Robust Training of Sequential Recommender Systems with Missing Input Data*

Federico Siciliano[1,†], Shoval Lagziel[2], Iftah Gamzu[2] and Gabriele Tolomei[1]

[1]Sapienza University of Rome, Rome, Italy
[2]Amazon, Israel

Abstract

In the realm of sequential recommender systems, understanding users' preferences based on their past actions is paramount. Yet, the susceptibility of these models to input perturbations has limited their practicality. Addressing this, we present an innovative approach to mitigate the impact of missing input items, a challenge that has been overlooked. Our method involves a novel training process that anticipates data loss and employs an optimization loss to predict multiple future items. Extensive evaluations on diverse datasets and recommender models underscore its effectiveness. Notably, our approach enhances NDCG@10 by up to 18% with one missing item and an impressive 230% with five missing items, underscoring its substantial impact on system resilience and performance. This work sheds light on the intricate dynamics of sequential recommendation and offers a solution to real-world data limitations.

Keywords

Recommender Systems, Sequential Recommendation, Model Stability

1. Introduction

Sequential recommendation models have raised interest in recent years for their promising increasing performance in various domains such as e-commerce, health and education [1, 2]. However, machine learning models are sensitive to input perturbations [3], and particularly, sequential recommendation models were shown to be vulnerable to even a single change in the training data [4, 5, 6]. The robustness of recommender systems to data perturbations is a desired property and is essential in various domains. Suppose a user regularly uses an e-commerce platform to buy clothes. The platform collects data on the user's past purchases and browsing behavior to make personalized recommendations for future purchases. However, the user decides to take a break from the platform for a few weeks and shops for clothes elsewhere. During this break, the e-commerce platform is unable to collect data on the user's behavior, resulting in missing data. When the user returns to the platform, the recommender system must take into account the missing data and still provide personalized recommendations based on the user's past purchases. Missing data can even be dangerous in some domains, such as healthcare [7], where patients might have been treated at different clinics, and this might result in incorrect diagnoses or treatments. Specifically, in sequential recommendation systems, the recommendation is based on the sequence of user actions, so the most recent actions might have an even stronger effect on the generated recommendations. Considering this, we explore the impact of missing data in the last items of the sequence and how to mitigate it by training the models differently. To the best of our knowledge, this is the first work verifying that existing sequential recommender systems suffer from this effect and applying a method to make sequential recommender models more robust to this type of data perturbation. We can summarise our contributions as follows:

- Our investigation shows that several sequential recommendation models heavily rely on the last items in the sequence.
- We apply a modified training method to make the models more robust to such missing data perturbations.
- Our model outperforms (as measured by Hit Rate and Normalized Discounted Cumulative Gain) classical models in cases of missing data while maintaining or improving performance in the next item prediction task.

2. Related Work

2.1. Sequential Recommendation

Sequential recommendation is a subfield of recommendation systems [8] that focuses on recommending items to users based on their recent interactions. The goal of sequential recommendation is to predict the next item a user will likely interact with, given their previous interactions. One of the earliest sequential recommendation methods is the Markov Chain model [9, 10, 11], which models users' interactions as a Markov process and uses the transition probabilities between items to make recommendations. Recently, there has been a growing interest in using deep learning techniques for sequential recommendation. These methods include using deep neural networks, such as Recurrent Neural Networks (RNN) [12], Long Short-Term Memory (LSTM) [13], Gated Recurrent Units (GRUs) [14, 15] and attention mechanisms [16, 17, 18], to model users' interactions and make recommendations, allowing the model to focus on the most relevant parts of the user's interaction history when making recommendations. Additionally, there has been an increased focus on Explainable AI in sequential recommenders [19, 20], some of which are based on counterfactuals [21, 22, 23, 24, 25], which are aimed at making the recommendations more tailored to the user [26, 27, 28] and providing more transparency into the decision-making

RobustRecSys: Design, Evaluation, and Deployment of Robust Recommender Systems Workshop @ RecSys 2024, 18 October, 2024, Bari, Italy.

*This work was partially supported by project FAIR (PE0000013) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU.

†Work done while at Amazon.

siciliano@diag.uniroma1.it (F. Siciliano); shovall@amazon.com (S. Lagziel); iftah@amazon.com (I. Gamzu); tolomei@di.uniroma1.it (G. Tolomei)

0000-0003-1339-6983 (F. Siciliano); 0000-0002-1657-2076 (S. Lagziel); 0000-0001-7471-6659 (G. Tolomei)

process of the model. Overall, the field of sequential recommendation is rapidly evolving, with a wide range of methods and techniques being proposed and evaluated[29, 30].

2.2. Robustness of recommender systems

One of the main challenges in the field of recommendation systems is ensuring the robustness of the models to data perturbations [31, 32, 33, 34]. Data perturbations refer to small changes in the input data, such as missing values or noisy observations, that can significantly impact the model's performance. Many common recommendation methods are sensitive to such perturbations [4, 5, 35] and can lead to poor performance or even complete failure. Recently, there has been an increased focus on developing robust sequential recommenders that can handle data perturbations. One approach is to use regularization techniques, such as dropout [36, 37], to reduce the impact of noise in the input data. Another approach is to use ensemble methods, such as bagging [38] and boosting, to combine the predictions of multiple models. Another area of research on robustness is the use of generative models, such as Variational Autoencoders (VAEs) [39] or Generative Adversarial Networks (GANs) [40], to learn the underlying distribution of the data and generate new samples, which can be used to augment the training data [41] and improve the robustness of the models. Additionally, there has been work on imputation techniques [42, 43], to infer the missing data to improve recommender systems, and on training instability [44]. Finally, other works focus on methods for evaluating the robustness of the model without using ranking evaluation metrics but rather by assessing the stability of the generated rankings in the presence of missing items in the data [4, 5, 6, 35]. Overall, robustness is a critical issue in sequential recommendation. There are many ongoing efforts to develop methods that can handle data perturbations and improve the performance of the models in practice.

3. Setting

The setting in question consists of N_U users and N_I items. Each user u_i has interacted with at least one item I_j at a given time $t_{i,j}$. The goal of a recommender system is to predict the compatibility between a given user and the items with which it has not yet interacted, knowing the items the user has interacted with. In the Sequential Recommendation case, the problem takes the form of predicting the next item in a sequence: given a sequence of i items $\{I_1, I_2, ..., I_i\}$ with which a user u has interacted, the goal is to predict the $i+1$-th item (I_{i+1}). A sequential neural network takes as input a sequence of at most L elements and performs, for each time step, the prediction of N_I values. These represent the estimated compatibility between user u, to which the sequence of items belong, and *all* items I.

