKNN | 0.050 | 0.418 | 0.065 | 0.290 | 0.086 | 0.197 | 0.099 | 0.141 |
| BPR-MF | 0.016 | 0.147 | 0.024 | 0.120 | 0.042 | 0.105 | 0.062 | 0.094 |

## D ADDITIONAL SINGLE SPLIT RESULTS

Table 31. Hit rate (HR), Mean reciprocal rank (MRR), item coverage (COV), and average popularity (POP) results for a list length of 20, 10, 5, 3, and 1 on the TMALL dataset with a single split (sorted by MRR@20).

| Algorithm | MRR@20 | HR@20 | COV@20 | POP@20 | MRR@10 | HR@10 | COV@10 | POP@10 |
|---|---|---|---|---|---|---|---|---|
| S-SKNN | 0.181 | 0.385 | 0.560 | 0.019 | 0.177 | 0.329 | 0.375 | 0.021 |
| S-KNN | 0.177 | 0.398 | 0.461 | 0.020 | 0.173 | 0.334 | 0.306 | 0.022 |
| V-SKNN | 0.169 | 0.353 | 0.570 | 0.019 | 0.166 | 0.298 | 0.405 | 0.021 |
| SF-SKNN | 0.141 | 0.237 | 0.577 | 0.015 | 0.140 | 0.220 | 0.441 | 0.018 |
| SMF | 0.140 | 0.298 | 0.461 | 0.023 | 0.136 | 0.245 | 0.341 | 0.023 |
| AR | 0.131 | 0.254 | 0.628 | 0.019 | 0.128 | 0.214 | 0.457 | 0.021 |
| SR | 0.131 | 0.243 | 0.683 | 0.019 | 0.129 | 0.209 | 0.507 | 0.020 |
| GRU4REC | 0.123 | 0.263 | 0.171 | 0.029 | 0.120 | 0.213 | 0.117 | 0.032 |
| MC | 0.123 | 0.214 | 0.673 | 0.018 | 0.121 | 0.188 | 0.516 | 0.019 |
| IKNN | 0.049 | 0.147 | 0.801 | 0.006 | 0.047 | 0.111 | 0.644 | 0.006 |
| Algorithm | MRR@5 | HR@5 | COV@5 | POP@5 | MRR@3 | HR@3 | MRR@1 | HR@1 |
| S-SKNN | 0.168 | 0.265 | 0.240 | 0.023 | 0.157 | 0.215 | 0.112 | 0.112 |
| S-KNN | 0.163 | 0.263 | 0.199 | 0.024 | 0.151 | 0.209 | 0.106 | 0.106 |
| V-SKNN | 0.158 | 0.241 | 0.282 | 0.023 | 0.148 | 0.199 | 0.109 | 0.109 |
| SF-SKNN | 0.136 | 0.194 | 0.309 | 0.020 | 0.131 | 0.168 | 0.101 | 0.101 |
| SMF | 0.129 | 0.195 | 0.244 | 0.024 | 0.122 | 0.160 | 0.092 | 0.092 |
| AR | 0.123 | 0.174 | 0.312 | 0.023 | 0.117 | 0.147 | 0.094 | 0.094 |
| SR | 0.124 | 0.173 | 0.348 | 0.021 | 0.118 | 0.148 | 0.095 | 0.095 |
| GRU4REC | 0.114 | 0.169 | 0.082 | 0.034 | 0.107 | 0.139 | 0.083 | 0.083 |
| MC | 0.117 | 0.160 | 0.361 | 0.021 | 0.113 | 0.139 | 0.092 | 0.092 |
| IKNN | 0.042 | 0.077 | 0.471 | 0.007 | 0.038 | 0.056 | 0.024 | 0.024 |

Table 32. Hit rate (HR), Mean reciprocal rank (MRR), item coverage (COV), and average popularity (POP) results for a list length of 20, 10, 5, 3, and 1 on the RETAILROCKET dataset with a single split (sorted by MRR@20).

