

A Case Study on Sampling Strategies for Evaluating Neural Sequential Item Recommendation Models

Alexander Dallmann*

Daniel Zoller*

Andreas Hotho

dallmann@informatik.uni-wuerzburg.de

zoller@informatik.uni-wuerzburg.de

hotho@informatik.uni-wuerzburg.de

University of Würzburg

Data Science Chair

Würzburg, Germany

ABSTRACT

At the present time, sequential item recommendation models are compared by calculating metrics on a small item subset (*target set*) to speed up computation. The target set contains the relevant item and a set of negative items that are sampled from the full item set. Two well-known strategies to sample negative items are *uniform random sampling* and *sampling by popularity* to better approximate the item frequency distribution in the dataset. Most recently published papers on sequential item recommendation rely on *sampling by popularity* to compare the evaluated models. However, recent work has already shown that an evaluation with *uniform random sampling* may not be consistent with the full ranking, that is, the model ranking obtained by evaluating a metric using the full item set as target set, which raises the question whether the ranking obtained by sampling by popularity is equal to the full ranking. In this work, we re-evaluate current state-of-the-art sequential recommender models from the point of view, whether these sampling strategies have an impact on the final ranking of the models. We therefore train four recently proposed sequential recommendation models on five widely known datasets. For each dataset and model, we employ three evaluation strategies. First, we compute the full model ranking. Then we evaluate all models on a target set sampled by the two different sampling strategies, *uniform random sampling* and *sampling by popularity* with the commonly used target set size of 100, compute the model ranking for each strategy and compare them with each other. Additionally, we vary the size of the sampled target set. Overall, we find that both sampling strategies can produce inconsistent rankings compared with the full ranking of the models. Furthermore, both sampling by popularity and uniform random sampling do not consistently produce the same ranking when compared over different sample sizes. Our results suggest that like uniform random sampling, rankings obtained by sampling by popularity do not equal the full ranking of recommender models and therefore both should be avoided in favor of the full ranking when establishing state-of-the-art.

CCS CONCEPTS

• **Information systems** → **Recommender systems; Evaluation of retrieval results.**

*Both authors contributed equally to this research.

KEYWORDS

Sequential Item Recommendation, Evaluation, Metrics, Sampled Metrics

1 INTRODUCTION

A crucial part of developing recommender systems is the evaluation of the model candidates during the process. Online evaluation is still the best choice for evaluating recommender models [28], but it is not applicable during the development of a new model (e.g., for finding the best hyper-parameter settings). So, offline evaluation remains the best option for the recommender community to evaluate new models during development. Because recommender systems must cover increasingly larger areas of application (in terms of number of items) in recent years, Koren [24] introduced an evaluation procedure to speed up the process of metric calculation during evaluation on these larger item sets. Instead of computing the metrics on the full item set to get a ranking (called *full ranking*) for the models, the metrics are computed on a target set, that is a small subset of items, containing all relevant items and a defined number of negative (non-relevant) items that are sampled *uniform* from the full item set. Many scientific work has adapted the method by mainly changing the number of negative samples in the target set (e.g., [10, 12, 21]). Others have changed the process by sampling the item set based on the *popularity* of the items [37] to make the sampling more representative and reliable. It became common practice to use the sampling not only while training or developing the model, but also for reporting the performance of the recommender models [21, 24, 37].

Recently, in [25] the authors formally showed that the expected values of most utilized metrics for recommender evaluation based on *uniform sampling* depend on the rank of the relevant item assigned by the model under evaluation. Therefore, the ranking on the sampled target set can differ from the one on the full item set. Attempts to correct the metrics were made by Krichene and Rendle [25] and also by Li et al. [26] in respect to the Hit Rate metric. Since current state-of-the-art models for sequential recommendation, that use neural networks to extract the user's preference, have also been evaluated using these sampling strategies, we aim to re-evaluate the results of the past years with respect to how their performance ranking changes under the different sampling options. Thereby, we want to validate and extend the findings of [25] to

these types of recommender models. In particular, we focus on *sampling by popularity* for target set generation, because its effects on comparative ranking with other models has not yet been studied. To that end, we train four neural sequential recommendation models, namely, GRU [38], a Recurrent Neural Network (RNN), NARM [27], a RNN with attention, SASRec [21] and BERT4Rec [37], both transformer-based models on five commonly used datasets Steam, Amazon Beauty and Games and Movielens ML-1m and ML-20m. Using the trained models, we provide a comparative evaluation of model performance rankings obtained using three different evaluation strategies (1) full item set, (2) sampling by popularity and (3) uniform random sampling. Furthermore, we also investigate changes in rank for different sizes of the sampled target set. In summary, our main contributions are:

- (1) We are the first, that analyze the effects of sampling the target set by popularity on the consistency with the full ranking and the ranking obtained by sampling the target set uniform random.
- (2) We re-evaluate four current state-of-the-art models for sequential recommendation on five commonly used datasets to confirm the previously reported ranking achieved with sampling the target set by popularity and compare this ranking with the full ranking.
- (3) We test, whether sampling the target set uniform random yields an inconsistent model ranking with the full ranking as reported by Krichene and Rendle [25] on different models and datasets compared to them.
- (4) We examine experimentally the consistency between the full ranking and the ranking obtained by sampling on different target set sizes.

In our experiments we can reproduce the results of the previously reported rankings of sequential recommender models using sampling the target set by popularity. However, our results also show that both rankings obtained by sampling are inconsistent with the full ranking on all datasets we tested. We thereby affirm the reported results of Krichene and Rendle [25] for more datasets and neural sequential recommender models. When varying the target set size on the datasets, we find that the sampled rankings are not consistent with the full ranking. Overall, our results suggest that the full ranking should be used when comparing model performance.

The remainder of this paper is structured as follows: First, we layout the general setting in Section 2. Then we discuss related work in Section 3 After describing the experimental setup in Section 4, we present the obtained results of our experiments in Section 5. Before we conclude the paper in Section 7, we discuss our findings in Section 6.

2 SETTINGS

In this section we first specify the sequential item recommendation task. Then we formally introduce the evaluation setup and metrics we used to score the recommender models. Furthermore, we define the strategies for sampling a target set, that we investigate in this paper.

2.1 Sequential Item Recommendation Task

The goal of a sequential item recommendation model is to learn a user’s preferences based on her history of item interactions (e.g., rating a movie) and to recommend new relevant items based on the accumulated information. Formally, let $I = \{i_1, i_2, \dots, i_{|I|}\}$ be the set of items and $U = \{u_1, u_2, \dots, u_{|U|}\}$ the set of users. We construct sequences $s^u = (s_1^u, s_2^u, \dots, s_{l_s^u}^u) \in S$ with length l_s^u and $s_j^u \in I$ for each user u .¹ A sequential recommendation model $M : S \rightarrow R_n$, with R_n being the set of all permutations of the position list $(1, 2, \dots, |I|)$, is now tasked with ranking the next item first in the position list, given a sequence of past interactions. For later, we define the function $\text{head}(s, t) = (s_1, s_2, \dots, s_t)$, that returns the first t steps of a sequence s and the function $\text{set}(s)$, that converts a sequence s to a set containing all items of sequence s .

