Integrating Item Relevance in Training Loss for Sequential Recommender Systems

ANDREA BACCIU, Sapienza University of Rome, Italy

FEDERICO SICILIANO, Sapienza University of Rome, Italy

NICOLA TONELLOTTO, University of Pisa, Italy

FABRIZIO SILVESTRI, Sapienza University of Rome, Italy

Sequential Recommender Systems (SRSs) are a popular type of recommender system that learns from a user's history to predict the next item they are likely to interact with. However, user interactions can be affected by noise stemming from account sharing, inconsistent preferences, or accidental clicks. To address this issue, we (i) propose a new evaluation protocol that takes multiple future items into account and (ii) introduce a novel relevance-aware loss function to train a SRS with multiple future items to make it more robust to noise. Our relevance-aware models obtain an improvement of 1.2% of NDCG@10 and 0.88% in the traditional evaluation protocol, while in the new evaluation protocol, the improvement is 1.63% of NDCG@10 and 1.5% of HR w.r.t the best performing models.

CCS Concepts: • **Information systems** → **Relevance assessment**; **Recommender systems**; • **Computing methodologies** → Neural networks.

Additional Key Words and Phrases: Recommender systems, Sequential recommendation, item relevance

1 INTRODUCTION

Recommender systems have become an integral part of our daily lives [36], as they assist us in making decisions by suggesting items we might like based on our preferences and behaviors [24]. In recent years, Sequential Recommender Systems (SRSs) have emerged as a promising solution to improve the accuracy and relevance of recommendations by incorporating the temporal aspect of user-item interactions. These systems aim to predict the next item a user is likely to interact with based on their past interactions, taking into account the sequence of actions and their temporal order [23].

SRSs have been successfully applied to various domains, including e-commerce [14], music streaming [26], and movie recommendations [7]. Traditional evaluation metrics, such as Hit Rate (HR) and Normalized Discounted Cumulative Gain (NDCG), fail to capture the complexity of sequential data when evaluated by measuring how well they predict a single future item [16, 19, 29]. The reason is that the hypothesis of a single "relevant" item might not reflect the users' true intentions or preferences, particularly when considering noisy sequences in real-world scenarios [32]. For example, users

Authors' addresses: Andrea Bacciu, Sapienza University of Rome, Rome, Italy, bacciu@diag.uniroma1.it; Federico Siciliano, Sapienza University of Rome, Rome, Italy, siciliano@diag.uniroma1.it; Nicola Tonellotto, University of Pisa, Pisa, Italy, ???; Fabrizio Silvestri, Sapienza University of Rome, Rome, Italy, fsilvestri@diag.uniroma1.it.

accidentally clicking on items, or performing multiple actions quickly, can greatly affect the evaluation results, as shown by Gupta and Gupta [8], Oh et al. [21]. The evaluation of SRSs has been thoroughly analyzed in recent research [15, 28, 37].

Our paper contributes to this line of research by proposing a novel approach: we depart from the single-relevant item approach typically taken in the literature and adopt a novel evaluation method and training approach that considers multiple future items and accounts for noise in the sequences. The proposed method provides, as shown in the experiments, a more accurate and robust solution for evaluating and training SRSs. We conducted experiments on four datasets typically used in this research domain, using SASRec [16], a widely cited SRS, as the reference model for our experiments.

The research questions we address in our experiments are the following:

- **RQ1:** Is there an alternative to the single relevant item evaluation protocol that is more suitable to determine the best-performing ranking model among several proposed options?
- **RQ2:** Can the item relevance be successfully incorporated into the training mechanism of an SRS to boost its performance?
- **RQ3:** What is the impact of the considered number of future items on the evaluation metrics as well as on the training performance of the models considered?

Our paper makes two key contributions to the field of SRSs. Firstly, we introduce a new evaluation protocol that considers multiple future items as potentially relevant instead of the typical single-relevant item hypothesis used in this research area. Secondly, our experiments show that when trained in the multiple-relevant item regime, SASRec outperforms the state-of-the-art models, as shown by the significant improvements in NDCG@10 and Recall@10 scores.