| Algorithm | MRR@20 | HR@20 | COV@20 | POP@20 | MRR@10 | HR@10 | COV@10 | POP@10 |
|---|---|---|---|---|---|---|---|---|
| S-SKNN | 0.333 | 0.580 | 0.272 | 0.051 | 0.329 | 0.524 | 0.170 | 0.060 |
| S-KNN | 0.332 | 0.571 | 0.250 | 0.052 | 0.329 | 0.520 | 0.155 | 0.061 |
| V-SKNN | 0.327 | 0.582 | 0.272 | 0.061 | 0.323 | 0.520 | 0.172 | 0.069 |
| SF-SKNN | 0.301 | 0.463 | 0.224 | 0.036 | 0.299 | 0.444 | 0.168 | 0.049 |
| SR | 0.270 | 0.504 | 0.299 | 0.047 | 0.267 | 0.452 | 0.201 | 0.054 |
| SMF | 0.270 | 0.557 | 0.313 | 0.056 | 0.264 | 0.479 | 0.198 | 0.061 |
| AR | 0.265 | 0.485 | 0.286 | 0.058 | 0.261 | 0.425 | 0.189 | 0.064 |
| MC | 0.261 | 0.468 | 0.250 | 0.040 | 0.258 | 0.425 | 0.184 | 0.049 |
| GRU4REC | 0.260 | 0.559 | 0.291 | 0.055 | 0.254 | 0.475 | 0.187 | 0.062 |
| IKNN | 0.121 | 0.284 | 0.334 | 0.031 | 0.118 | 0.238 | 0.219 | 0.036 |
| Algorithm | MRR@5 | HR@5 | COV@5 | POP@5 | MRR@3 | HR@3 | MRR@1 | HR@1 |
| S-SKNN | 0.320 | 0.456 | 0.098 | 0.069 | 0.305 | 0.391 | 0.236 | 0.236 |
| S-KNN | 0.320 | 0.453 | 0.090 | 0.070 | 0.305 | 0.386 | 0.242 | 0.242 |
| V-SKNN | 0.313 | 0.446 | 0.104 | 0.075 | 0.298 | 0.383 | 0.230 | 0.230 |
| SF-SKNN | 0.293 | 0.399 | 0.110 | 0.060 | 0.281 | 0.346 | 0.230 | 0.230 |
| SR | 0.257 | 0.379 | 0.123 | 0.062 | 0.244 | 0.322 | 0.184 | 0.184 |
| SMF | 0.252 | 0.388 | 0.119 | 0.067 | 0.234 | 0.309 | 0.177 | 0.177 |
| AR | 0.253 | 0.366 | 0.115 | 0.074 | 0.240 | 0.310 | 0.187 | 0.187 |
| MC | 0.250 | 0.366 | 0.121 | 0.059 | 0.236 | 0.308 | 0.180 | 0.180 |
| GRU4REC | 0.240 | 0.377 | 0.114 | 0.069 | 0.223 | 0.300 | 0.164 | 0.164 |
| IKNN | 0.111 | 0.185 | 0.131 | 0.041 | 0.103 | 0.154 | 0.067 | 0.067 |

Table 33. Hit rate (HR), Mean reciprocal rank (MRR), item coverage (COV), and average popularity (POP) results for a list length of 20, 10, 5, 3, and 1 on the ZALANDO dataset with a single split (sorted by MRR@20).