2.2 Dataset Split Strategy

We use the leave-one-out evaluation strategy as is common in most related work (e.g., [21, 37]). Therefore, for training we extract for each sequence in S the subsequence containing all items except the two last items: $S_{\text{train}} = \{\text{head}(s, l_s - 2) \mid s \in S\}$. The model is validated on the penultimate item $s_{l_s - 1}$ and later tested on the last item s_{l_s} for each sequence in the dataset using the metrics defined in the next section.

2.3 Evaluation Metrics

We use two common evaluation metrics for evaluating the models. Given a sequence s and relevant target items $i_r = s_{l_s - 1}$ and $i_r = s_{l_s}$ during validation and testing respectively, the hit rate at position k (HR@k) [34] measures if the relevant item is in the head/top k of the returned ranking of model M given the sequence $\hat{s} = \text{head}(s, l_s - 2)$ for validation and $\hat{s} = \text{head}(s, l_s - 1)$ for testing:

$$\text{HR@k} = |\{m_i \in \text{head}(M(\hat{s}), k) \mid m_i = r\}|. \quad (1)$$

Note, that the HR@k is equal to the Recall@k in our setting with only one relevant item. The second metric we use for our experiments is the Normalized Discounted Cumulative Gain (NDCG) [34], that can be formalized in our setting as:

$$\text{NDCG@k} = \sum_{i=1}^k \delta(M(\hat{s})_i, r) \frac{1}{\log(i+1)}, \quad (2)$$

where $M(\hat{s})_i$ is the i -th entry of the ranking and $\delta(a, b) = 1$ if $a = b$ otherwise 0.

2.4 Ranking of Recommendation Models

To rank a set of recommender models, we rank a target set of items for every sequence in the test set using each model. We calculate the metrics on the ranked items and then average the values for each model and rank the models using the mean. In this paper we investigate three different strategies to create the target set of items and name the ranking according to the used method to extract the target set for calculating the metrics: (1) For the *full ranking* we calculate the metrics on the target set that is equal to

¹Since we only leverage the user information to build the interaction sequence, we drop the superscript u in the following for better readability.

²Because the recommendation model M returns the item identifiers, we compare these identifiers with the identifier r of the relevant item.

the full item set. (2) The target set for the *uniform ranking* consists of η non-relevant items that are uniform random sampled from the item space and the relevant item i_r . In this paper, we remove all items that are part of the sequence \hat{s} from the sampled item space, because in our setting the user cannot interact with an item twice. So, the set of items, where we can sample from, for a sequence \hat{s} , is $N = I \setminus \{i_r\} \setminus \text{set}(\hat{s})$. (3) The *popularity ranking* is the same as the previous ranking except we sample items from N using the item’s popularity.

Further, we adapt the definition of Krichene and Rendle [25] and define a ranking R of a set of recommender models *consistent* with ranking Q of the same models iff the rank of all models in R is equal to the rank in Q . That means, that the metric value of a recommender model in R used to determine the ranking can be different to the one obtained for determining the rank in ranking Q , but two obtained ranks of the model must be the same.

3 RELATED WORK AND BACKGROUND

In this section we give a brief background overview about sequence recommendation and list the different random sampling evaluation settings used in previous model evaluation. Further we list other studies that analyze the sampling evaluation methodology for recommender systems.

3.1 Background

In this subsection we provide some background about sequence recommender models and the evaluation methodologies used in previous work.

3.1.1 Sequential Recommendation Models. First work on recommender systems is based on Collaborative Filtering (CF) to model the user’s interest given her previous interactions with items [33]. All first proposed CF models, like Matrix Factorization [33], ignore the order of the user’s interactions to learn the user’s preference. To overcome this drawback, sequential recommendation models were introduced. For example, Zimdars et al. [42] models the sequence for the recommendation task for the first time using first-order Markov Chains. In recent years, models, that leverage different neural network types for encoding the sequence into a representation, have been published. Early work used Recurrent Neural Networks (RNNs) with Gated Recurrent Units (GRUs) [7] or a Long Short-Term Memory (LSTM) [17] units as the encoder for the sequence. While Hidasi et al. [16] use GRUs with a ranking loss, Tan et al. [38] find that GRUs with cross-entropy loss outperforms GRUs without the ranking loss used in [16]. A Convolutional Neural Network is used in the work of Tang and Wang [39]. Attention, a mechanism that improves Natural Language Processing (NLP) tasks like machine translation [2, 29], was first applied to the sequential recommendation task by Li et al. [27]. Their model NARM uses attention with a GRU-based RNN network to encode the sequence into a representation. In [21] the authors propose the SASRec model, that is only based on attention and encodes the sequence using a unidirectional Transformer network [41]. Sun et al. [37] extend SASRec by using a bidirectional encoder and adapt the BERT architecture and the training objective of the model, that is successfully used

for language modelling in the NLP community, for the recommendation setting. At the very time of this writing this model is the current state-of-the-art for sequence recommendation.³

3.1.2 Negative Sampling for Recommender Model Evaluation. In this section, we will point to some work that uses random sampling to build the target set for evaluating recommender models. Koren [24] use for the first time random sampling to build target sets. The author draws 1000 negative samples at random when evaluating different movie recommenders with the Root Mean Squared Error (RMSE) as metric. In [10] the authors use uniform negative random sampling with a size of 9 to evaluate their deep learning model for news, application and movie/tv recommendation using the Mean Reciprocal Rank (MRR) and HR@1. The current commonly used negative sample size of 100 samples was introduced by [12] to calculate the HR@10 and NDCG@10 for different recommenders on the ML-1m dataset and a Pinterest dataset. While the authors of the SASRec model follow the procedure of [21] to calculate the Hit Rate and the NDCG, the authors of the BERT4Rec model sample the 100 negative item based on the popularity distribution of the dataset [37]. In contrast to this, in [20] the authors apply a different approach and draw 50 negative samples *uniform* from the item set and another 50 items *by their popularity*.