3.1. Classic Training Method

The goal of the network, at each timestep i, is to predict the next item in the sequence I_{i+1}. During network training, if the number of possible items N_I is too large, it becomes intractable to calculate predictions for all of them, therefore, only a chosen few are calculated. In particular, computing the one corresponding to the next item, called *positive item*, and at least one corresponding to an item that is irrelevant, called *negative item*, chosen randomly at each epoch. An attempt is then made to increase the former value at the expense of the latter. Notably, the negative items are indeed chosen randomly, but excluding items already in the input sequence. To achieve this, the loss used for the models we have considered is the Binary Cross-Entropy; this is typically used in a classification scenario.

The definition of loss, positive items, and negative items are defined leads to a specific ranking that the network should achieve at time step i of a sequence of L elements. The ranking can be described to be as follows: first the positive item I_{i+1}, followed by, in indifferent order, all the other items in the sequence I_j such that $j \in \{1, ..., i, i+2, ..., L, L+1\}$, and finally, again in indifferent order, all the remaining items I_j such that $j \notin \{1, 2, ..., L+1\}$. This particular ranking would result in zero loss.

We can simplify, as shown in Figure 1a, the functioning of the network by imagining that for the sequence $[I_1]$, the model should output item I_2, for the sequence $[I_1, I_2]$ it should output item I_3, and so on.

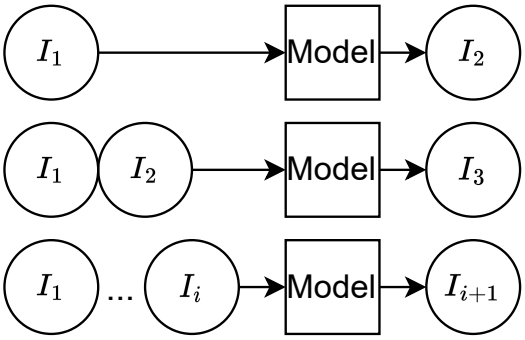
4. Problem Statement

A sequential recommender system receives as input a sequence $S = \{I_1, I_2, ..., I_i\}$ and tries to predict the next item in the sequence, item I_{i+1}. How would the network behave if the last item I_i is missing? If the last item is removed from the sequence $[I_1, I_2]$, we would be left only with the sequence $[I_1]$. Since the network is trained to predict only item I_2, it will have no preference on predicting I_3. Furthermore, considering an out-of-sequence division of the dataset (more info on data division in Section 5.4.3), removing two items from the sequence results in replicating a training sequence. Thus, the effect of removal may be even more detrimental if the model is overfitting on the training set. Lack of user-item interactions in real-world scenarios pose a challenge for sequential recommenders. For example, a streaming service offering a movie trilogy may not have data on a user's interaction with the second movie if it was watched on a different platform. This could result in the recommender suggesting the second movie as the top recommendation without considering the third. This issue also applies to non-sequential items like sequels to movies or books. We start by demonstrating that existing sequential recommender models suffer from this effect, and then we devise a method to make them robust to this type of data perturbation.

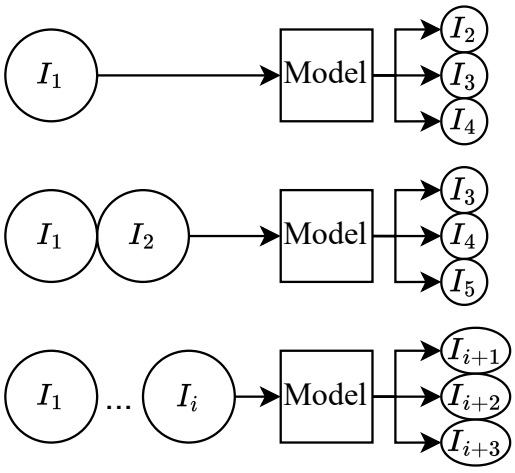
5. Methodology

5.1. More Positive Items

We assume that, in a real scenario, an ideal model should yield, at a given time step i, a ranking containing, in order, all future items in the sequence, and only upon finishing these, all other (negative) items. Therefore, the solution we have applied is to choose N_{pos} positive items, such that the network learns to simultaneously predict N_{pos} future instances. Please note that we are not trying to predict the whole sequence of future interactions but only the relevance of the items at time step i. In this case, the loss would become as in 1.



(a) Simplified visualization of a Sequential Recommender System functioning. Each I_i represents the item at time step i. Each line represents a different inference.



(b) Simplified visualization of a Sequential Recommender System functioning in the case of $N_{pos} = 3$ positives items. Each I_i represents the item at time step i. Each line represents a different inference.

Figure 1: Simplified visualizations in the two different scenarios

$$\ell_{\text{BCE,mp}}(\vec{x} \mid \vec{pos}, \vec{neg}) = -\sum_{j \in \vec{pos}} \log(x_j) - \sum_{j \in \vec{neg}} \log(1 - x_j) \tag{1}$$

where $\vec{x}$ represents the output of the network, $\vec{pos} = \{p_1, ..., p_{N_{pos}}\}$ the identifier of the N_{pos} positive items, and $\vec{neg} = \{n_1, ..., n_{N_{neg}})\}$ those of the selected N_{neg} negative items. The loss takes the same form as that presented in [45]. Although the authors mention the loss function, its potential for improving the model's robustness in the face of missing data has not been explored. Our work fills this gap by showing how this loss function can be effectively used to increase the robustness of the model and improve its performance in the presence of missing data. Replicating the simplified illustration in Figure 1a, we can visualize the idea of predicting multiple positive items as in Figure 1b.

5.2. Margin Loss

Considering more positive items poses a clear limitation, as it becomes more challenging for the next item to rank high in the network's ranking. This is because the Binary Cross-Entropy loss does not distinguish between positive items; a perfect model would rank all P positive items in the first P positions, regardless of their order. This might limit the model performance, as the item may end-up in the P-th position, thus reducing common metrics that take into account the order of results, such as NDCG. To solve this problem, we decide to use the Margin Loss. Given pairs of inputs x_1 and x_2, and a preferred ordering of them y, such that $y = 1$ if we assume that the first input should be ranked higher than the second input, vice-versa for $y = -1$, the margin loss ℓ takes values according to $\ell_{\text{MRG}}(x_1, x_2, y \mid margin) = \max(0, y(x_2 - x_1) + margin)$. This tells us that if the network outputs for the two inputs respect the expected ordering and are at least $margin$ apart, the loss is equal to 0; otherwise, it is proportional to the distance between them. Input pairs are formed between all pairs of positive items. The expected order is the order in which the user interacted with them: an item at a time step i must come first in ranking than one at a time step $i + k$. The Margin Loss formula is $\ell_{\text{MRG,pos}}(\vec{x} \mid \vec{pos}, margin) = \sum_{c=1}^{N_{pos}} \sum_{k=c+1}^{N_{pos}} \max(0, x_{p_k} - x_{p_c} + margin)$, where $\vec{x}$ represents the output of the network, $\vec{pos} = \{p_1, ..., p_{N_{pos}})\}$ the identifier of the N_{pos} positive items and $margin$ the margin value. The equation holds only if the order of the identifiers of the positive elements follows the expected order.