2 RELATED WORK

2.1 Sequential Recommender Systems

Sequential Recommender Systems (SRSs) are a class of recommender systems that personalizes recommendations to users based on their historical interactions with items in a sequence [31], capturing its temporal dynamics. SRSs have received considerable attention in the research community in recent years [36] and have been applied in various domains [36], including movies [7, 9, 22], music [26, 27], and e-commerce [14, 25]. Various techniques have been developed to model the temporal dependencies in the sequences, including Markov Chain models, Recurrent Neural Networks (RNNs), and Attention mechanisms. Markov Chain models are a type of probabilistic model that assume the future state of a sequence only depends on the current state; for this reason, they struggle to capture complex dependencies in long-term sequences [5, 6]. RNNs are a type of neural network architecture that can capture the long-term dependencies in sequential data. They have shown great potential in modeling sequential data, and they have been used to develop various SRSs, such as session-based recommenders [11–13, 18, 20], context-aware recommenders [1, 17, 35], and sequential graph neural networks [3, 4]. Attention mechanisms have recently gained attention in SRSs due to their ability to dynamically weight the importance of different parts of the sequence [16]. By doing so, attention mechanisms can better capture the important features in the sequence and improve the prediction accuracy [33, 38].

2.2 Evaluating Sequential Recommender Systems

Evaluating Sequential Recommender Systems (SRSs) has been a topic of great interest in recent years. Both [28] and [37] examined common data splitting methods for SRSs and discussed why commonly used evaluation methods are ill-defined, suggesting appropriate offline evaluation for SRSs. In particolar, they showed that existing evaluation protocols do not consider the temporal dynamics of user behaviour, which can affect the accuracy of the recommendations.

In [15] it is shown that the current evaluation protocols for SRSs can lead to data leakage, where the model learns information from the test data that is not available during training. They address the problem by proposing an evaluation methodology that takes into account the global timeline of data samples in the evaluation of SRSs. A metric called Rank List Sensivity (RLS) is introduced in [21] to evaluate the discrepancy between two rankings, so to evaluate models' sensitivity with respect to training data. Finally, [2] presented an evaluation methodology specifically designed to evaluate the precision of algorithms for the Search Shortcut Problem. This metric considers item relevance, which allows an effective evaluation.

3 METHODOLOGY

3.1 Current Evaluation Protocol

In line with previous research employing SRSs [16, 19, 29], the current evaluation method involves shuffling one positive item (the next item in the sequence) with 100 random negative items not part of the input sequence. These items are then ranked based on their relevance scores determined by the model. The resulting rank is used to calculate evaluation metrics like Normalized Discounted Cumulative Gain (NDCG) and Hit Rate (HR), which can be measured with various cut-offs, typically 10.

3.2 Problems with the current protocol

The current evaluation protocol assumes only one item is relevant to a user, which may not be true in real-world scenarios where multiple interactions could be relevant. Users' history can be noisy, as shown in [32]. Noise can come from account sharing, inconsistent preferences, or accidental clicks. For instance, in e-commerce, many clicks don't lead to purchases, and some purchases receive negative reviews. Evaluating a model based on one positive item, which might be an error, negatively impacts its performance, and this issue is not addressed in the current evaluation protocol.

3.3 Multi Future Items: a new evaluation protocol

To address the aforementioned problem, we propose a new evaluation protocol for SRSs called Multi Future Items (MFI). In MFI, we make the Assumption 1, that a good ranking should not only contain the single future next item in the sequence, but the whole sequence of future items in the correct order. Hence, MFI is rewarding the models that produce a better ranking considering multiple future items.

ASSUMPTION 1. *Given a user u and its ordered interactions' sequence $[I_1, I_2, ..., I_i]$, the ideal ranking of length K is the sequence of future items $[I_{i+1}, I_{i+2}, ..., I_{i+K}]$ arranged user's interaction temporal order.*

To evaluate the performance of SRSs under Assumption 1, we use traditional evaluation metrics for sequential recommendation models such as NDCG and HR. To have multiple future items per evaluation, we therefore split the user's history differently. Given a sequence $[I_1, I_2, ..., I_L]$, in the traditional evaluation protocol only item I_L is reserved for testing. In our proposed evaluation protocol, the sequence $[I_1, I_2, ..., I_{L-K}]$ is allocated for training the model, while the sequence $[I_{L-K+1}, I_{L-K+2}, ..., I_L]$ is used for testing purposes.