3.2 Related Work

Early work already studied the effects of the evaluation methodology while testing CF recommender methods [3]. In this work, the authors find that the ranking between the tested three matrix factorization models is inconsistent when evaluating them using four different methods, including *uniform random sampling*, on the Movielens ML-1m dataset. They calculate Recall, Precision and NDCG as metric to rank the studied models. Later, Steck [36] also finds differences in model rankings of two matrix factorization models when using all unrated items or only the observed ratings in the test set of a user for a rating prediction setup and RMSE as metric. In [4] the authors conduct studies varying the number of negative samples while comparing two collaborative filtering models, a k-Nearest Neighbor (kNN) [8] and a probabilistic Latent Semantic Analysis (pLSA) [18] recommender model. They find no ranking inconsistency in the rankings of the two models while testing on the Movielens ML-1m dataset obtained by metrics like Precision. More recently, Rendle [31] shows on a constructed sample setting that evaluation using the *uniform sampling* is not consistent when using metrics like HR or NDCG. In [25] the authors extend the work of Rendle by providing adapted metrics to overcome this problem. They test the proposed corrected metrics on the ML-1m dataset using a matrix factorization and two item-based collaborative filtering models [33] and find that the rankings stay consistent starting with fewer negative samples than without the corrected metrics. At the same time, Li et al. propose a dataset independent mapping function for the Hit Rate (HR), to approximate the HR obtained on the full item set of a model and therefore determine the full ranking of the considered models [26]. Their experiments conducted with collaborative filtering methods like MultiVAE [28], a variational autoencoder, for example, on the ML-1m dataset demonstrate the

³Here, we only consider models that exclusively use the sequence information and nothing else like, for example, the user or the time of the interaction.

applicability of their approximation function. Closest to our work, Cañamares and Castells [5] find that rankings of kNN [6] and implicit matrix factorization [19] on different sampling sizes drawn from a *uniform* distribution using a k-fold evaluation on the MovieLens 1M dataset and the Yahoo! R3 dataset are inconsistent when varying the number of negative samples.

To the best of our knowledge this is the first work, that conduct studies about the consistency of recommender model rankings when evaluating the models by sampling negative items by popularity. All previous work has only considered the uniform sampling approach in their analysis. In addition, we compare the model rankings of current state-of-the-art sequential recommendation models, instead of simple CF recommendation models like [5, 25, 26, 31], on different sampling based evaluation methods. Furthermore, we perform these studies on five datasets commonly used for evaluation instead of just one or two datasets as before to find possible dependencies of rankings and evaluation methods on the dataset’s properties. We extend the work of [5], that examine the consistency of the model rankings when uniform sampling different numbers of negative items, and use the provided method to perform the same investigations for sampling by popularity.

4 EXPERIMENTAL SETUP

In this section we describe the experimental setup, including the recommender models and datasets we used for our analysis. We also introduce the methods used to analyze the difference in the obtained model rankings.

4.1 Sequential Item Recommendation Models

For investigating the random sampling evaluation, we use four state-of-the-art neural sequential recommendation networks. Each network encodes the sequence s using three different sub-layers: (1) an embedding layer, that embeds each item s_t in the sequence using an embedding matrix $M \in \mathbb{R}^{|I| \times e}$, with $e_t = M_{s_t}$, where M_j is the j -th row of matrix M , (2) a sequence encoder, that transforms the embedded sequence to a representation $h \in \mathbb{R}^m$, where m is the size of the representation, (3) and an output layer, that projects the sequence representation back to the item space. In the following we describe each used model in more detail.

4.1.1 Gated Recurrent Unit (GRU). The GRU model of Tan et al. [38] uses a Recurrent Neural Network (RNN) layer with Gated Recurrent Units (GRUs) [7], as the sequence encoder for the sequence. RNNs encode sequences by learning internal hidden states $h_t \in \mathbb{R}^d$, where d is the size of the sequence representation, at each sequence step t , given the current input e_t and the hidden state of the previous step h_{t-1} .⁴ Given $W_h \in \mathbb{R}^{e \times d}$ and $R_h \in \mathbb{R}^{d \times d}$ as the weight matrices for the nonlinear transformation of the current input and the previous state and an activation function g , the hidden state of an RNN network is calculated by:

$$h_t = g(W_h e_t + R_h h_{t-1}).^5 \quad (3)$$

RNNs suffer from the problem of vanishing gradients for long sequences. To overcome this, the GRU memory unit utilizes two

gates: (1) The *update gate* z controls how much of the previous state is passed to the next state(s) and (2) the *reset gate* r , that controls to which amount the network forgets the past state(s). The state of recurrent network with GRUs can then be calculated with:

$$z_t = \sigma(W_Z e_t + R_Z h_{t-1}) \quad (4)$$

$$r_t = \sigma(W_R e_t + R_R h_{t-1}) \quad (5)$$

$$h_t = (1 - z_t) \cdot h_{t-1} + z_t \cdot \tanh(W_H e_t + R_H (r_t \cdot h_{t-1})), \quad (6)$$

where $W_{Z,R,H} \in \mathbb{R}^{d \times e}$ and $R_{Z,R,H} \in \mathbb{R}^{d \times d}$ are the weight matrices for the nonlinear transformation. We use the last hidden state of the GRU h_{I_s} as the sequence representation and scale the network output to the item space using a feed forward network for the recommendation task: $o = W_s h_{I_s}$ with $W_s \in \mathbb{R}^{n \times d}$. We train the model with all subsequences of all sequences and use the cross-entropy loss to optimize it:

$$L = - \sum_{s \in S} \sum_{t=1}^{s_{I_s}-1} \log(\text{softmax}(o(\text{head}(s, t))))_{s_{t+1}} \quad (7)$$

4.1.2 Neural Attentive Recommendation Machine (NARM). The Neural Attentive Recommendation Machine (NARM) [27] consists of two different GRUs: (1) a global encoder and (2) a local encoder. The global encoder is a GRU, that we already defined in the previous paragraph, thus the output of this encoder is $c^G = h_{I_s}^G \in \mathbb{R}^d$. In contrast, the local encoder learns based on another GRU an attention to all hidden states of the sequence [2, 29]. The output of the local encoder is defined as $c^L = \sum_{j=1}^{I_s} \alpha_{I_s,j} h_j^L$, where $h_m^L \in \mathbb{R}^d$ are the hidden states of the local encoder. The attention score $\alpha_{I_s,j}$ calculates an alignment between the j -th hidden state h_j^L and the last hidden state $h_{I_s}^L$ using:

$$\alpha_{I_s,j} = q(h_{I_s}^L, h_j^L) = V^\top \sigma(A_1 h_{I_s}^L + A_2 h_j^L), \quad (8)$$

where, given a latent space a for α ,⁶ A_1 and $A_2 \in \mathbb{R}^{d \times a}$, $V \in \mathbb{R}^a$. For the overall output of the encoder the representations of the global and local encoder are concatenated to $c = [c^G; c^L]$ and then transformed with a learn-able weight matrix $B \in \mathbb{R}^{e \times 2d}$. To scale the output to the item space, NARM uses the transposed item embedding matrix M : $o = BcM^\top$. We train NARM the same way as the GRU model and also use the same loss.