5.3. Mixed Loss

The margin loss applied on the positive items is not enough to train the neural network as we desire. It is always necessary to discourage the model from predicting negative items. We, therefore, decide to use it in conjunction with the traditional Cross-Entropy loss. This naturally brings up the need to add some hyperparameters to weigh the importance of the two losses. We also separate the components of the Binary Cross-Entropy loss pertaining to positive items and negative items. This Mixed Loss formula is $\ell_{\text{MIX}}(\vec{x} \mid \vec{pos}, \vec{neg}, margin) = l_{\text{BCE,pos}} + \lambda_1 l_{\text{BCE,neg}} + \lambda_2 \ell_{\text{MRG,pos}}(\vec{x} \mid \vec{pos}, margin)$.

5.4. Experiments

5.4.1. Datasets

We select three datasets that are widely used in this field [46, 47]: MovieLens-1M [48], MovieLens-100K [48] and Amazon Beauty [49]. The first two are movie ratings taken from the MovieLens website[1] and differ on the period they were collected and the size of the set. The third dataset[2] contains reviews and metadata from Amazon, spanning May 1996 - Oct 2018. The three datasets have 165, 106 and 5 interactions per user, respectively. The statistics for all the considered datasets are shown in Table 1.

[1] https://movielens.org
[2] https://nijianmo.github.io/amazon/index.html

Table 1
Dataset statistics after preprocessing

Dataset	Users	Items	Actions /User Average	Actions /User Median	Actions
MovieLens 1M	6040	3706	165	96	1M
MovieLens 100k	943	1682	106	65	100K
Amazon Beauty	2417	2821	5	5	12K

5.4.2. Models

We select three sequential recommendation models. The first, GRU4REC [15], is an RNN based on Gated Recurrent Unit. SASRec [16], on the other hand, is a sequential self-attention based model that uses an attention mechanism to make predictions based on a relatively small number of actions. TiSASRec (Time Interval aware Self-attention based sequential recommendation) [17] is instead a modification of this that adds to the input the time intervals between elements in the sequence.

5.4.3. Preprocessing

Consistent with other work, we use implicit ratings, so we do not consider the score but simply the existence of an interaction of a given user with a given product. Given a user u, the products he interacted with are ordered in a sequence S_u based on the timestamp. An out-of-sequence split (i.e. the last two items in each sequence are kept aside to be the target output of validation and test, respectively, while the rest of the sequence is used for training) is performed to partition the data into training, validation and test sets, in line with what has been done by other works in the same domain.

5.4.4. Evaluation

In line with what has been done in other works involving Neural Recommenders [16], in order to avoid to avoid heavy computation, the evaluation is carried out in the following way: the prediction made by the network is taken for the positive item (the next item in the sequence) and 100 items chosen randomly, not in the input sequence. The predictions (for 100 negative items + the positive item) are then sorted according to the values obtained; this represents the final ranking.

We want to emphasize that while we use multiple positive items during training, this is not done during the evaluation phase. The reason for this decision is that changing the evaluation method could naturally result in our proposed losses appearing better, thus rendering the comparison invalid. By adhering to the traditional evaluation setting, we align ourselves with the evaluation methods used in other works in this field. However, we acknowledge that this places our proposed method at a disadvantage compared to the baseline method for obvious reasons. We are training the model to predict multiple items to increase its robustness, but only one of these items will be used during evaluation. On the other hand, the baseline model focuses solely on a single positive item, the same one used for evaluation, which inherently gives it an advantage. In Section 6, we will demonstrate how our model still manages to achieve superior results. In addition to the standard metrics, to evaluate the sensitivity of the models in cases of input data perturbations, we utilized the recently introduced metric Rank List Sensitivity (RLS)[4], enabling to compare rankings produced with and without perturbations. RLS is defined as $RLS = \frac{1}{N}\sum_i^N sim\left(R_{A,i}, R_{B,i}\right)$. where N is the number of samples, sim is a similarity function, $R_{A,i}$ and $R_{B,i}$ are two rankings produced for sample i. In our specific case, A represents the ranking when sample i is unaltered, while B represents the case when the input sequence is perturbed, i.e. items are removed. The similarity (sim) of two rankings R_a and R_b can be calculated using the Jaccard similarity [50], but it does not consider order. On the other hand, Rank-biased Overlap (RBO) is more valuable for a recommendation system as it considers top-ranked items as more significant using specific weighting (see Equation 2).

$$
\begin{aligned}
JAC(R_A, R_B) &= \frac{|R_A \cap R_B|}{|R_A \cup R_B|} \\
RBO(R_a, R_b) &= (1-p)\sum_{i=1}^{k} p^{i-1}\frac{|R_A[1:i] \cap R_B[1:i]|}{i}
\end{aligned}
\tag{2}
$$

5.4.5. Hyperparameter Optimization

The hyper-parameters to be optimised are the number of positive items to be used, the number of negative items to be used and the Mixed Loss parameters. The number of positive items N_{pos} and negative items N_{neg} varies in the set $\{1, 3, 10\}$, and the Mixed Loss parameters λ_1 and λ_2 in the set $\{1, 10^{-1}, ..., 10^{-5}\}$.

5.4.6. Implementation

All code is written in Python 3. In particular, with Pytorch and Pytorch Lightning.

6. Results

In this section, we present experimental results showing the strong reliance of sequential recommender models on the last items in the sequence as well as the performance of the proposed training method to mitigate this effect.

6.1. Last Items Importance

Figure 2a visualizes the effect of removing an item at different positions in the sequence on the model outputs, using the SASRec model and the MovieLens-1M dataset, has on the ranking of the top item. We identify this item with the term *previous top-ranked item*: that item which, prior to the input data perturbation, was at the top of the ranking. In the case of the base model, removing the last item can push the previously top-ranked item by over 25 positions on average. While SASRec is trained using dropout, this does not seem to be sufficient to make it robust to missing data. In contrast, when the model is trained with more positive items, the removal of the last item results in a significantly lower drop in the ranking of the previous top item: 5 positions or less. We also observe that the difference between the different models becomes less pronounced as we move towards the earlier items in the sequence. These results

Table 2
Results in terms of ranking evaluation (NDCG@10 and HR@10) and robustness metrics (RLS with Jaccard or RBO) for GRU4Rec model and the considered datasets, varying the number of items removed from the end of the sequence. To assist visualization leading zeroes are removed.