| Algorithm | MRR@20 | HR@20 | COV@20 | POP@20 | MRR@10 | HR@10 | COV@10 | POP@10 |
|---|---|---|---|---|---|---|---|---|
| SR | 0.306 | 0.498 | 0.525 | 0.059 | 0.304 | 0.473 | 0.381 | 0.060 |
| MC | 0.304 | 0.469 | 0.504 | 0.052 | 0.302 | 0.452 | 0.381 | 0.055 |
| IKNN | 0.271 | 0.410 | 0.597 | 0.033 | 0.269 | 0.388 | 0.432 | 0.036 |
| GRU4REC | 0.267 | 0.483 | 0.290 | 0.073 | 0.265 | 0.442 | 0.223 | 0.074 |
| AR | 0.265 | 0.483 | 0.431 | 0.073 | 0.262 | 0.450 | 0.311 | 0.073 |
| SMF | 0.253 | 0.463 | 0.329 | 0.075 | 0.250 | 0.422 | 0.248 | 0.076 |
| SF-SKNN | 0.251 | 0.451 | 0.419 | 0.046 | 0.250 | 0.440 | 0.323 | 0.053 |
| V-SKNN | 0.237 | 0.517 | 0.396 | 0.077 | 0.234 | 0.478 | 0.275 | 0.076 |
| S-SKNN | 0.224 | 0.510 | 0.395 | 0.066 | 0.221 | 0.464 | 0.257 | 0.068 |
| S-KNN | 0.181 | 0.461 | 0.301 | 0.069 | 0.176 | 0.392 | 0.197 | 0.071 |
| Algorithm | MRR@5 | HR@5 | COV@5 | POP@5 | MRR@3 | HR@3 | MRR@1 | HR@1 |
| SR | 0.299 | 0.435 | 0.256 | 0.058 | 0.287 | 0.384 | 0.210 | 0.210 |
| MC | 0.298 | 0.422 | 0.262 | 0.056 | 0.288 | 0.379 | 0.216 | 0.216 |
| IKNN | 0.266 | 0.363 | 0.287 | 0.040 | 0.258 | 0.330 | 0.199 | 0.199 |
| GRU4REC | 0.258 | 0.394 | 0.167 | 0.073 | 0.245 | 0.336 | 0.174 | 0.174 |
| AR | 0.256 | 0.404 | 0.216 | 0.069 | 0.243 | 0.347 | 0.163 | 0.163 |
| SMF | 0.243 | 0.372 | 0.182 | 0.074 | 0.230 | 0.317 | 0.163 | 0.163 |
| SF-SKNN | 0.245 | 0.406 | 0.227 | 0.057 | 0.231 | 0.345 | 0.143 | 0.143 |
| V-SKNN | 0.226 | 0.419 | 0.185 | 0.071 | 0.208 | 0.343 | 0.103 | 0.103 |
| S-SKNN | 0.211 | 0.391 | 0.161 | 0.070 | 0.193 | 0.311 | 0.100 | 0.100 |
| S-KNN | 0.164 | 0.300 | 0.124 | 0.074 | 0.147 | 0.226 | 0.087 | 0.087 |

Table 34. Hit rate (HR), Mean reciprocal rank (MRR), item coverage (COV), and average popularity (POP) results for a list length of 20, 10, 5, 3, and 1 on the 8TRACKS dataset with a single split (sorted by MRR@20).