4.1.3 Self-Attention based Sequential Recommendation Model (SAS-Rec). The Self-Attention based Sequential Recommendation model (SASRec) [21] is based on Transformer Networks [41]. Because Transformer Networks are not recurrent networks they do not have the information about the position, an additional embedding $P \in \mathbb{R}^{T \times d}$, where T is the maximum sequence length of the network and d its embedding and sequence representation size, is added to learn the position t of the item s_t in the sequence: $\hat{e}_t = e_t + P_t$. SASRec consists of L attention layer blocks, that are stacked on top of each other. Each block at depth j is a Multi-Head Self-Attention layer MH^j followed by a pointwise feed-forward network F . For a Multi-Head Self-Attention layer, a single Self-Attention layer SA_m^j is applied H times to its input, the results are concatenated and

⁴We omit the output part of the RNN because we do not use this feature in our setting.

⁵For this equation and all following equations in this paper we omit the bias term of the neural networks for readability.

⁶In practice the latent space is equal to the hidden space of the RNN encoder ($a = d$).

transformed using a weight matrix $W_O^j \in \mathbb{R}^{d \times d}$:

$$MH^j(x) = [SA_1^j(x); SA_2^j(x); \dots; SA_H^j(x)]W_O^j. \quad (9)$$

Each Self-Attention layer linearly projects the input into a smaller space and then applies a mechanism called *scaled-dot* attention [41]:

$$SA_m^j(x) = \text{attention}(xW_Q^j, xW_K^j, xW_V^j), \quad (10)$$

where $W_Q^j, W_K^j, W_V^j \in \mathbb{R}^{d \times d/H}$ are linear projection matrices. The attention function can be seen as a proportional retrieval of values V given keys K and queries Q (in the case of self-attention $K = Q = V$) and is defined as:

$$\text{attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right)V, \quad (11)$$

The pointwise feed-forward network F uses a ReLU [30] activation and two weight matrices $W_{F_1}^j$ and $W_{F_2}^j \in \mathbb{R}^{d \times d}$:

$$F^j(x) = \text{ReLU}(MH^j(x)W_{F_1}^j)W_{F_2}^j \quad (12)$$

To stabilize training and to prevent overfitting, a residual connection [11] is added to both layers z of the block. Further, layer normalization [1] and Dropout [35] is applied to the input x of the layers, resulting in:

$$z(x) = x + \text{Dropout}(z(\text{LayerNormalization}(x))). \quad (13)$$

SASRec uses the state of the last sequence step \hat{i} of the last layer L as the representation of the sequence. To project the presentation into the item space the model leverages the embedding matrix M by $o(x) = F_{\hat{i}}^L(x)M^\top$. At training time the network tries to predict each next sequence step in the sequence s_{t_s} by taking s_{t_s-1} as input. For this learning objective, all attention connection between Q_m and K_n with $m > n$ are masked to prevent the network to attend to subsequent items in the sequence. SASRec uses the BPR loss [32] for optimizing the network. Therefore, it samples a negative item $i_{t,\text{neg}}$ (i.e., an item that was not in the sequence and is not the target) randomly from the item set for each sequence step. The overall loss is then:

$$L = - \sum_{s \in S} \sum_{t=1}^{s_{t_s}-1} \log(\sigma(o(\text{head}(s, t)))_{s_{t+1}}) + \log(1 - \sigma(o(\text{head}(s, t)))_{i_{t,\text{neg}}}). \quad (14)$$

4.1.4 BERT4Rec. The BERT4Rec model [37] makes two modifications to the transformer architecture of SASRec and one to the training objective. In contrast to SASRec, BERT4Rec allows the model to attend to all positions of the sequence and uses GELU [13] instead of a ReLU as activation function for all non-linear feed-forward networks. Instead of predicting the next item in the sequence, BERT4Rec adapts the training objective of the language model BERT [9], that uses the Cloze task [40] for self-supervised training. The model randomly masks items in the sequence and then tries to predict the masked item. Since after training the prediction is generated by masking the last item in the sequence, the model additionally sometimes only masks the last item of the sequence while training. Given the set $T_{s,\text{mask}}$, that contains the sequence positions of each masked item in the sequence s , the loss of the

Table 1: Statistics about the five datasets after pre-processing. We report the number of users, items and actions as well as the average sequence length (A. L.) and the density.

Dataset	#users	#items	#actions	Avg. Length	Density
Amazon Beauty	40,226	54,542	0.4m	8.80	0.02%
Amazon Games	29,341	23,464	0.3m	9.58	0.04%
ML-1m	6,040	3,416	1.0m	165.50	4.84%
ML-20m	138,493	26,729	20.0m	144.41	0.54%
Steam	334,537	13,046	4.2m	12.59	0.10%

network is a cross entropy loss:

$$L = - \sum_{s \in S} \sum_{t \in T_{s,\text{mask}}} \log(\text{softmax}(o(s))_{s_t}). \quad (15)$$

4.2 Datasets

We perform all experiments on five frequently used datasets that are based on user reviews in different domains. The first two datasets *ML-1m* and *ML-20m*⁷ are based on movie reviews that have been collected from the non-commercial movie recommendation site Movielens. While *ML-1m* only contains about 1 million interactions, the *ML-20m* contains about 20 million interactions. Further, we use datasets consisting of collected product reviews from Amazon organized into categories.⁸ We will restrict our experiments to the two most widely used categories in related work: *Beauty* and *Games*. The last dataset is comprised of game reviews crawled from the digital video game distribution service Steam.⁹

We apply the following common pre-processing steps: First, all reviews are grouped by users and ordered by timestamp. We treat a review of an item as an interaction with an item and building on that the sequence of interactions for each user. Next, we remove all items and users with fewer than five occurrences from the dataset. We follow the procedure in [37] and first build statistics for items and remove items under the threshold and then compute the user statistics and remove the users accordingly. This is handled differently throughout the literature. For example, in [21] statistics are computed on the original dataset and then both users and items are removed in one pass which leads to a slightly different dataset after pre-processing. We skip the filtering step for *ML-20m* since the statistics of the original dataset already closely match the reported statistics in different papers and application of the filtering yields large deviations from these reported values. The overall statistics of all datasets after pre-processing can be found in Table 1.

4.3 Model Training and Verification

In this section we describe the model training. Further, we also verify that our implemented models achieve similar results to the previously reported results.¹⁰

⁷<https://grouplens.org/datasets/movielens/>

⁸<http://jmcauley.ucsd.edu/data/amazon/links.html>

⁹<https://github.com/kang205/SASRec>

¹⁰All code for training and evaluating the models is available at <https://professor-x.de/papers/metrics-sampling-eval/>.

Table 2: Performance comparison of our implementations of GRU, SASRec and BERT4Rec with values from Sun et al. [37] for the HR@10 metric with popularity sampling and negative sample size $\eta = 100$. GRU is compared to the values of the *GRU4Rec+*.