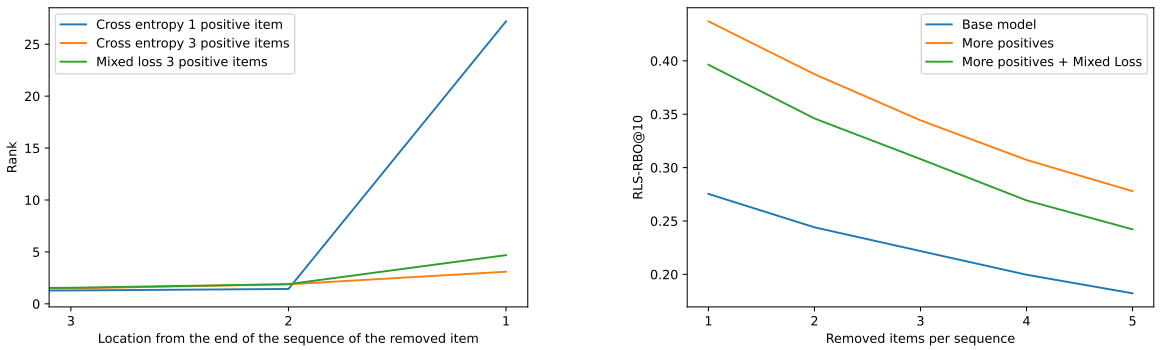
Dataset	Missing Items	NDCG@10			HR@10			RLS-JAC@10			RLS-RBO@10		
		Base	MP	MP+ML	Base	MP	MP+ML	Base	MP	MP+ML	Base	MP	MP+ML
ML-1M	0	.4227	.4706	**.4795**	.6618	.7098	**.7134**	—	—	—	—	—	—
	1	.3898	.4368	**.4498**	.6356	.6843	**.6873**	.0489	**.0867**	.0794	.0313	**.0550**	.0489
	2	.3635	.4101	**.4180**	.6162	.6579	**.6606**	.0442	**.0752**	.0676	.0293	**.0475**	.0403
	3	.3482	.3983	**.4044**	.5881	.6452	**.6475**	.0392	**.0672**	.0604	.0252	**.0415**	.0358
	4	.3257	**.3748**	.3737	.5642	**.6214**	.6167	.0376	**.0628**	.0552	.0241	**.0398**	.0334
	5	.3099	.3608	**.3611**	.5440	**.6081**	.6038	.0347	**.0568**	.0502	.0219	**.0346**	.0298
ML-100k	0	.3621	**.4004**	.3976	.6182	.6607	**.6713**	—	—	—	—	—	—
	1	.3182	.3730	**.3772**	.5705	.6288	**.6490**	.1216	.2284	**.2292**	.0809	**.1689**	.1680
	2	.3253	.3459	**.3558**	.5779	.6161	**.6193**	.1181	**.2123**	.2214	.0756	.1547	**.1606**
	3	.3145	**.3381**	.3364	.5493	.5875	**.5938**	.1102	.2047	**.2060**	.0712	**.1524**	.1517
	4	.2843	.3323	**.3352**	.5154	**.5907**	.5843	.1084	.1947	**.1959**	.0689	.1433	**.1435**
	5	.2833	.3252	**.3410**	.5080	.5663	**.5832**	.0967	**.1826**	.1801	.0624	.1304	**.1312**
Amazon Beauty	0	.4683	.4656	**.4687**	**.5114**	.5077	.5060	—	—	—	—	—	—
	1	.4539	.4517	**.4623**	**.5072**	.4969	.5056	**.7110**	.6633	.6780	**.4987**	.4858	.4676
	2	.4471	.4499	**.4531**	.5027	.4990	**.5064**	**.6472**	.5829	.6100	**.4793**	.4550	.4386
	3	.3500	**.3955**	.3938	.4059	.4688	**.4737**	**.4059**	.4026	.3827	.2947	**.3185**	.2915
	4	.1325	**.2774**	.2626	.1849	**.3620**	.3562	.2413	**.3298**	.2390	.1400	**.2017**	.1783
	5	.1191	**.2662**	.1966	.1676	**.3442**	.2892	.1583	**.2092**	.1164	.0773	**.1122**	.0930

Table 3
Results in terms of ranking evaluation (NDCG@10 and HR@10) and robustness metrics (RLS with Jaccard or RBO) for SASRec model and the considered datasets, varying the number of items removed from the end of the sequence. To assist visualization leading zeroes are removed.

Dataset	Missing Items	NDCG@10			HR@10			RLS-JAC@10			RLS-RBO@10		
		Base	MP	MP+ML	Base	MP	MP+ML	Base	MP	MP+ML	Base	MP	MP+ML
ML-1M	**0**	**.5969**	.5772	.5874	**.8222**	.8142	.8207	—	—	—	—	—	—
	1	**.5572**	.5490	.5570	.7925	.7925	**.7962**	.4116	**.6157**	.5584	.2754	**.4370**	.3964
	2	.5292	.5274	**.5326**	.7768	.7748	**.7783**	.3625	**.5348**	.4849	.2441	**.3875**	.3460
	3	.5090	.5028	**.5108**	**.7647**	.7594	.7634	.3276	**.4731**	.4287	.2218	**.3443**	.3080
	4	.4838	.4906	**.4931**	.7452	.7425	**.7520**	.2933	**.4215**	.3741	.1997	**.3072**	.2693
	5	.4691	.4752	**.4795**	.7316	.7344	**.7411**	.2676	**.3778**	.3368	.1823	**.2779**	.2422
ML-100k	**0**	**.4559**	.4456	.4527	.7349	.7349	**.7455**	—	—	—	—	—	—
	1	.4200	.4219	**.4357**	**.7243**	.6988	.7232	.2734	**.6457**	.5411	.1668	**.4636**	.3732
	2	.4196	.4113	**.4282**	.6999	.6978	**.7179**	.2565	**.6217**	.5179	.1599	**.4458**	.3615
	3	.3792	**.4032**	.4008	.6607	.6861	**.6935**	.2441	**.5916**	.4920	.1508	**.4268**	.3448
	4	.3878	.3855	**.4036**	.6755	.6734	**.6925**	.2350	**.5565**	.4707	.1450	**.4083**	.3333
	5	.3632	.3764	**.3896**	.6511	.6670	**.6797**	.2226	**.5261**	.4369	.1409	**.3869**	.3106
Amazon Beauty	**0**	**.4682**	.4597	.4643	.5038	**.5075**	.5067	—	—	—	—	—	—
	1	.4499	.4496	**.4502**	.4959	.5032	**.5037**	.5730	.6087	**.6164**	.4076	**.4332**	.4238
	2	.4474	**.4500**	.4461	.5034	.5053	**.5077**	.5123	.5314	**.5375**	.3827	**.4152**	.3954
	3	.3850	**.4273**	.4257	.4510	**.5108**	.5086	.3967	**.4531**	.4442	.3043	**.3585**	.3335
	4	.2317	.3582	**.3640**	.3055	.4759	**.4879**	.2987	**.3934**	.3800	.2193	**.2909**	.2818
	5	.2193	.3549	**.3593**	.2931	.4763	**.4878**	.2579	**.3718**	.3602	.1806	.2522	**.2563**