| Algorithm | MRR@20 | HR@20 | COV@20 | POP@20 | MRR@10 | HR@10 | COV@10 | POP@10 |
|---|---|---|---|---|---|---|---|---|
| AR | 0.0135 | 0.0410 | 0.7769 | 0.0399 | 0.0126 | 0.0276 | 0.5772 | 0.0545 |
| SR | 0.0123 | 0.0329 | 0.8875 | 0.0243 | 0.0116 | 0.0233 | 0.7261 | 0.0302 |
| SMF | 0.0115 | 0.0476 | 0.0772 | 0.1197 | 0.0102 | 0.0290 | 0.0556 | 0.1303 |
| V-SKNN | 0.0110 | 0.0490 | 0.7180 | 0.0290 | 0.0098 | 0.0313 | 0.5317 | 0.0322 |
| MC | 0.0101 | 0.0234 | 0.8365 | 0.0152 | 0.0098 | 0.0179 | 0.7050 | 0.0179 |
| S-KNN | 0.0098 | 0.0438 | 0.6122 | 0.0272 | 0.0086 | 0.0267 | 0.4343 | 0.0298 |
| S-SKNN | 0.0097 | 0.0402 | 0.8543 | 0.0197 | 0.0087 | 0.0265 | 0.6465 | 0.0238 |
| GRU4REC | 0.0095 | 0.0376 | 0.0593 | 0.1930 | 0.0085 | 0.0231 | 0.0445 | 0.2140 |
| SF-SKNN | 0.0089 | 0.0217 | 0.7713 | 0.0157 | 0.0086 | 0.0171 | 0.6555 | 0.0219 |
| IKNN | 0.0072 | 0.0251 | 0.9852 | 0.0063 | 0.0066 | 0.0165 | 0.8756 | 0.0069 |
| Algorithm | MRR@5 | HR@5 | COV@5 | POP@5 | MRR@3 | HR@3 | MRR@1 | HR@1 |
| AR | 0.0114 | 0.0185 | 0.3933 | 0.0774 | 0.0104 | 0.0140 | 0.0078 | 0.0078 |
| SR | 0.0107 | 0.0166 | 0.5237 | 0.0374 | 0.0099 | 0.0128 | 0.0078 | 0.0078 |
| SMF | 0.0087 | 0.0172 | 0.0406 | 0.1401 | 0.0073 | 0.0113 | 0.0044 | 0.0044 |
| V-SKNN | 0.0079 | 0.0167 | 0.4041 | 0.0324 | 0.0063 | 0.0097 | 0.0040 | 0.0040 |
| MC | 0.0092 | 0.0134 | 0.5477 | 0.0216 | 0.0086 | 0.0107 | 0.0070 | 0.0070 |
| S-KNN | 0.0068 | 0.0128 | 0.3056 | 0.0267 | 0.0055 | 0.0074 | 0.0044 | 0.0044 |
| S-SKNN | 0.0070 | 0.0137 | 0.3984 | 0.0256 | 0.0057 | 0.0079 | 0.0043 | 0.0043 |
| GRU4REC | 0.0073 | 0.0140 | 0.0336 | 0.2350 | 0.0062 | 0.0093 | 0.0040 | 0.0040 |
| SF-SKNN | 0.0080 | 0.0131 | 0.5072 | 0.0286 | 0.0074 | 0.0101 | 0.0052 | 0.0052 |
| IKNN | 0.0058 | 0.0107 | 0.6635 | 0.0075 | 0.0051 | 0.0075 | 0.0034 | 0.0034 |

Table 35. Hit rate (HR), Mean reciprocal rank (MRR), item coverage (COV), and average popularity (POP) results for a list length of 20, 10, 5, 3, and 1 on the AOTM dataset with a single split (sorted by MRR@20).

| Algorithm | MRR@20 | HR@20 | COV@20 | POP@20 | MRR@10 | HR@10 | COV@10 | POP@10 |
|---|---|---|---|---|---|---|---|---|
| SF-SKNN | 0.0275 | 0.0440 | 0.8591 | 0.0828 | 0.0273 | 0.0403 | 0.7673 | 0.1025 |
| SMF | 0.0204 | 0.0468 | 0.9262 | 0.0941 | 0.0197 | 0.0361 | 0.8294 | 0.0941 |
| GRU4REC | 0.0154 | 0.0427 | 0.6523 | 0.1665 | 0.0146 | 0.0312 | 0.5081 | 0.1759 |
| SR | 0.0152 | 0.0449 | 0.9439 | 0.1061 | 0.0143 | 0.0318 | 0.8140 | 0.1143 |
| MC | 0.0134 | 0.0348 | 0.8996 | 0.0813 | 0.0128 | 0.0262 | 0.7889 | 0.0902 |
| AR | 0.0119 | 0.0426 | 0.8523 | 0.1409 | 0.0109 | 0.0283 | 0.6723 | 0.1536 |
| V-SKNN | 0.0104 | 0.0721 | 0.7971 | 0.1567 | 0.0083 | 0.0415 | 0.6049 | 0.1662 |
| IKNN | 0.0100 | 0.0384 | 0.9854 | 0.0482 | 0.0090 | 0.0242 | 0.8660 | 0.0490 |
| S-SKNN | 0.0095 | 0.0737 | 0.8917 | 0.1192 | 0.0071 | 0.0406 | 0.6652 | 0.1322 |
| S-KNN | 0.0087 | 0.0740 | 0.6400 | 0.1414 | 0.0059 | 0.0345 | 0.4599 | 0.1416 |
| Algorithm | MRR@5 | HR@5 | COV@5 | POP@5 | MRR@3 | HR@3 | MRR@1 | HR@1 |
| SF-SKNN | 0.0267 | 0.0360 | 0.6205 | 0.1192 | 0.0257 | 0.0315 | 0.0212 | 0.0212 |
| SMF | 0.0186 | 0.0280 | 0.6845 | 0.0927 | 0.0175 | 0.0232 | 0.0132 | 0.0132 |
| GRU4REC | 0.0134 | 0.0225 | 0.3629 | 0.1840 | 0.0121 | 0.0166 | 0.0087 | 0.0087 |
| SR | 0.0130 | 0.0217 | 0.6154 | 0.1226 | 0.0118 | 0.0167 | 0.0083 | 0.0083 |
| MC | 0.0119 | 0.0198 | 0.6326 | 0.0985 | 0.0109 | 0.0151 | 0.0078 | 0.0078 |
| AR | 0.0095 | 0.0178 | 0.4853 | 0.1689 | 0.0083 | 0.0127 | 0.0052 | 0.0052 |
| V-SKNN | 0.0051 | 0.0174 | 0.4490 | 0.1646 | 0.0027 | 0.0067 | 0.0001 | 0.0001 |
| IKNN | 0.0079 | 0.0153 | 0.6629 | 0.0499 | 0.0067 | 0.0103 | 0.0040 | 0.0040 |
| S-SKNN | 0.0035 | 0.0124 | 0.4265 | 0.1392 | 0.0016 | 0.0040 | 0.0000 | 0.0000 |
| S-KNN | 0.0028 | 0.0096 | 0.3142 | 0.1387 | 0.0014 | 0.0033 | 0.0001 | 0.0001 |

Table 36. Hit rate (HR), Mean reciprocal rank (MRR), item coverage (COV), and average popularity (POP) results for a list length of 20, 10, 5, 3, and 1 on the 30MUSIC dataset with a single split (sorted by MRR@20).

| Algorithm | MRR@20 | HR@20 | COV@20 | POP@20 | MRR@10 | HR@10 | COV@10 | POP@10 |
|---|---|---|---|---|---|---|---|---|
| SR | 0.2690 | 0.3744 | 0.5904 | 0.0373 | 0.2672 | 0.3499 | 0.4619 | 0.0389 |
| MC | 0.2653 | 0.3302 | 0.4001 | 0.0283 | 0.2646 | 0.3203 | 0.3455 | 0.0340 |
| GRU4REC | 0.2354 | 0.3651 | 0.4029 | 0.0665 | 0.2334 | 0.3372 | 0.3077 | 0.0669 |
| SMF | 0.2145 | 0.3614 | 0.3974 | 0.0608 | 0.2121 | 0.3275 | 0.3013 | 0.0605 |
| SF-SKNN | 0.2123 | 0.3320 | 0.3404 | 0.0284 | 0.2118 | 0.3258 | 0.2929 | 0.0366 |
| IKNN | 0.1352 | 0.3412 | 0.6758 | 0.0219 | 0.1319 | 0.2943 | 0.5043 | 0.0234 |
| V-SKNN | 0.1192 | 0.4134 | 0.4384 | 0.0589 | 0.1130 | 0.3263 | 0.2911 | 0.0599 |
| AR | 0.1157 | 0.3518 | 0.5352 | 0.0437 | 0.1107 | 0.2810 | 0.3886 | 0.0456 |
| S-SKNN | 0.1153 | 0.4119 | 0.4195 | 0.0544 | 0.1087 | 0.3190 | 0.2681 | 0.0593 |
| S-KNN | 0.0938 | 0.3609 | 0.2848 | 0.0595 | 0.0869 | 0.2626 | 0.1799 | 0.0658 |
| Algorithm | MRR@5 | HR@5 | COV@5 | POP@5 | MRR@3 | HR@3 | MRR@1 | HR@1 |
| SR | 0.2629 | 0.3179 | 0.3270 | 0.0387 | 0.2572 | 0.2930 | 0.2282 | 0.2282 |
| MC | 0.2625 | 0.3051 | 0.2790 | 0.0374 | 0.2590 | 0.2900 | 0.2332 | 0.2332 |
| GRU4REC | 0.2286 | 0.3011 | 0.2289 | 0.0653 | 0.2212 | 0.2688 | 0.1830 | 0.1830 |
| SMF | 0.2064 | 0.2851 | 0.2200 | 0.0587 | 0.1981 | 0.2488 | 0.1583 | 0.1583 |
| SF-SKNN | 0.2083 | 0.3013 | 0.2364 | 0.0432 | 0.1987 | 0.2598 | 0.1507 | 0.1507 |
| IKNN | 0.1215 | 0.2176 | 0.3427 | 0.0227 | 0.1084 | 0.1600 | 0.0700 | 0.0700 |
| V-SKNN | 0.0958 | 0.1972 | 0.1927 | 0.0630 | 0.0785 | 0.1202 | 0.0494 | 0.0494 |
| AR | 0.0985 | 0.1905 | 0.2579 | 0.0493 | 0.0849 | 0.1304 | 0.0519 | 0.0519 |
| S-SKNN | 0.0913 | 0.1881 | 0.1700 | 0.0660 | 0.0745 | 0.1135 | 0.0471 | 0.0471 |
| S-KNN | 0.0719 | 0.1481 | 0.1129 | 0.0742 | 0.0589 | 0.0904 | 0.0367 | 0.0367 |