Dataset	Source	GRU		SASRec		BERT4Rec	
		ours	[37]	ours	[37]	ours	[37]
Amazon Beauty		0.188	0.265	0.241	0.265	0.281	0.303
ML-1m		0.628	0.635	0.656	0.669	0.663	0.697
ML-20m		0.651	0.652	0.706	0.714	0.732	0.747
Steam		0.344	0.359	0.384	0.378	0.414	0.401

4.3.1 Implementation and Training. For the evaluation we implemented all four models using PyTorch.¹¹ Then, we retrain the three sequence models SASRec and BERT4Rec with the hyper-parameters reported in [37], or if not reported we used the parameters found in the corresponding repositories of the papers. For GRU and NARM we used the hyper-parameter of the corresponding papers. All models are trained for a maximum of 800 epochs. We use the Adam optimizer [23] for training the networks and selected the best model based on the metric HR@10 using the validation split. All configurations of the models can be found in the additional material of this paper.

4.3.2 Verifying Models. To verify our implementations we compare the performance of our models with the reported ones in [37]. Since NARM was not evaluated on the considered datasets in this paper, we can not verify our implementation. But we will later show in the results section that the obtained relative rankings seems plausible. Furthermore, we compare the results of our GRU implementation of [38] with the results of *GRU4Rec+* [15]. Table 2 shows our results for HR@10 compared to the results reported in [37] with popularity sampling and $\eta = 100$ negative samples. Overall, the results of our implementations align with previously reported findings for this evaluation setting, where BERT4Rec is best across the considered models and SASRec comes in second, and GRU comes in third. Our GRU implementation is competitive, compared with the reported values for *GRU4Rec+* in [37]. Slight differences in the obtained metric scores, can be explained by small differences in parameter initialization due to the use of different neural network frameworks. From these results we conclude that our implementations match the originals close enough for a meaningful comparison.

4.4 Ranking Evaluation Methods

To stabilize the obtained results, we run the experiments involving random sampling 20-times for each model. We report the mean value over all 20 runs in the results section, but we omit the standard deviation, since we find that the standard deviation is zero, when we round it up the last three digits after the decimal point. To compare two calculated rankings, we use the Kendall’s Tau correlation coefficient [22] to determine how similar two rankings are. Since we do not have ties in our rankings, we use the Kendall’s

¹¹<https://pytorch.org>

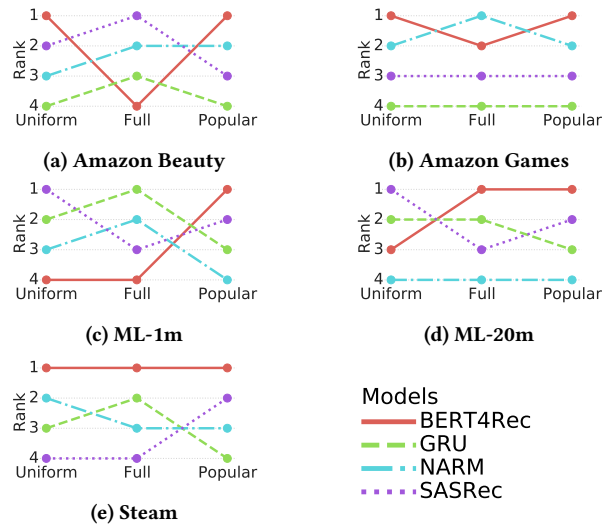


Figure 1: Rankings based on HR@10 of the four recommender models GRU, NARM, SASRec and BERT4Rec on the five considered datasets Amazon Beauty and Game, MovieLens ML-1m and ML-20m and Steam. For each dataset we compare the the *full ranking* in the middle with *uniform ranking* on the left side and the *popular ranking* on the right side. The number of negative samples is set to $\eta = 100$ for both sampling rankings.

Tau-a, that can be calculated by:

$$\tau = \frac{2(m_c - m_d)}{m(m-1)}, \quad (16)$$

where m is the number of rankings (in our case 4) and m_c the number of *concordant* pairs and m_d the number of *discordant* pairs. A pair (x_1, y_1) and (x_2, y_2) is *concordant* if either $x_1 < x_2$ and $y_1 < y_2$ or $x_1 > x_2$ and $y_1 > y_2$, otherwise the pair is *discordant*. The values of the Kendall’s Tau correlation range between -1 and 1 , where -1 indicates a perfect inversion and 1 a perfect agreement of the rankings.

5 RESULTS

First, in this section we compare the full ranking of the considered models with the rankings achieved by sampling by popularity and uniform sampling. Second, we consider how the number of negative samples affects the rankings and whether it can be chosen such that the rankings generated by the random selection strategies are consistent with the full ranking.

5.1 Full Ranking and Random Sampling Rankings

In this section we take a look at how well the rankings based on one of the two sampling strategies approximates the full ranking. Therefore, we evaluate the model rankings with all three options (full, uniform and popular) for both metrics HR@10 and NDCG@10. Additionally, we calculate the Kendall’s Tau correlation (τ) between the sampling rankings and the full ranking. For the uniform and

Table 3: The full, popular and uniform rankings of the four recommender models GRU, NARM, SASRec and BERT4Rec on the five considered datasets Amazon Beauty and Game, Movielens ML-1m and ML-20m Steam with the corresponding HR@10 metric, that was used to build the rankings. For both sample-based ranking we used $\eta = 100$ negative samples and we also report the Kendall’s Tau correlation coefficient τ with respect to the full ranking in the first column. For each dataset the best HR@10 (highest) value is marked as bold and second best value is underlined. Additionally, the ranking is visualized with ■ symbols for each dataset independently. The number of ■ symbols corresponds to the achieved rank (fewer symbols are better).

Dataset	Model	full		popularity			uniform		
		HR@10	rank	HR@10	rank	τ	HR@10	rank	τ
Amazon Beauty	GRU	0.031	■■■	0.188	■■■■	-0.33	0.349	■■■■	0.00
	NARM	<u>0.033</u>	■■	<u>0.243</u>	■■		0.417	■■■	
	SASRec	0.036	■	0.241	■■■		<u>0.420</u>	■■	
	BERT4Rec	0.027	■■■■	0.281	■		0.425	■	
Amazon Games	GRU	0.066	■■■■	0.380	■■■■	0.67	0.596	■■■■	0.67
	NARM	0.087	■	<u>0.514</u>	■■		<u>0.712</u>	■■	
	SASRec	0.068	■■■	0.451	■■■		0.688	■■■	
	BERT4Rec	<u>0.082</u>	■■	0.518	■		0.720	■	
ML-1m	GRU	0.224	■	0.628	■■■	-0.67	<u>0.768</u>	■■	0.33
	NARM	<u>0.202</u>	■■	0.628	■■■■		0.764	■■■	
	SASRec	0.185	■■■	<u>0.656</u>	■■		0.798	■	
	BERT4Rec	0.160	■■■■	0.663	■		0.763	■■■■	
ML-20m	GRU	<u>0.199</u>	■■	0.651	■■■	0.67	<u>0.964</u>	■■	0.00
	NARM	0.118	■■■■	0.569	■■■■		0.952	■■■■	
	SASRec	0.137	■■■	<u>0.706</u>	■■		0.974	■	
	BERT4Rec	0.241	■	0.732	■		0.964	■■■	
Steam	GRU	<u>0.198</u>	■■	0.344	■■■■	0.00	0.829	■■■	0.67
	NARM	0.196	■■■	0.355	■■■		<u>0.840</u>	■■	
	SASRec	0.183	■■■■	<u>0.384</u>	■■		0.826	■■■■	
	BERT4Rec	0.215	■	0.414	■		0.861	■	

popular rankings, we use $\eta = 100$ negative samples which is used by recent work [21, 37].