Table 4
Results in terms of ranking evaluation (NDCG@10 and HR@10) and robustness metrics (RLS with Jaccard or RBO) for TiSASRec model and the considered datasets, varying the number of items removed from the end of the sequence. To assist visualization leading zeroes are removed.

Dataset	Missing Items	NDCG@10			HR@10			RLS-JAC@10			RLS-RBO@10		
		Base	MP	MP+ML	Base	MP	MP+ML	Base	MP	MP+ML	Base	MP	MP+ML
ML-1M	0	**.5494**	.5348	.5402	**.7823**	.7760	.7773	—	—	—	—	—	—
	1	**.5159**	.5086	.5148	.7523	.7522	**.7550**	.3622	**.5714**	.5605	.2373	**.4069**	.3946
	2	.4906	.4920	**.4997**	**.7387**	.7386	.7444	.3070	.4968	**.4769**	.2000	**.3592**	.3396
	3	.4667	.4743	**.4808**	.7166	.7242	**.7311**	.2685	**.4401**	.4202	.1747	**.3215**	.3028
	4	.4522	.4584	**.4662**	.7043	.7144	**.7194**	.2348	**.3891**	.3669	.1536	**.2880**	.2649
	5	.4338	.4436	**.4517**	.6889	.7070	**.7126**	.2072	**.3488**	.3270	.1347	**.2590**	.2369
ML-100k	0	**.4154**	.3984	.4044	**.6935**	.6702	.6670	—	—	—	—	—	—
	1	.3665	.3922	**.3972**	.6394	.6426	**.6490**	.2704	**.5377**	.5238	.1638	**.3869**	.3708
	2	.3632	**.3792**	.3761	.6108	.6225	**.6246**	.2406	**.4855**	.4735	.1424	**.3547**	.3394
	3	.3477	.3637	**.3642**	**.6182**	.6172	**.6182**	.2194	**.4546**	.4418	.1292	**.3360**	.3186
	4	.3413	**.3568**	.3535	.5938	**.6161**	.6151	.2020	**.4227**	.4113	.1194	**.3145**	.2959
	5	.3321	.3481	**.3529**	.5822	.6034	**.6076**	.1965	**.4004**	.3858	.1163	**.3021**	.2819
Amazon Beauty	0	.4670	**.4700**	.4693	.4994	**.5106**	.5077	—	—	—	—	—	—
	1	.4467	.4564	**.4600**	.4911	**.5077**	.5056	.5819	**.6580**	.6559	.3931	**.4641**	.4575
	2	.4426	**.4553**	.4542	.4944	**.5101**	**.5101**	.5402	.6353	**.6403**	.3765	**.4598**	.4563
	3	.3667	**.4297**	.4286	.4241	.5097	**.5130**	.4135	**.5438**	.5512	.3096	**.3784**	.3676
	4	.1606	.3453	**.3469**	.2122	**.4886**	.4878	.2567	**.4031**	.3969	.2155	**.2892**	.2655
	5	.1483	**.3409**	.3345	.1953	**.4882**	.4866	.1926	**.3183**	.3013	.1831	**.2338**	.2015



(a) Rank of the previous top-ranked item when removing an item from the input sequence in a specific position

(b) Rank List Sensitivity using Rank-Biased Overlap@10 for SASRec Model on MovieLens-1M dataset with different number of missing items

Figure 2: The effect of removing the last items in a sequence with three training methods

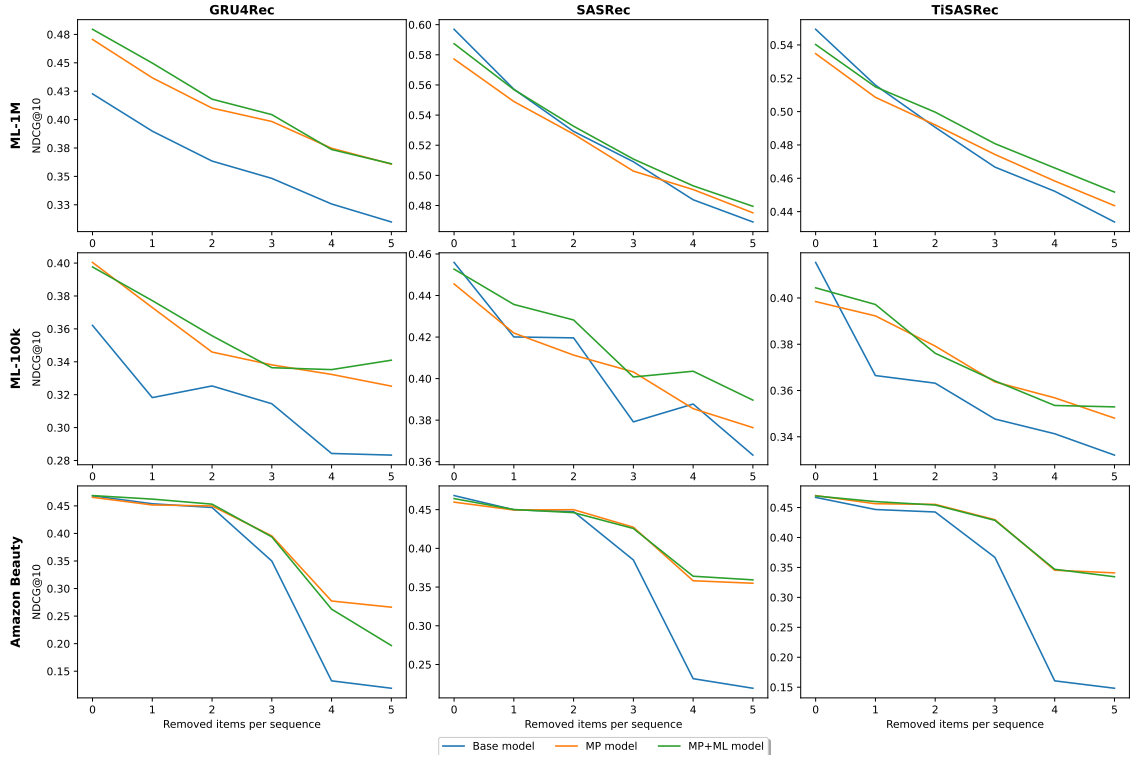


Figure 3: NDCG@10 with different number of missing items for each model-dataset pair

demonstrate that incorporating more positive items in the training process and using our proposed Mixed Loss can help to mitigate the impact of missing data and improve the robustness of Sequential Recommender Systems.