Table 37. Hit rate (HR), Mean reciprocal rank (MRR), item coverage (COV), and average popularity (POP) results for a list length of 20, 10, 5, 3, and 1 on the NOWPLAYING dataset with a single split (sorted by MRR@20).

| Algorithm | MRR@20 | HR@20 | COV@20 | POP@20 | MRR@10 | HR@10 | COV@10 | POP@10 |
|---|---|---|---|---|---|---|---|---|
| SR | 0.0856 | 0.1825 | 0.4629 | 0.0309 | 0.0831 | 0.1466 | 0.3390 | 0.0346 |
| MC | 0.0813 | 0.1474 | 0.3576 | 0.0248 | 0.0798 | 0.1255 | 0.2833 | 0.0287 |
| SF-SKNN | 0.0787 | 0.1602 | 0.3015 | 0.0264 | 0.0774 | 0.1431 | 0.2383 | 0.0333 |
| SMF | 0.0782 | 0.1881 | 0.3560 | 0.0322 | 0.0753 | 0.1454 | 0.2585 | 0.0332 |
| GRU4REC | 0.0771 | 0.1792 | 0.2202 | 0.0523 | 0.0742 | 0.1370 | 0.1647 | 0.0568 |
| V-SKNN | 0.0670 | 0.2291 | 0.3358 | 0.0445 | 0.0624 | 0.1627 | 0.2248 | 0.0497 |
| S-SKNN | 0.0669 | 0.2406 | 0.3436 | 0.0407 | 0.0618 | 0.1674 | 0.2186 | 0.0468 |
| AR | 0.0647 | 0.1866 | 0.4137 | 0.0381 | 0.0613 | 0.1378 | 0.2869 | 0.0441 |
| S-KNN | 0.0604 | 0.2241 | 0.2312 | 0.0453 | 0.0554 | 0.1512 | 0.1487 | 0.0509 |
| IKNN | 0.0467 | 0.1554 | 0.5526 | 0.0144 | 0.0437 | 0.1120 | 0.3960 | 0.0145 |
| Algorithm | MRR@5 | HR@5 | COV@5 | POP@5 | MRR@3 | HR@3 | MRR@1 | HR@1 |
| SR | 0.0789 | 0.1146 | 0.2290 | 0.0390 | 0.0744 | 0.0949 | 0.0584 | 0.0584 |
| MC | 0.0769 | 0.1041 | 0.2090 | 0.0336 | 0.0738 | 0.0903 | 0.0610 | 0.0610 |
| SF-SKNN | 0.0737 | 0.1157 | 0.1743 | 0.0403 | 0.0682 | 0.0914 | 0.0500 | 0.0500 |
| SMF | 0.0705 | 0.1085 | 0.1799 | 0.0341 | 0.0658 | 0.0879 | 0.0491 | 0.0491 |
| GRU4REC | 0.0700 | 0.1053 | 0.1196 | 0.0604 | 0.0651 | 0.0839 | 0.0510 | 0.0510 |
| V-SKNN | 0.0543 | 0.1019 | 0.1465 | 0.0543 | 0.0466 | 0.0677 | 0.0316 | 0.0316 |
| S-SKNN | 0.0532 | 0.1017 | 0.1329 | 0.0523 | 0.0450 | 0.0655 | 0.0306 | 0.0306 |
| AR | 0.0555 | 0.0947 | 0.1830 | 0.0531 | 0.0499 | 0.0701 | 0.0350 | 0.0350 |
| S-KNN | 0.0472 | 0.0888 | 0.0936 | 0.0560 | 0.0402 | 0.0578 | 0.0282 | 0.0282 |
| IKNN | 0.0384 | 0.0724 | 0.2597 | 0.0141 | 0.0332 | 0.0494 | 0.0214 | 0.0214 |