Table 3 shows the obtained three rankings of the four models on the five datasets with the corresponding metric scores for HR@10 and the corresponding Kendall’s Tau correlation values between the sampled rankings and the full ranking. Figure 1 visualizes the difference of the three model rankings for each dataset. As we can observe from the table, the uniform rankings are not consistent with the full ranking across all five datasets when comparing the four models. For example, the inconsistency is especially evident when considering the Kendall’s Tau correlation coefficient for the Amazon Beauty dataset, where $\tau = 0.00$, that is, that there are no matches in rank between the two rankings. This can also clearly be seen in Figure 1a. Overall, our findings are in line with the ones by Krichene and Rendle [25]. Furthermore, from this it can be concluded that the results in [25] obtained on a matrix factorization and two collaborative filtering models from [33] can be extended to the more recent state-of-the-art deep learning models. When we now look at the popularity ranking, we find that this ranking is also not consistent with the full ranking like the uniform ranking. Indeed for ML-1m we report $\tau = -0.67$, which means that the ranking is almost inverse (see Figure 1c) to the full ranking. After all (when we exclude NARM, since it was not evaluated on the five considered datasets) we can confirm the model ranking GRU, SASRec,

BERT4Rec on the datasets Amazon Beauty, ML-1m, ML-20m and Steam reported by [37] using the popular ranking. But looking at the full ranking, we can see that the models perform differently well on the five datasets. For example, when considering the full ranking the BERT4Rec model outperforms the other methods on the ML-20m and Steam datasets, but is the model with the worst HR@10 on the ML-1m and Amazon Beauty datasets. This fact suggests that the model ranking depends on the evaluation method used (with or without sampling). At last, when we compare both sampling methods with each other, Table 3 and Figure 1 reveal that both sampling methods are mostly inconsistent with each other. The exception is the Amazon Games dataset, where both sampling methods agree on a ranking, but are still inconsistent with the full ranking. Due to space limitations, we omit the table for the rankings using NDCG@10 as metric, since we observe overall the same behavior on all rankings on all datasets when using NDCG@10 instead of HR@10.

To summarize, with our findings we extend previous results, that uniform rankings are inconsistent with the rankings obtained on the full item set, to recent state-of-the-art sequential recommendation methods. Furthermore, we find that the popular rankings are also inconsistent with the exact ranking when considering 100 negative samples ($\eta = 100$).

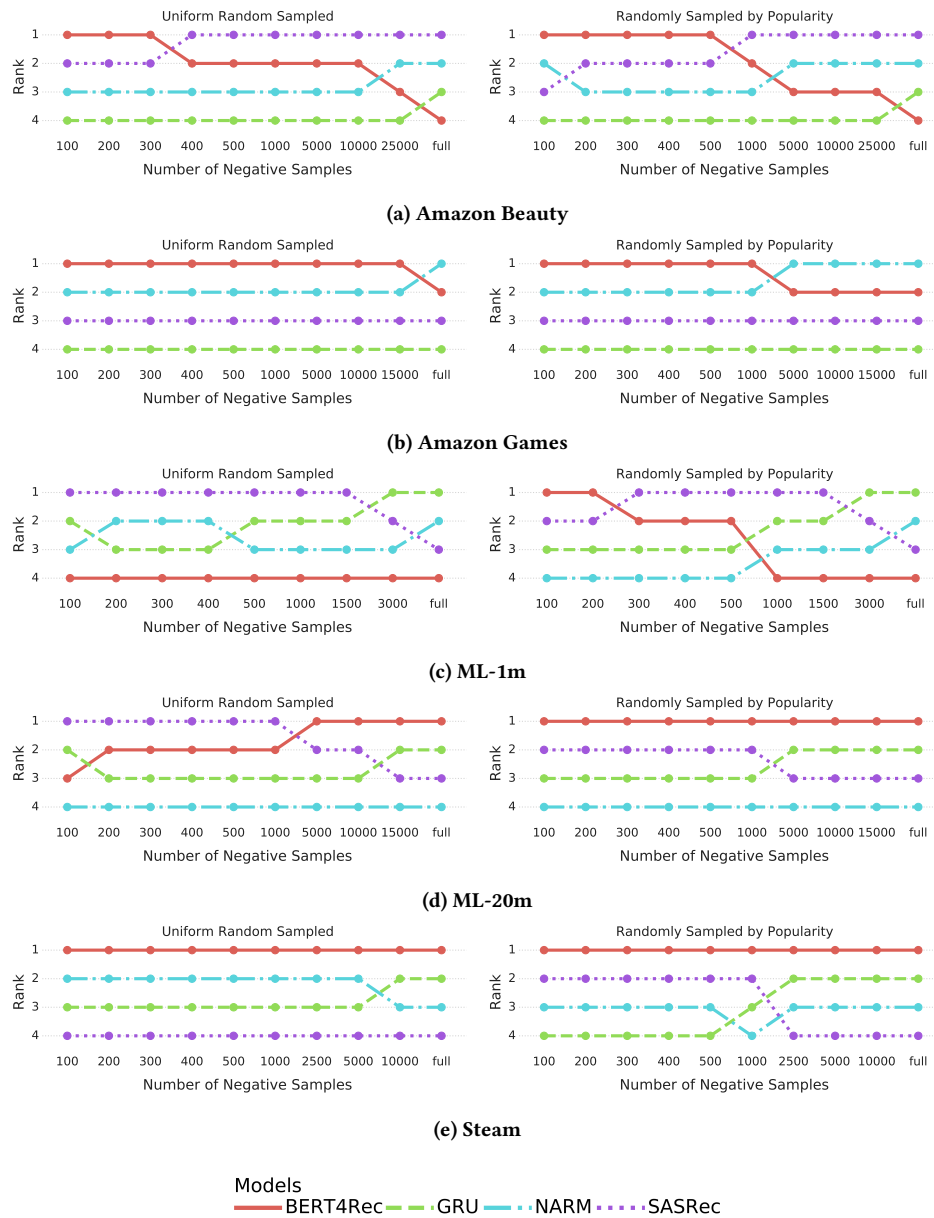


Figure 2: Rankings of the four recommender models GRU, NARM, SASRec and BERT4Rec on the five considered datasets Amazon Beauty and Game, Movielens ML-1m and ML-20m and Steam for different negative sample sizes η . We vary the negative sample sizes from 100 to "full", which means that we sample from the full item set, resulting in the *full ranking*. On the left side, we plot the *uniform* rankings and on the right side the *popular* rankings.