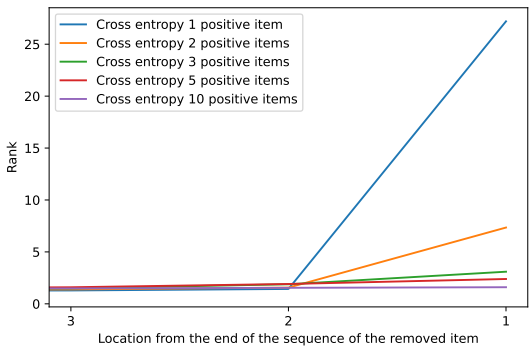
6.2. Performance of Different Training Methods in Cases of Missing Last Items

NDCG@10 score is used to gauge the impact of the modified training method on the performance of the models. Figure 3 provides a visualization of the results. The results are also expressed integrally in Tables 2, 3 and 4.

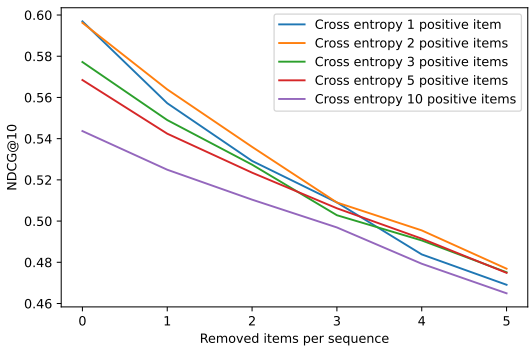
6.2.1. A Clear Advantage in Handling Missing Data

One striking observation is that the models trained with more positive items and the Mixed Loss consistently outperform the base model when it comes to dealing with missing data. Although the base model performs slightly better in the absence of missing data, the new models are able to sustain their performance even as the number of missing items increases. This is especially evident in the case of the Amazon Beauty dataset, where the new training method is able to maintain acceptable performance as the missing data becomes more prominent.

As mentioned in Section 5.4.4, the slight predominance of the base model in the absence of missing data is expected because the evaluation setting naturally favors the base model: both the loss function and the evaluation technique consider



(a) Rank of the previous top-ranked item when removing an item from the input sequence in a specific position



(b) NDCG@10 for SASRec Model on MovieLens-1M dataset with different number of missing items

Figure 4: Study on the Number of Positives

only a single item. We emphasize the significance that our model, trained in a manner that deviates slightly from the traditional evaluation setting, is able to retain minimal performance loss in the same setting while gaining robustness to missing data.

6.2.2. Length of sequences

It is worth noting that the average sequence length of the three datasets is vastly different (see Section 5.4.1). The results in Figure 3 indicate that the impact of missing data is much less severe for datasets with longer sequences, such as ML-1M. The model trained with the classic training method is even able to compensate for this deficiency, particularly in the case of SASRec. However, as the average sequence length decreases, such as in the ML-100k dataset, the robustness to missing data seems to decline rapidly, and the difference between the models becomes more pronounced when the number of missing items increases. This trend is especially evident in the Amazon Beauty dataset, where the difference between the models is particularly noticeable when the number of missing items is higher than 2, probably because the average length of the sequences for this dataset is 5.

6.3. Rank List Stability

The robustness of the models in the face of item removal at the end of the input sequence is illustrated through the Rank List Stability with Rank-biased Overlap metric in Figure 2b. It is evident that the new models exhibit higher stability, with Cross-Entropy with more positive items proving to be even more robust than the Mixed Loss model. We observed a similar trend for all datasets and models, so only one plot is presented; further results can be found in the additional repository. While the multiple positive model (MP) provides in most cases higher performance in the Rank List Stability metrics compared to the model with multiple positive and the mixed loss (MP+ML), it is worth noting that MP+ML provides higher performance on the HR@10 and NDCG@10. This can be explained by the fact that MP is not optimized using the ranks of the positive items as done by the mixed loss (MP+ML model). However, for precisely the same reason, MP benefits from higher stability.

More specifically, both models are trained to predict, at time t and for a given input sequence, N_{pos} positive items, specifically $[p_t, p_{t+1}, ..., p_{t+N_{pos}}]$. However, the MP model is trained with a loss function that does not consider the order of the positive items: the same sequence in reverse order would yield the same loss value. As discussed in Section 5.4.4, in the classic evaluation setting, only p_t is used during evaluation. If the loss function treats p_t equally important as the other positive items $[p_{t+1}, ..., p_{t+N_{pos}}]$, it is more likely to be ranked lower, thus reducing metrics such as NDCG and Recall. On the other hand, the model using the Mixed Loss, which aims to prioritize the position of p_t at the top of the ranking, has an advantage in achieving higher metrics in this regard.

6.4. Study on the Number of Positives

To understand the impact of the number of positive items used for training, experiments were performed using different numbers of positive items for just one model, SasRec, and one dataset, MovieLens-1M, due to the computational time required. As seen in Figure 4a, as the number of positive items increases, the change in ranking for previous top-ranked items decreases significantly. However, Figure 4b shows that the performance in the absence of missing data degrades as the number of positive items increases. This trend begins to change as the number of missing items increases, and the gap between the new models and the base model narrows, with the latter's performance deteriorating more.

7. Implications of the Research Findings

The findings of this study hold both theoretical and practical implications that contribute to the advancement of sequential recommender systems and their application in real-world scenarios. By addressing the specific challenges posed by missing input data, our research offers a novel perspective on enhancing the robustness and reliability of these systems.

7.1. Theoretical Implications

1. **Uncovering Last-Item Dependence:** Our research uncovers the strong reliance of sequential recommender systems on the last items in the input se-

quence. This revelation contributes to a deeper understanding of the dynamics within these systems, emphasizing the need for strategies that can mitigate the performance degradation caused by missing items.

2. **New Training Paradigm:** The introduction of a training approach that anticipates data loss and simulates prediction of multiple future items presents a paradigm shift in the methodology for handling missing input data. This approach establishes a theoretical foundation for designing more resilient recommender systems.