3.4 Item Relevance

The new evaluation protocol requires higher ranking capabilities than the traditional evaluation protocol. In the original evaluation protocol, the ability to rank a single item and treat all other items as not relevant to the user is evaluated.
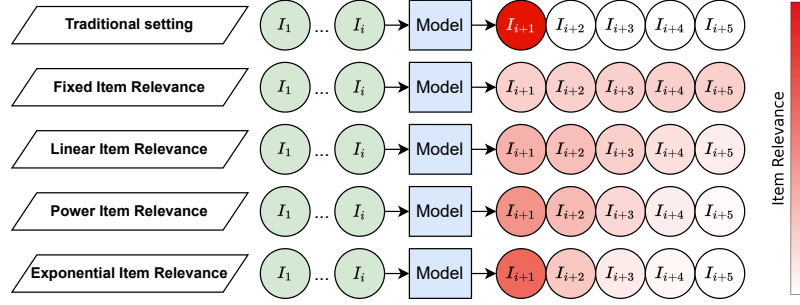
Fig. 1. A visualization of how the various loss scaling strategy weigh the item relevance.

Conversely, in the new evaluation protocol, it is important to define item relevance to give importance to multiple future items and scale their importance according to their position in the sequence.

DEFINITION 1. *Given a ranking $[I_1, I_2, ..., I_K]$ of length K, we define the item relevance function $r \colon \mathbb{N} \rightarrow [0, 1]$*

In order to compare the relevance of items on sequences of different lengths K, we further define that the item relevance sums to one.

DEFINITION 2. *Given a ranking $[I_1, I_2, ..., I_K]$ of length K, r must satisfy $\sum_{i=1}^{K} r(i) = 1$*

Item relevance should account for the fact that some interactions will occur further in the future and therefore assign higher importance to them: we establish that the relevance of an item at time t cannot be lower than the relevance of the item at time $t + 1$.

DEFINITION 3. *Given a ranking $[I_1, I_2, ..., I_K]$ of length K, r must satisfy $r(i + 1) \leq r(i) \quad \forall i \in \{1, 2, ..., K\}$*

Our approach to item relevance is inspired by [2] who proposed a similarity function, which takes into account the item relevance, to evaluate query recommendation using collaborative filtering. They suggested four different functions for item relevance: $r(i) = 1$, $r(i) = K - i$, $r(i) = (K - i)^2$, and $r(i) = e^{K-i}$ with $i \in \{1, 2, ..., K\}$. We refer to these functions respectively as *Fixed*, *Linear*, *Power*, and *Exponential*. The first assigns equal relevance to all items, while the others assign higher relevance to the next items in the sequence at the expense of the more distant ones. We show a visualization of these functions in Figure 1. The functions can be easily normalized to comply with the Definition 2. It should be noted that our evaluation protocol generalizes the traditional one because we can revert to it by setting a maximum importance to the next item and zero importance for all other future items.

3.5 Relevance-based Loss

Current Neural SRSs use Binary Cross-Entropy as the loss function; we propose a modification of it called Relevance-based loss. Relevance-based loss uses multiple positive items, similarly to what proposed by [30], except that it assigns weights to them following the functions defined in Section 3.4. [30], though, propose a formulation that assigns equal relevance to all future items (equivalent to our Fixed formulation). However, by assigning equal importance to all items, the model ignores the natural order of interactions, making this strategy unsuitable for some tasks. For instance, in the case of movie recommendations, it is not realistic to suggest *Back To The Future 3* before the user has watched the

first and the second. To address this limitation, we propose a more general formula that accounts for item relevance through time, which we call Relevance-based loss:

$$\ell(\vec{x} \mid pos, neg, r) = -\sum_{i=1}^{pos} \log\left(\left(\vec{x}_{pos}\right)_i\right) r(pos - i + 1) - \sum_{i=1}^{neg} \log\left(1 - \left(\vec{x}_{neg}\right)_i\right) \tag{1}$$

where *pos* and *neg* represent respectively the number of positive and negative items considered. $\vec{x}$ is the score given by the model to each item, while $\vec{x}_{pos}$ and $\vec{x}_{neg}$ represent respectively the subset of $\vec{x}$ containing only the positive and negative items and r is the item relevance function defined in Section 3.4. In Equation 1, we weigh the loss of each item i by its relevance score $r(i)$. By doing so, the model is encouraged to focus more on the relevant items while learning.

4 EXPERIMENTAL SETUP

We assess our techniques using four datasets derived from real-world use cases: MovieLens[10] and Foursquare [34]. MovieLens is a popular benchmark dataset for Recommendation Systems containing movie ratings. We utilize both MovieLens-1M and MovieLens-100K, which comprises 1 million and 100.000 user ratings, respectively. The Foursquare NYC and Foursquare TKY datasets, presented in [34], are collections of check-in records from New York and Tokyo. The NYC dataset contains 227,428 records, while the TKY dataset has 573,703 records.

We select the Self-Attentive Sequential Recommendation (SASRec) model[1] [16] for our experiment, as it has consistently demonstrated exceptional performance across multiple benchmarks and garnered significant recognition in the literature. To have a fair comparison, we keep the original hyper-parameter of SASRec's paper. We perform our experiments on a workstation equipped with an Intel Core i9-10940X (14-core CPU running at 3.3GHz) and 256GB of RAM, and a single Nvidia RTX A6000 with 48GB of VRAM.

5 RESULTS

In this Section, we present the results to answer our posed research questions (Section 1). During our experiments, we vary (i) The number of evaluation positives; (ii) The number of training positives; (iii) the item relevance function. From here on, we will denote by 'our' o 'new' models, the models trained with our proposed training loss.

***RQ1:** Is there an alternative to the single relevant item evaluation protocol that is more suitable to determine the best-performing ranking model among several proposed options?*
Table 1 compares various models in the traditional evaluation protocol, including two baselines (Baseline and Fixed) and the models trained with our proposed item relevance-based loss (Linear, Power, Exponential). As expected, in the traditional evaluation protocol, there is no clear winner. Instead, in Table 2, the performance discrepancies are more pronounced and it is, therefore, possible to identify a better model. Hence, the new evaluation protocol is more robust to the noise introduced in 3.2 and can better identify performance differences between models and reward the model with the best ranking performance as evaluated by means of NDCG and HR. We can also assert that models that discount the future excessively, i.e. only (or almost only) take into account the next item (Baseline and Exponential), generally perform worse.

***RQ2:** Can the item relevance be successfully incorporated into the training mechanism of an SRS to boost its performance?*
As can be seen from Table 1, in the traditional evaluation protocol, models that integrate the item relevance are on-par or better than the baselines for at least one of the metrics. In particular, Linear item relevance outperforms the other approaches in 6 out of 10 cases while getting the second-best result in 2 cases. When considering more positive items

---

[1]https://github.com/pmixer/SASRec.pytorch

	ML-1M		ML-100k		Foursquare NYC		Foursquare TKY	
Model	NDCG	HR	NDCG	HR	NDCG	HR	NDCG	HR
Baseline [16]	0.5855	**0.8228**	0.4300	0.7041	0.6649	0.7525	0.7217	**0.8116**
Fixed [30]	0.5483	0.8017	0.4246	0.7084	0.6698	0.7581	0.7247	0.8094
Linear	**0.5896**	0.8187	**0.4358**	**0.7137**	0.6715	0.7581	**0.7295**	0.8081
Power	0.5751	0.8220	0.4309	0.6999	**0.6765**	**0.7719**	0.7229	0.8090
Exponential	0.5169	0.7763	0.4233	0.6999	0.5717	0.7442	0.6195	0.7872

Table 1. The table shows the values of the NDCG@10 (higher is better) and HR@10 (higher is better) metrics that the five models obtain on the four datasets in the traditional evaluation protocol. **Bold** shows the best result for each column, underlined the second best.

	ML-1M		ML-100k		Foursquare NYC		Foursquare TKY	
Model	NDCG	HR	NDCG	HR	NDCG	HR	NDCG	HR
Baseline [16]	0.6532	0.6151	0.5638	0.5269	0.7617	**0.6864**	0.8080	0.7433
Fixed [30]	0.6562	0.6188	0.5632	0.5266	0.7611	0.6848	0.8139	0.7508
Linear	**0.6694**	**0.6307**	**0.5753**	**0.5368**	0.7549	0.6765	**0.8234**	**0.7570**
Power	0.6607	0.6230	0.5694	0.5298	**0.7625**	0.6848	0.8193	0.7549
Exponential	0.6536	0.6147	0.5667	0.5297	0.7624	0.6845	0.8185	0.7544

Table 2. The table shows the values of the NDCG@10 (higher is better) and HR@10 (higher is better) metrics that the five models obtain on the four datasets in the new evaluation protocol using 10 positive items. **Bold** shows the best result for each column, underlined the second best.

for evaluation (see Table 2) it is possible to notice that the Linear model outperforms the baselines and the other models, except in a benchmark where Power item relevance yields a better NDCG score and the baselines a better HR. Fixed, on the other hand, returns performance comparable to Baseline, suggesting that incorporating more positive items during training, even if without weighting them in the loss, can lead to a slight increase in performance.

Figure 2 shows the performance using NDCG@10 for the MovieLens-1M dataset for both the traditional and new evaluation protocol. From Figure 2, the improvement provided by item-relevance loss is more evident. We can in fact see that the Baseline (the original SASRec), which does not consider multiple future items during training, has a consistently slower convergence rate in both evaluation protocols than the item relevance-aware models. Instead, Linear item relevance obtains the fastest convergence. This shows that the item relevance loss allows the training of better models that yield higher performances in both evaluation protocols.

To summarize, our relevance-aware models obtain an improvement of 1.2% of NDCG@10 and 0.88% in the traditional evaluation protocol, while in the new evaluation protocol, the improvement is 1.63% of NDCG@10 and 1.5% of HR.

RQ3: *What is the impact of the considered number of future items on the evaluation metrics as well as on the training performance of the models considered?*

To assess this question, we conduct two ablation studies, one in which we vary the number of evaluation positive items and another in which we vary the number of training positive items. In Figure 3, we report the results of all the models

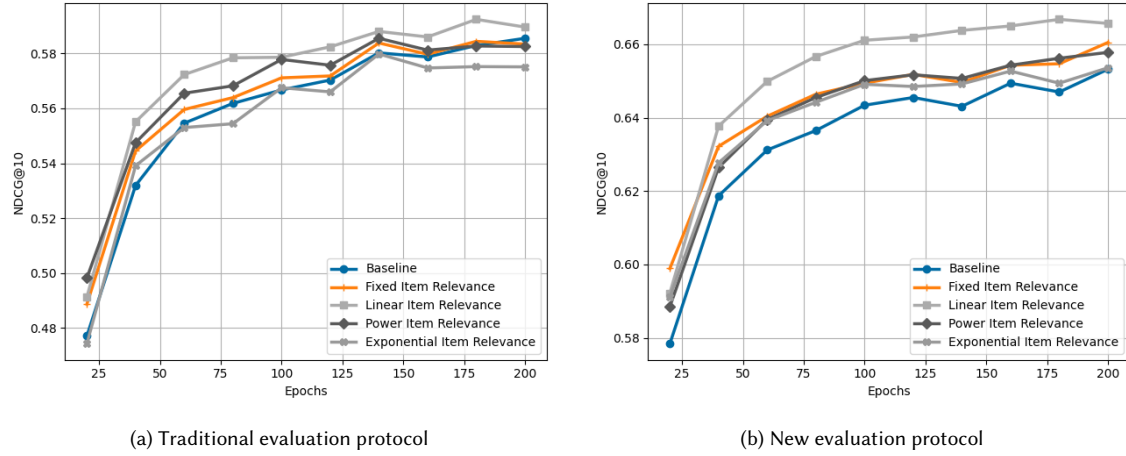(a) Traditional evaluation protocol   (b) New evaluation protocol

Fig. 2. NDCG@10 at varying training epochs for all 5 models on the MovieLens 1M dataset

using 1, 5, and 10 evaluation positive items. We can notice that the relative performance of the relevance-aware models remains consistent when varying numbers of evaluation positive items, despite variations in their absolute performance.
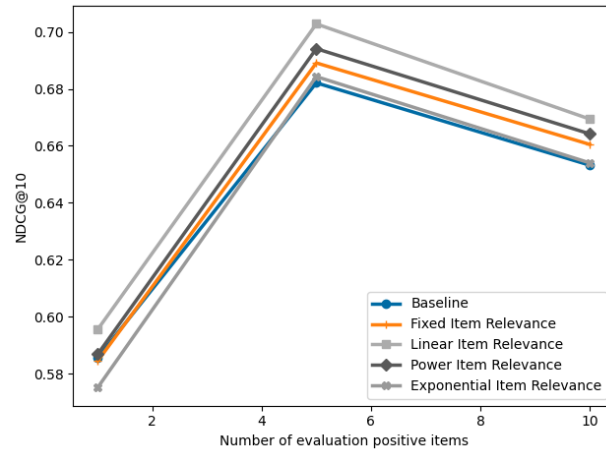

Fig. 3. NDCG@10 at varying number of positive items used for evaluation on Movielens-1M

In Figure 4 we report the results of all models using 2, 3, 4, 5, 10 training positive items. In the traditional evaluation protocol (Figures 4a, 4c), increasing the number of training positive items seems to impact the performances negatively: increasing the number of training positive items leads to a deterioration of performance. This is particularly evident for Fixed Item Relevance, where an increase in training positive items always leads to a decrease in performance. For Linear, this behaviour is more attenuated, although using 10 training positive items still generates the worst performance. In the traditional evaluation protocol, the behaviour is generally expected, as we are only evaluating the model with one positive item; the other positive items used for training can only confound the model. Instead, in the new evaluation

(a) Traditional evaluation protocol - Fixed Item Relevance

(b) New evaluation protocol - Fixed Item Relevance

(c) Traditional evaluation protocol - Linear Item Relevance
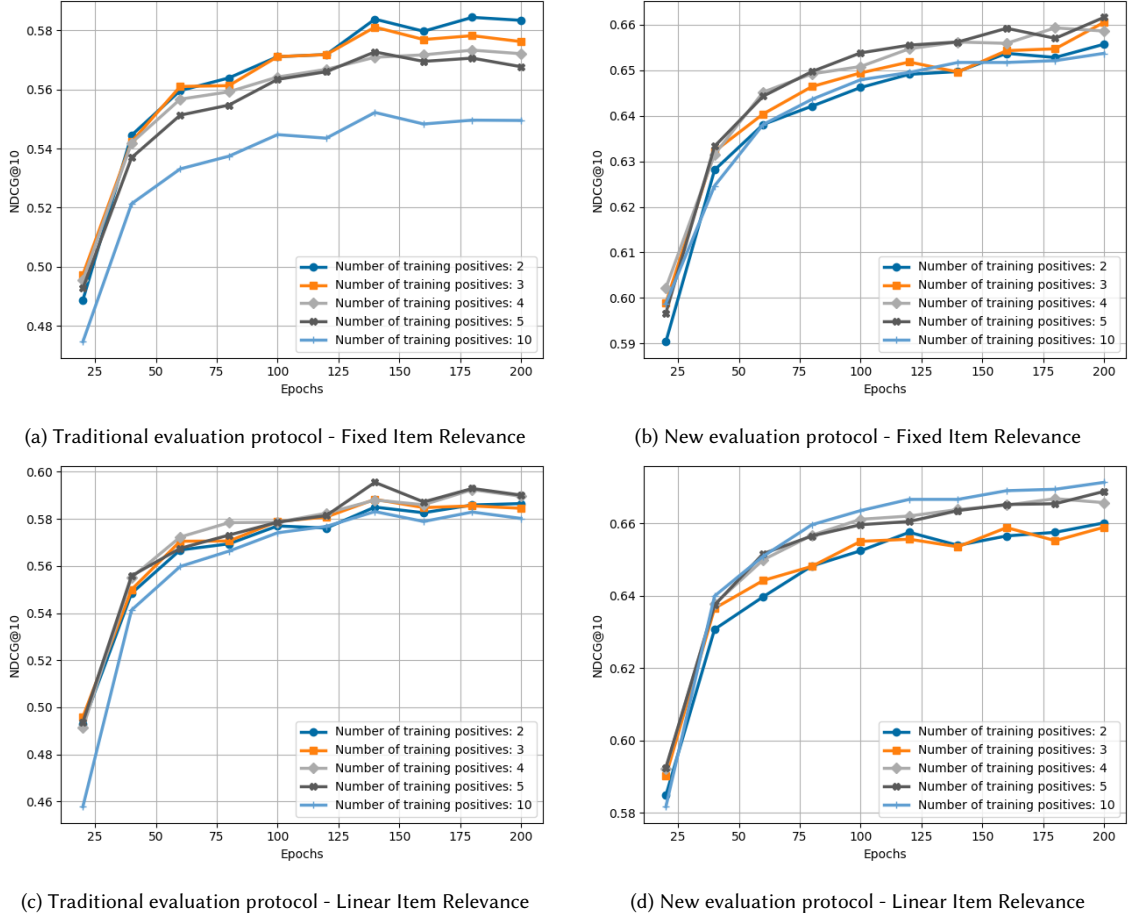
(d) New evaluation protocol - Linear Item Relevance

Fig. 4. NDCG@10 at varying training epochs for the new item-relevance models for different numbers of training positives on Movielens-1M

protocol (Figures 4b, 4d), where we are evaluating models with 10 positive items, we see something different. Fixed Item Relevance increases performance up to 5 training positives, but at 10 it performs worst. This indicates that this type of relevance function is not particularly suitable, even when the training is similar to the evaluation. Conversely, Linear shows interesting results: performance increases with the number of training positives. This suggests that our models still have room for further improvement. Although not shown, the results for Power Item Relevance are similar to those for Linear, while Exponential shows a similar trend to Fixed in general.

6 CONCLUSIONS

In this work, we challenged the typical assumption made in Sequential Recommendation Systems (SRSs), which only consider the immediate next item in a sequence as the one to predict. We have relaxed the evaluation protocol to not penalize the model in case of noisy sequences and designed an item relevance loss in order to optimize the model to predict multiple future items. We demonstrated the importance of having more positive examples both in the training

and evaluation of SRSs. Our experiments show that when trained in the multiple relevant item regime, our systems outperform the state-of-the-art models, with an improvement of 1.2% of NDCG@10 and 0.88% in the original evaluation protocol. In the new evaluation protocol, the improvement is 1.63% of NDCG@10 and 1.5% of HR. Among all the variants of grading multiple relevant items that we experimented with, it turns out that the Linear one outperforms the others and demonstrates its potential usefulness in practical applications.

ACKNOWLEDGMENTS

REFERENCES

[1] Gediminas Adomavicius and Alexander Tuzhilin. 2010. Context-aware recommender systems. , 217–253 pages.

[2] Ranieri Baraglia, Fidel Cacheda, Victor Carneiro, Diego Fernandez, Vreixo Formoso, Raffaele Perego, and Fabrizio Silvestri. 2009. Search shortcuts: a new approach to the recommendation of queries. , 77–84 pages.

[3] Jianxin Chang, Chen Gao, Yu Zheng, Yiqun Hui, Yanan Niu, Yang Song, Depeng Jin, and Yong Li. 2021. Sequential recommendation with graph neural networks. , 378–387 pages.

[4] Ziwei Fan, Zhiwei Liu, Jiawei Zhang, Yun Xiong, Lei Zheng, and Philip S Yu. 2021. Continuous-time sequential recommendation with temporal graph collaborative transformer. , 433–442 pages.

[5] Francois Fouss, Stephane Faulkner, Manuel Kolp, Alain Pirotte, Marco Saerens, et al. 2005. Web Recommendation System Based on a Markov-Chainmodel. , 56–63 pages.

[6] Francois Fouss, Alain Pirotte, and Marco Saerens. 2005. A novel way of computing similarities between nodes of a graph, with application to collaborative recommendation. , 550–556 pages.

[7] Mahesh Goyani and Neha Chaurasiya. 2020. A review of movie recommendation system: Limitations, Survey and Challenges. , 0018–37 pages.

[8] Shivani Gupta and Atul Gupta. 2019. Dealing with noise problem in machine learning data-sets: A systematic review. , 466–474 pages.

[9] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. , 19 pages.

[10] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. , 1–19 pages. https://doi.org/10.1145/2827872

[11] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent neural networks with top-k gains for session-based recommendations. , 843–852 pages.

[12] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks.

[13] Balázs Hidasi, Massimo Quadrana, Alexandros Karatzoglou, and Domonkos Tikk. 2016. Parallel recurrent neural network architectures for feature-rich session-based recommendations. , 241–248 pages.

[14] Hyunwoo Hwangbo, Yang Sok Kim, and Kyung Jin Cha. 2018. Recommendation system development for fashion retail e-commerce. , 94–101 pages.

[15] Yitong Ji, Aixin Sun, Jie Zhang, and Chenliang Li. 2023. A critical study on data leakage in recommender system offline evaluation. , 27 pages.

[16] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. , 197–206 pages.

[17] Saurabh Kulkarni and Sunil F Rodd. 2020. Context Aware Recommendation Systems: A review of the state of the art techniques. , 100255 pages.

[18] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. , 1419–1428 pages.

[19] Jiacheng Li, Yujie Wang, and Julian McAuley. 2020. Time interval aware self-attention for sequential recommendation. , 322–330 pages.

[20] Duo Liu, Yang Sun, Xiaoyan Zhao, Gengxiang Zhang, and Rui Liu. 2020. Adversarial training for session-based item recommendations. , 1162–1168 pages.

[21] Sejoon Oh, Berk Ustun, Julian McAuley, and Srijan Kumar. 2022. Rank List Sensitivity of Recommender Systems to Interaction Perturbations. , 1584–1594 pages.

[22] Aleksandr Petrov and Craig Macdonald. 2022. Effective and Efficient Training for Sequential Recommendation using Recency Sampling. In *Proceedings of the 16th ACM Conference on Recommender Systems*. 81–91.

[23] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. 2018. Sequence-aware recommender systems. , 36 pages.

[24] Francesco Ricci, Lior Rokach, and Bracha Shapira. 2011. Introduction to recommender systems handbook. , 35 pages.

[25] J Ben Schafer, Joseph A Konstan, and John Riedl. 2001. E-commerce recommendation applications. , 115–153 pages.

[26] Markus Schedl, Peter Knees, Brian McFee, Dmitry Bogdanov, and Marius Kaminskas. 2015. Music recommender systems. , 453–492 pages.

[27] Markus Schedl, Hamed Zamani, Ching-Wei Chen, Yashar Deldjoo, and Mehdi Elahi. 2018. Current challenges and visions in music recommender systems research. , 95–116 pages.

[28] Aixin Sun. 2022. From counter-intuitive observations to a fresh look at recommender system.

[29] 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. , 1441–1450 pages.

[30] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. , 565–573 pages.

[31] Shoujin Wang, Liang Hu, Yan Wang, Longbing Cao, Quan Z Sheng, and Mehmet Orgun. 2019. Sequential recommender systems: challenges, progress and prospects.

[32] Wenjie Wang, Fuli Feng, Xiangnan He, Liqiang Nie, and Tat-Seng Chua. 2021. Denoising implicit feedback for recommendation. , 373–381 pages.

[33] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. , 165–174 pages.

[34] Dingqi Yang, Daqing Zhang, Vincent W Zheng, and Zhiyong Yu. 2014. Modeling user activity preference by leveraging user spatial temporal characteristics in LBSNs. , 129–142 pages.

[35] Libin Yang, Yu Zheng, Xiaoyan Cai, Hang Dai, Dejun Mu, Lantian Guo, and Tao Dai. 2018. A LSTM based model for personalized context-aware citation recommendation. , 59618–59627 pages.

[36] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. , 38 pages.

[37] Wayne Xin Zhao, Zihan Lin, Zhichao Feng, Pengfei Wang, and Ji-Rong Wen. 2022. A revisiting study of appropriate offline evaluation for top-N recommendation algorithms. , 41 pages.

[38] Chang Zhou, Jinze Bai, Junshuai Song, Xiaofei Liu, Zhengchao Zhao, Xiusi Chen, and Jun Gao. 2018. Atrank: An attention-based user behavior modeling framework for recommendation.