Table 38. Hit rate (HR), Mean reciprocal rank (MRR), item coverage (COV), and average popularity (POP) results for a list length of 20, 10, 5, 3, and 1 on the CLEF dataset with a single split (sorted by MRR@20).

| Algorithm | MRR@20 | HR@20 | COV@20 | POP@20 | MRR@10 | HR@10 | COV@10 | POP@10 |
|---|---|---|---|---|---|---|---|---|
| GRU4REC | 0.253 | 0.724 | 0.154 | 0.051 | 0.242 | 0.564 | 0.123 | 0.064 |
| SR | 0.213 | 0.623 | 0.754 | 0.055 | 0.202 | 0.451 | 0.715 | 0.084 |
| MC | 0.213 | 0.625 | 0.755 | 0.056 | 0.202 | 0.459 | 0.724 | 0.080 |
| SMF | 0.207 | 0.646 | 0.765 | 0.042 | 0.194 | 0.472 | 0.709 | 0.042 |
| V-SKNN | 0.197 | 0.683 | 0.756 | 0.053 | 0.183 | 0.487 | 0.682 | 0.080 |
| S-SKNN | 0.190 | 0.670 | 0.764 | 0.052 | 0.175 | 0.463 | 0.682 | 0.081 |
| S-KNN | 0.186 | 0.661 | 0.760 | 0.052 | 0.172 | 0.453 | 0.675 | 0.083 |
| SF-SKNN | 0.179 | 0.636 | 0.750 | 0.052 | 0.165 | 0.433 | 0.680 | 0.082 |
| AR | 0.178 | 0.631 | 0.733 | 0.058 | 0.164 | 0.435 | 0.653 | 0.103 |
| IKNN | 0.158 | 0.510 | 0.796 | 0.009 | 0.148 | 0.363 | 0.757 | 0.010 |
| Algorithm | MRR@5 | HR@5 | COV@5 | POP@5 | MRR@3 | HR@3 | MRR@1 | HR@1 |
| GRU4REC | 0.219 | 0.384 | 0.101 | 0.088 | 0.195 | 0.280 | 0.132 | 0.132 |
| SR | 0.183 | 0.308 | 0.636 | 0.110 | 0.167 | 0.238 | 0.113 | 0.113 |
| MC | 0.184 | 0.325 | 0.652 | 0.110 | 0.165 | 0.241 | 0.107 | 0.107 |
| SMF | 0.172 | 0.300 | 0.639 | 0.023 | 0.155 | 0.225 | 0.104 | 0.104 |
| V-SKNN | 0.160 | 0.316 | 0.585 | 0.082 | 0.137 | 0.215 | 0.081 | 0.081 |
| S-SKNN | 0.154 | 0.304 | 0.595 | 0.101 | 0.133 | 0.208 | 0.077 | 0.077 |
| S-KNN | 0.151 | 0.298 | 0.592 | 0.103 | 0.130 | 0.205 | 0.076 | 0.076 |
| SF-SKNN | 0.145 | 0.284 | 0.593 | 0.104 | 0.126 | 0.199 | 0.072 | 0.072 |
| AR | 0.143 | 0.278 | 0.517 | 0.170 | 0.125 | 0.198 | 0.070 | 0.070 |
| IKNN | 0.131 | 0.242 | 0.673 | 0.010 | 0.117 | 0.179 | 0.073 | 0.073 |