5.2 Influence of Sample Size

Previous work Cañameres and Castells [5] showed that changing the sample size can have significant effects on the uniform model rankings. The full ranking can be seen as the boundary case of any sampled evaluation where η is equal to the number of items in the dataset. In this case we can expect the approximation of the full ranking to improve with increasing sample size. We calculate both

sampled rankings for different sample size for all datasets and plot the rank changes in Figure 2 when using HR@10 as the metric.

Overall, we cannot observe many regularities from the plots. The most striking observation is that increasing the sample size by one step only changes the rank by one place in most cases, hinting at a gradual change in recommendation performance with respect to sample size. Additionally, the volatility of the ranking with respect

to sample size seems to be highly dataset and, to some degree, sampling strategy dependent. For example, Figure 2b shows that for Amazon Games almost no changes in rank occur for both sampling rankings and for most sample sizes the full ranking is achieved. However, the simple statistics of Amazon Games presented in Table 1 do not stand out and we cannot derive an explanation from the observed dataset properties. In contrast, Amazon Beauty exhibits a high volatility with frequent ranking changes for both sampling rankings. Even when almost half of the 54,542 items are sampled the ranking has not stabilized and still changes compared to the full ranking. An instance where the behavior is different between sampling strategies is the Steam dataset in Figure 2e. While uniform rankings are almost stable across different sample sizes, popular rankings are more volatile, at least between $\eta = 500$ and $\eta = 2500$. In summary, we cannot clearly identify a safe choice for η for one of the rankings based on sampling nor for the dataset.

6 DISCUSSION AND LIMITATIONS

In our study we trained four recently proposed neural sequential item recommendation models on five often used datasets and compared three rankings, which use different approaches to sample items for the target set used for calculating the metrics. First, we wanted to investigate whether sampling by popularity produces model rankings that exhibit less inconsistencies regarding the full ranking and how it compares to uniform random sampling. We found that both sampling strategies did not produce good approximations of the full ranking for most datasets, with the noteworthy exception of Amazon Games. The obtained rankings on this dataset are for the most sample size consistent with each other. Further analysis of why this is the case is necessary but that is out of the scope of this paper. While sampling by popularity with $\eta = 100$ negative samples produced the model ranking established in recent work [37], the full evaluation showed that BERT4Rec does not hold state-of-the-art across all the datasets. Our results suggest that BERT4Rec profits from larger training corpora, since it outperforms all other models on the two larger datasets ML-20m and Steam using the full ranking, while it was unable to yield the best performance on the other, much smaller, datasets. Further experiments (e.g., hyperparameter studies to find maybe different optima for the metrics calculated on the full item set) must be carried out to analyse how the ranking of the model is really effected by the small datasets. Nevertheless, these results call the current way of establishing state-of-the-art into question. Considering that the ranking depends on the selection for dataset, η and how to sample the target set for metric calculation and that there is no consistent choice for η that yields rankings consistent with the full ranking, the only safe choice for comparing models is to use the full item space as target set, although this may lead to longer runtimes for model evaluation. Additionally, the results show that the clear picture, where one model outperforms all others across a range of datasets does not hold anymore if the full ranking is considered, at least for the considered datasets and models with their reported settings, as can be observed from the results for BERT4Rec. At last, even the full ranking may not be the true ranking, since like all offline evaluation methods, it tends to underestimate the true

metric score by assuming that unrated items are all non-relevant for the user [14].

Despite the extensive evaluation across models and datasets, our study has some limitations. First, we are only comparing recent neural models based on Recurrent Neural Networks or Transformer architectures and do not consider Convolution Neural Networks based models like the one presented in [39] or other collaborative filtering methods or baselines. Therefore, it is possible that for some classes of models a different conclusion could be drawn. Another limitation is the choice of datasets. Although the review-based datasets have been popular in recent publications, there is a variety of datasets from a range of other domains available, that might exhibit different characteristics with impact on the sampled evaluation that were not considered in this evaluation. Finally, we only used the common *leave-one-out* dataset split that is often used with review datasets and did not consider alternative splits like a split by time.

7 CONCLUSION

In our study we focused on the impact of evaluation through sampling of small item sets on inconsistencies with the evaluation on the full item set when comparing recently published neural sequential item recommendation models. While prior research [5] has studied inconsistencies between the ranking obtained using the full item set and the ranking obtained by uniform random sampling mostly on collaborative filtering models, our study extends this line of research in two key ways. First, we include sampling by popularity, which is often used in recent work, as a second sampling strategy in our study and find that although it intuitively better approximates the item distribution in the dataset, it does not improve the consistency with the ranking obtained on the full item set. Second, we study recent neural item recommendation models that can model complex interactions in the item sequences and the results of our experiments indicate that these models are also affected by the ranking inconsistencies when using sampling for the evaluation. Our results strongly suggest that independent of the dataset, type of sampling or choice of the number of negative items η , a sampled evaluation will likely fail to approximate the ranking gained by considering the full item set correctly. Therefore, it is a bad choice when comparing the performance of different sequential recommender models and cannot help to avoid calculating the metrics on the full item set.

In future work we want to extend the study to more models to get better insights into the differences in performance utilizing the full ranking. Additionally, we can extend the study using different types of datasets. We only used datasets constructed from user reviews or ratings, but other types of item sequence datasets exist (e.g., user clicks in online stores), and might exhibit different characteristics. Also we can study the effectiveness of already existing robust metrics introduced by Krichene and Rendle [25] on the evaluated models in this paper and different types of datasets.

ACKNOWLEDGMENTS

This research was partly supported by the Bavarian State Ministry for Science and the Arts within the "Digitalisierungszentrum

für Präzisions- und Telemedizin" (DZ.PTM) project, as part of the master plan "BAYERN DIGITAL II".