7.2. Practical Implications

1. **Real-World Data Challenges:** In real-world scenarios, complete user action sequences are often not available due to various constraints. Our research highlights the practical significance of addressing this data scarcity and provides a concrete solution to mitigate the negative effects of missing items, improving the usability of recommender systems.
2. **Enhanced System Resilience:** The proposed training method significantly improves the performance of sequential recommender systems when faced with missing items. This directly translates into a more reliable and user-centric experience, thus benefiting various domains, such as e-commerce, content recommendation, and personalized services.
3. **Impact on User Satisfaction:** The performance enhancement demonstrated by our approach can lead to improved user satisfaction by providing more accurate and relevant recommendations, even when there are gaps in the available data. This practical outcome can foster greater user engagement and loyalty.
4. **General Applicability:** The effectiveness of our method across various datasets and recommender models underscores its general applicability. This widens its potential adoption and impact, making it a valuable tool for researchers and practitioners alike.

8. Discussion and Conclusions

Our findings show that the last items in a sequence have a significant impact on the predictions of sequential recommenders, and their removal results in unstable rankings. However, by incorporating multiple future items in the training process, model robustness can be improved. Our results demonstrate that the proposed training methods improve rankings stability (RLS metric) and performance (HR and NDCG) on various popular sequential recommender models (SasRec[16], TiSasRec[17], and GRU4Rec[15]) and datasets. In contrast, the performance without missing data is not noticeably affected but even improves for specific models/datasets. Using more positive items with Cross-Entropy loss improves robustness of sequential recommenders to removal of elements at the end of the input sequence. However, increasing the number of future items excessively can lead to stability increase at the cost of decreased performance. Mixed Loss, combining Cross-Entropy with Margin Loss, can prioritize the next item over other positives. Our method opens up opportunities for further research in the field. Future work may focus on the development of a loss function that balances performance and robustness as the number of positive items increases, as well as modifying the method for models that use bi-directional connections (e.g., [18]). Moreover, out proposal is easily extendable to other approaches, as it is solely tied to a different training method and not to a specific architecture. To summarize, our work represents a step forward in improving the robustness of sequential recommender models. We demonstrate the strong influence of the last items in a sequence and the effectiveness of our method in mitigating the impact of missing data. Overall, we expect that our findings and proposed methods will be a valuable tool in the field of sequential recommender systems.

References

[1] M. Quadrana, P. Cremonesi, D. Jannach, Sequence-aware recommender systems, ACM Computing Surveys (CSUR) 51 (2018) 1–36.

[2] I. Brilhante, J. A. Macedo, F. M. Nardini, R. Perego, C. Renso, Tripbuilder: A tool for recommending sightseeing tours, in: Advances in Information Retrieval: 36th European Conference on IR Research, ECIR 2014, Amsterdam, The Netherlands, April 13-16, 2014. Proceedings 36, Springer, 2014, pp. 771–774.

[3] S. Gupta, A. Gupta, Dealing with noise problem in machine learning data-sets: A systematic review, Procedia Computer Science 161 (2019) 466–474.

[4] S. Oh, B. Ustun, J. McAuley, S. Kumar, Rank list sensitivity of recommender systems to interaction perturbations, in: Proceedings of the 31st ACM International Conference on Information & Knowledge Management, 2022, pp. 1584–1594.

[5] S. Oh, S. Kumar, Robustness of deep recommendation systems to untargeted interaction perturbations, arXiv preprint arXiv:2201.12686 (2022).

[6] B. Filippo, S. Federico, M. Pushkar, S. Fabrizio, Investigating the robustness of sequential recommender systems against training data perturbations, in: Advances in Information Retrieval: 46th European Conference on Information Retrieval (ECIR 2024), Springer, 2024, pp. 205–220. URL: https://doi.org/10.1007/978-3-031-28241-6_14. doi:10.1007/978-3-031-28241-6_14, first Online: 16 March 2024.

[7] T. N. T. Tran, A. Felfernig, C. Trattner, A. Holzinger, Recommender systems in the healthcare domain: state-of-the-art and research issues, Journal of Intelligent Information Systems 57 (2021) 171–201.

[8] F. Ricci, L. Rokach, B. Shapira, Introduction to recommender systems handbook, in: Recommender systems handbook, Springer, 2011, pp. 1–35.

[9] F. Fouss, S. Faulkner, M. Kolp, A. Pirotte, M. Saerens, et al., Web recommendation system based on a markov-chainmodel., in: ICEIS (4), 2005, pp. 56–63.

[10] G. Shani, D. Heckerman, R. I. Brafman, C. Boutilier, An mdp-based recommender system., Journal of Machine Learning Research 6 (2005).

[11] Y. Ren, M. Tomko, F. D. Salim, J. Chan, C. L. Clarke, M. Sanderson, A location-query-browse graph for contextual recommendation, IEEE Transactions on Knowledge and Data Engineering 30 (2017) 204–218.

[12] B. Hidasi, A. Karatzoglou, Recurrent neural networks

with top-k gains for session-based recommendations, in: Proceedings of the 27th ACM international conference on information and knowledge management, 2018, pp. 843–852.

[13] L. Yang, Y. Zheng, X. Cai, H. Dai, D. Mu, L. Guo, T. Dai, A lstm based model for personalized context-aware citation recommendation, IEEE access 6 (2018) 59618–59627.

[14] J. Chung, C. Gulcehre, K. Cho, Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, arXiv preprint arXiv:1412.3555 (2014).

[15] B. Hidasi, A. Karatzoglou, L. Baltrunas, D. Tikk, Session-based recommendations with recurrent neural networks, arXiv preprint arXiv:1511.06939 (2015).

[16] W.-C. Kang, J. McAuley, Self-attentive sequential recommendation, in: 2018 IEEE international conference on data mining (ICDM), IEEE, 2018, pp. 197–206.

[17] J. Li, Y. Wang, J. McAuley, Time interval aware self-attention for sequential recommendation, in: Proceedings of the 13th international conference on web search and data mining, 2020, pp. 322–330.

[18] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, P. Jiang, Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer, in: Proceedings of the 28th ACM international conference on information and knowledge management, 2019, pp. 1441–1450.

[19] I. Nunes, D. Jannach, A systematic review and taxonomy of explanations in decision support and recommender systems, User Modeling and User-Adapted Interaction 27 (2017) 393–444.

[20] Y. Zhang, X. Chen, et al., Explainable recommendation: A survey and new perspectives, Foundations and Trends® in Information Retrieval 14 (2020) 1–101.

[21] A. Ghazimatin, O. Balalau, R. Saha Roy, G. Weikum, Prince: Provider-side interpretability with counterfactual explanations in recommender systems, in: Proceedings of the 13th International Conference on Web Search and Data Mining, 2020, pp. 196–204.

[22] J. Tan, S. Xu, Y. Ge, Y. Li, X. Chen, Y. Zhang, Counterfactual explainable recommendation, in: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, 2021, pp. 1784–1793.

[23] K. H. Tran, A. Ghazimatin, R. Saha Roy, Counterfactual explanations for neural recommenders, in: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2021, pp. 1627–1631.

[24] Z. Chen, F. Silvestri, J. Wang, Y. Zhang, G. Tolomei, The dark side of explanations: Poisoning recommender systems with counterfactual examples, arXiv preprint arXiv:2305.00574 (2023).

[25] Z. Chen, F. Silvestri, J. Wang, Y. Zhang, Z. Huang, H. Ahn, G. Tolomei, Grease: Generate factual and counterfactual explanations for gnn-based recommendations, arXiv preprint arXiv:2208.04222 (2022).

[26] X. Wang, Y. Chen, J. Yang, L. Wu, Z. Wu, X. Xie, A reinforcement learning framework for explainable recommendation, in: 2018 IEEE international conference on data mining (ICDM), IEEE, 2018, pp. 587–596.

[27] Z. Wang, J. Zhang, H. Xu, X. Chen, Y. Zhang, W. X. Zhao, J.-R. Wen, Counterfactual data-augmented sequential recommendation, in: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2021, pp. 347–356.

[28] S. Zhang, D. Yao, Z. Zhao, T.-S. Chua, F. Wu, Causerec: Counterfactual user sequence synthesis for sequential recommendation, in: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2021, pp. 367–377.

[29] A. Sbandi, F. Siciliano, F. Silvestri, Mitigating extreme cold start in graph-based recsys through reranking, in: Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, CIKM '24, Association for Computing Machinery, New York, NY, USA, 2024, p. 4844–4851. URL: https://doi.org/10.1145/3627673.3680069. doi:10.1145/3627673.3680069.

[30] F. Betello, A. Purificato, F. Siciliano, G. Trappolini, A. Bacciu, N. Tonellotto, F. Silvestri, A reproducible analysis of sequential recommender systems, IEEE Access 13 (2025) 5762–5772. doi:10.1109/ACCESS.2024.3522049.

[31] R. Burke, M. P. O'Mahony, N. J. Hurley, Robust collaborative recommendation, in: Recommender systems handbook, Springer, 2015, pp. 961–995.

[32] M. O'Mahony, N. Hurley, N. Kushmerick, G. Silvestre, Collaborative recommendation: A robustness analysis, ACM Transactions on Internet Technology (TOIT) 4 (2004) 344–377.

[33] Y. Deldjoo, T. Di Noia, E. Di Sciascio, F. A. Merra, How dataset characteristics affect the robustness of collaborative recommendation models, in: Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval, 2020, pp. 951–960.

[34] V. Guarrasi, F. Siciliano, F. Silvestri, Robustrecsys @ recsys2024: Design, evaluation and deployment of robust recommender systems, in: Proceedings of the 18th ACM Conference on Recommender Systems, RecSys '24, Association for Computing Machinery, New York, NY, USA, 2024, p. 1265–1269. URL: https://doi.org/10.1145/3640457.3687106. doi:10.1145/3640457.3687106.

[35] F. Betello, F. Siciliano, P. Mishra, F. Silvestri, Finite rank-biased overlap (frbo): A new measure for stability in sequential recommender systems, in: Proc. of the 14th Italian Information Retrieval Workshop, volume 3802, 2024, pp. 78–81.

[36] Y. Gal, Z. Ghahramani, Dropout as a bayesian approximation: Representing model uncertainty in deep learning, in: international conference on machine learning, PMLR, 2016, pp. 1050–1059.

[37] Y. Li, W. Ma, C. Chen, M. Zhang, Y. Liu, S. Ma, Y. Yang, A survey on dropout methods and experimental verification in recommendation, arXiv preprint arXiv:2204.02027 (2022).

[38] J. Vinagre, A. M. Jorge, J. Gama, Online bagging for recommender systems, Expert Systems 35 (2018) e12303.

[39] D. Liang, R. G. Krishnan, M. D. Hoffman, T. Jebara, Variational autoencoders for collaborative filtering, in: Proceedings of the 2018 world wide web conference, 2018, pp. 689–698.

[40] Q. Wu, Y. Liu, C. Miao, B. Zhao, Y. Zhao, L. Guan, Pdgan: Adversarial learning for personalized diversity-promoting recommendation., in: IJCAI, volume 19, 2019, pp. 3870–3876.

[41] D. Liu, Y. Sun, X. Zhao, G. Zhang, R. Liu, Adversarial

training for session-based item recommendations, in: 2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC), volume 9, IEEE, 2020, pp. 1162–1168.
[42] S.-Y. Ihm, S.-E. Lee, Y.-H. Park, A. Nasridinov, M. Kim, S.-H. Park, A technique of recursive reliability-based missing data imputation for collaborative filtering, Applied Sciences 11 (2021) 3719.
[43] W. Xia, L. He, J. Gu, K. He, Effective collaborative filtering approaches based on missing data imputation, in: 2009 Fifth International Joint Conference on INC, IMS and IDC, IEEE, 2009, pp. 534–537.
[44] J. Tang, Y. Drori, D. Chang, M. Sathiamoorthy, J. Gilmer, L. Wei, X. Yi, L. Hong, E. H. Chi, Improving training stability for multitask ranking models in recommender systems, arXiv preprint arXiv:2302.09178 (2023).
[45] J. Tang, K. Wang, Personalized top-n sequential recommendation via convolutional sequence embedding, in: Proceedings of the eleventh ACM international conference on web search and data mining, 2018, pp. 565–573.
[46] J. Y. Chin, Y. Chen, G. Cong, The datasets dilemma: How much do we really know about recommendation datasets?, in: Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining, 2022, pp. 141–149.
[47] K. Ong, S.-C. Haw, K.-W. Ng, Deep learning based-recommendation system: An overview on models, datasets, evaluation metrics, and future trends, in: Proceedings of the 2019 2nd International Conference on Computational Intelligence and Intelligent Systems, 2019, pp. 6–11.
[48] F. M. Harper, J. A. Konstan, The movielens datasets: History and context, Acm transactions on interactive intelligent systems (tiis) 5 (2015) 1–19.
[49] J. Ni, J. Li, J. McAuley, Justifying recommendations using distantly-labeled reviews and fine-grained aspects, in: Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP), 2019, pp. 188–197.
[50] P. Jaccard, The distribution of the flora in the alpine zone. 1, New phytologist 11 (1912) 37–50.