REFERENCES

- [1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer Normalization. <http://arxiv.org/abs/1607.06450> cite arxiv:1607.06450.
- [2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *International Conference on Learning Representations*.
- [3] Alejandro Bellogin, Pablo Castells, and Ivan Cantador. 2011. Precision-oriented evaluation of recommender systems. In *Proceedings of the fifth ACM conference on Recommender systems*. ACM Press. <https://doi.org/10.1145/2043932.2043996>
- [4] Alejandro Bellogin, Pablo Castells, and Iván Cantador. 2017. Informational biases in Information Retrieval metrics for recommender systems. *Information Retrieval Journal* 20, 6 (2017), 606–634. <https://doi.org/10.1007/s10791-017-9312-z>
- [5] Rocío Cañamares and Pablo Castells. 2020. On Target Item Sampling In Offline Recommender System Evaluation. In *Fourteenth ACM Conference on Recommender Systems* (Virtual Event, Brazil). Association for Computing Machinery, New York, NY, USA, 259–268. <https://doi.org/10.1145/3383313.3412259>
- [6] Rocío Cañamares and Pablo Castells. 2017. A Probabilistic Reformulation of Memory-Based Collaborative Filtering. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM. <https://doi.org/10.1145/3077136.3080836>
- [7] Kyunghyun Cho, Bart van Merriënboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *EMNLP, Alessandro Moschitti, Bo Pang, and Walter Daelemans* (Eds.). ACL, 1724–1734.
- [8] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. 2010. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*. ACM Press. <https://doi.org/10.1145/1864708.1864721>
- [9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT (1)*, Jill Burstein, Christy Doran, and Tamar Solorio (Eds.). Association for Computational Linguistics, 4171–4186.
- [10] Ali Mamdouh Elkahky, Yang Song, and Xiaodong He. 2015. A Multi-View Deep Learning Approach for Cross Domain User Modeling in Recommendation Systems. In *Proceedings of the 2018 World Wide Web Conference*, Aldo Gangemi, Stefano Leonardi, and Alessandro Panconesi (Eds.). ACM, 278–288. <https://doi.org/10.1145/2736277.2741667>
- [11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition* (Las Vegas, NV, USA). IEEE, 770–778. <https://doi.org/10.1109/CVPR.2016.90>
- [12] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web* (Perth, Australia). International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 173–182. <https://doi.org/10.1145/3038912.3052569>
- [13] Dan Hendrycks and Kevin Gimpel. 2016. Gaussian Error Linear Units (GELUs). <http://arxiv.org/abs/1606.08415>
- [14] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. 2004. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.* 22, 1 (2004), 5–53. <https://doi.org/10.1145/963770.963772>
- [15] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent Neural Networks with Top-k Gains for Session-based Recommendations. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM. <https://doi.org/10.1145/3269206.3271761>
- [16] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks. In *Proceedings of the 4th International Conference on Learning Representations* (San Juan, Puerto Rico). 1–10.
- [17] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural computation* 9, 8 (1997), 1735–1780.
- [18] Thomas Hofmann. 2004. Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems* 22, 1 (Jan. 2004), 89–115. <https://doi.org/10.1145/963770.963774>
- [19] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. In *2008 Eighth IEEE International Conference on Data Mining*, 263–272. <https://doi.org/10.1109/ICDM.2008.22>
- [20] Jin Huang, Wayne Xin Zhao, Hongjian Dou, Ji-Rong Wen, and Edward Y. Chang. 2018. Improving Sequential Recommendation with Knowledge-Enhanced Memory Networks. In *SIGIR*, Kevyn Collins-Thompson, Qiaozhu Mei, Brian D. Davison, Yiqun Liu, and Emine Yilmaz (Eds.). ACM, 505–514. <https://doi.org/10.1145/3209978.3210017>
- [21] Wang-Cheng Kang and Julian J. McAuley. 2018. Self-Attentive Sequential Recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE Computer Society, 197–206. <https://doi.org/10.1109/ICDM.2018.00035>
- [22] Maurice George Kendall. 1948. Rank correlation methods. (1948).
- [23] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR (Poster)*, Yoshua Bengio and Yann LeCun (Eds.).
- [24] Yehuda Koren. 2008. Factorization meets the neighborhood. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD 08*. ACM Press. <https://doi.org/10.1145/1401890.1401944>
- [25] Walid Krichene and Steffen Rendle. 2020. On Sampled Metrics for Item Recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM. <https://doi.org/10.1145/3394486.3403226>
- [26] Dong Li, Ruoming Jin, Jing Gao, and Zhi Liu. 2020. On Sampling Top-K Recommendation Evaluation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Virtual Event, CA, USA). Association for Computing Machinery, New York, NY, USA, 2114–2124. <https://doi.org/10.1145/3394486.3403262>
- [27] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, and Jun Ma. 2017. Neural Attentive Session-based Recommendation. In *Proceedings of the 2017 ACM Conference on Information and Knowledge Management*. ACM Press, New York, NY, USA, 1419–1428. <https://doi.org/10.1145/3132847.3132926>
- [28] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering. In *WWW, Pierre-Antoine Champin, Fabien L. Gandon, Mounia Lalmas, and Panagiotis G. Ipeirotis* (Eds.). ACM, 689–698. <https://doi.org/10.1145/3178876.3186150>
- [29] Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *EMNLP*, Lluís Màrquez, Chris Callison-Burch, Jian Su, Daniele Pighin, and Yuval Marton (Eds.). The Association for Computational Linguistics, 1412–1421.
- [30] Vinod Nair and Geoffrey E. Hinton. 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, Johannes Fürnkranz and Thorsten Joachims (Eds.). 807–814.
- [31] Steffen Rendle. 2019. Evaluation Metrics for Item Recommendation under Sampling. *ArXiv abs/1912.02263* (2019).
- [32] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence* (Montreal, Quebec, Canada) (UAI '09). AUAI Press, Arlington, Virginia, United States, 452–461.
- [33] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John Reidl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*. ACM Press, 285–295. <https://doi.org/10.1145/371920.372071>
- [34] Guy Shani and Asela Gunawardana. 2011. Evaluating Recommendation Systems. In *Recommender Systems Handbook*, Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor (Eds.). Springer, 257–297. https://doi.org/10.1007/978-0-387-85820-3_8
- [35] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research* 15, 1 (2014), 1929–1958.
- [36] Harald Steck. 2013. Evaluation of recommendations: rating-prediction and ranking. In *Proceedings of the 7th ACM conference on Recommender systems*. ACM. <https://doi.org/10.1145/2507157.2507160>
- [37] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management* (Beijing, China) (CIKM '19). Association for Computing Machinery, New York, NY, USA, 1441–1450. <https://doi.org/10.1145/3357384.3357895>
- [38] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved Recurrent Neural Networks for Session-based Recommendations. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems* (Boston, MA, USA) (DLRS 2016). ACM, New York, NY, USA, 17–22. <https://doi.org/10.1145/2988450.2988452>
- [39] Jiayi Tang and Ke Wang. 2018. Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining* (Marina Del Rey, CA, USA) (WSDM '18). Association for Computing Machinery, New York, NY, USA, 565–573. <https://doi.org/10.1145/3159652.3159656>
- [40] Wilson L Taylor. 1953. "Cloze procedure": A new tool for measuring readability. *Journalism quarterly* 30, 4 (1953), 415–433.
- [41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. 5998–6008.
- [42] Andrew Zimdars, David Maxwell Chickering, and Christopher Meek. 2001. Using Temporal Data for Making Recommendations. In *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers

