The Role of Fake Users in Sequential Recommender Systems*

Filippo Betello[1]

[1]Sapienza University of Rome, Rome, Italy

Abstract

Sequential Recommender Systems (SRSs) are widely used to model user behavior over time, yet their robustness remains an under-explored area of research. In this paper, we conduct an empirical study to assess how the presence of fake users —who engage in random interactions, follow popular or unpopular items, or focus on a single genre —impacts the performance of SRSs in real-world scenarios. We evaluate two SRS models across multiple datasets, using established metrics such as Normalized Discounted Cumulative Gain (NDCG) and Rank Sensitivity List (RLS) to measure performance. While traditional metrics like NDCG remain relatively stable, our findings reveal that the presence of fake users severely degrades RLS metrics, often reducing them to near-zero values. These results highlight the need for further investigation into the effects of fake users on training data and emphasize the importance of developing more resilient SRSs that can withstand different types of adversarial attacks.

Keywords

Recommender Systems, Evaluation of Recommender Systems, Model Stability, Input Data Perturbation

1. Introduction

Recommender Systems (RSs) have become an essential part of our daily lives, helping users navigate the vast online information landscape [1]. With the global expansion of e-commerce services, social media platforms and streaming services, these systems have become essential for personalising content delivery and increasing user engagement [2].

Over the last several years, Sequential Recommender Systems (SRSs) have gained significant popularity as an effective method for modeling user behavior over time [3]. By capitalizing on the temporal dependencies within users' interaction sequences, these systems can make more precise predictions about user preferences [4]. This approach allows for a more nuanced understanding of user behavior, leading to recommendations that are better tailored to individual needs and preferences. As a result, SRSs have become a critical component in various applications, ranging from e-commerce [5] to music recommendation [6], where understanding and anticipating user preferences is key to enhancing user experience and engagement.

In recent years, the prevalence of bots (fake users) on social media platforms has increased dramatically [7]. It is estimated that Amazon, for example, spends 2% of its net revenue each year fighting counterfeiting [8]. While several techniques have been identified to counteract this growing problem [9, 10], a detailed investigation in the area of sequential recommendation systems is still lacking. Li et al. [11] aims to fill this gap by investigating the impact of bot-generated data on sequential recommendation models. Specifically, it seeks to determine an optimal bot-generation budget and analyze its impact on popular matrix factorization models. Indeed, controlling and maintaining a large number of bots is costly.

Therefore, it is possible to create a limited number of bots that can significantly influence the prominence of a particular item or category. By strategically deploying these bots, the visibility and perceived importance of the targeted item or category can be enhanced, making it stand out more compared to others. Imagine if, by using fake users, it were possible to raise the profile of a certain category or product or, conversely, to lower the profile of another. This scenario represents a form of unfair competition and is therefore crucial to study. Understanding how fake users behave in controlled environments allows us to assess their impact on real users. It is also important to investigate whether partially coordinated fake users can actively improve the performance or predictions of a particular category or item.

In this paper, we investigate the impact of fake users on sequential recommendation systems. Specifically, we investigate how the inclusion of a certain percentage of bots affects the performance of real users. These bots are programmed to deal with random items, popular items, unpopular items and items within the same category.

Our experiments focus on the following research questions:

- **RQ1**: How does the value of standard metrics such as NDCG change for real users depending on the type and increasing number of fake users?
- **RQ2**:How do recommendation lists for real users differ from those generated without fake users?
- **RQ3**: Are more or less popular items favoured by the presence of fake users with certain types of interactions?

We evaluate our hypothesis using two different models, SASRec [12] and GRU4Rec [13], and by employing four different datasets, namely MovieLens 1M, MovieLens 100k [14], Foursquare New York City and Foursquare Tokyo [15].

2. Related Work

2.1. Sequential Recommender Systems

Sequential recommendation systems (SRSs) use algorithms that analyze a user's past interactions with items to provide personalized recommendations over time. These systems have found widespread application in areas such as

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

*This work was partially supported by projects FAIR (PE0000013) and SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU. Supported also by the ERC Advanced Grant 788893 AMDROMA, EC H2020RIA project "SoBigData++" (871042), PNRR MUR project IR0000013-SoBigData.it. This work has been supported by the project NEREO (Neural Reasoning over Open Data) project funded by the Italian Ministry of Education and Research (PRIN) Grant no. 2022AEFHAZ.

betello@diag.uniroma1.it (F. Betello)

0009-0006-0945-9688 (F. Betello)

e-commerce [16, 5], social media [17, 18], and music streaming services [19, 20, 6]. Unlike traditional recommender systems, SRSs take into account the sequence and timing of user interactions, resulting in more precise predictions of user preferences and behaviors [4].

Various methods have been developed to implement SRSs. Early approaches used Markov Chain models [21, 22], which, despite their simplicity, struggled with capturing complex dependencies in long-term sequences. More recently, Recurrent Neural Networks (RNNs) have become prominent in this domain [23, 13, 24]. RNNs encode a user's historical preferences into a vector that is updated at each time step to predict the next item in the sequence. However, RNNs can encounter difficulties with long-term dependencies and generating diverse recommendations.

The attention mechanism [25] has introduced another promising approach. Models like SASRec [12] and BERT4Rec [26] leverage this mechanism to dynamically weight different parts of the sequence, capturing key features to enhance prediction accuracy.

Additionally, Graph Neural Networks have recently gained traction in the recommendation system field, particularly within the sequential domain [27, 28]. These networks excel at modeling complex relationships and dependencies, further advancing the capabilities of SRSs [29, 30, 31].

2.2. Training Perturbations

Robustness is an important aspect of SRSs as they are vulnerable to noisy and incomplete data. [32, 33] investigated the effects of removing items at the beginning, middle and end of a sequence of temporally ordered items and found that removing items at the end of the sequence significantly affected all performances.

Yin et al. [34] design an attacker-chosen targeted item in federated recommender systems without requiring knowledge about user-item rating data, user attributes, or the aggregation rule used by the server. While studies are being conducted in other areas of recommendation [35, 36] and several techniques have been identified to counteract this growing problem [9, 10], a detailed investigation in the area of sequential recommendation systems is still lacking.

Li et al. [11] aim to address this issue by examining how bot-generated data affects sequential recommendation models. Their research focuses on finding the optimal budget for bot generation and assessing its influence on widely used matrix factorization models. Indeed, controlling and maintaining a large number of bots is costly. Previous research has proposed attacks using a limited number of users and clustering models [37], but these have not been extensively studied in the context of sequential recommendations.

To the best of our knowledge, our research is completely novel and breaks new ground. It explores the role that fake users might play in influencing real users. This study aims to shed light on the potential impact that fake users could have on the behaviour, opinions and interactions of real users within sequential recommendation systems.

3. Methodology

3.1. Background

The main objective of sequential recommendation systems is to predict the user's next interaction in a given sequence. Suppose we have a set of n users, represented as $\mathcal{U} \subset \mathbb{N}^+$, and a corresponding set of n items, represented as $\mathcal{I} \subset \mathbb{N}^+$. Each user $u \in \mathcal{U}$ is associated with a time-ordered sequence of interactions $S_u = [s_1, \ldots, s_{L_u}]$, where each $s_i \in \mathcal{I}$ denotes the i-th item with which the user has interacted. The length of this sequence, L_u, is greater than 1 and varies from user to user.

A sequential recommendation system (SRS), denoted $\mathcal{M}$, processes the sequence up to the L-th item, denoted $S_u^L = [s_1, \ldots, s_L]$, to suggest the next item, s_{L+1}. The recommendation output, $r^{L+1} = \mathcal{M}(S_u^L) \in \mathbb{R}^m$, is a score distribution over all possible items. This distribution is used to create a ranked list of items, predicting the most likely interactions for user u in the next step, $L + 1$.

3.2. Fake user design

Given that each item in the set $\mathcal{I}$ has a popularity value determined by user interactions, we designed four types of fake user scenarios:

- **Random**: Items are randomly sampled from the entire set $\mathcal{I}$. Formally, each item s_i in the sequence S_u is selected with probability $\frac{1}{|\mathcal{I}|}$.
- **Popularity**: Items are sampled according to a popularity-based probability distribution P_{pop}, where the probability of selecting item s_i is proportional to its popularity p_i.
- **Unpopularity**: Similar to the popularity-based scenario, but with a distribution P_{unpop} that inversely favors popular items. Here, the probability of selecting item s_i is inversely proportional to its popularity, $\Pr(s_i) \propto \frac{1}{p_i}$, favoring less popular items.
- **Genre**: In this scenario, items are sampled exclusively from a specific genre. It is only applied to the ML datasets.

These fake users sequences will contain unique items to ensure there are no repetitions. While the first scenario involves users acting independently without any sense of cooperation, the middle two scenarios introduce a level of implicit cooperation. Specifically, users in these scenarios tend to converge on viewing either highly popular or highly unpopular items, reflecting a collective behavior. The average length of the sequences will be the same as that of real users. The proportion of synthetic users will vary, comprising 1%, 5%, 10%, 15% and 20% of the original dataset. The fake users are only used in the training data, leaving the test data unaffected.

3.3. Models

In our study, we use two different architectures to validate our results:

- **SASRec** [12], which uses self-attention mechanisms to evaluate the importance of each interaction between the user and the item.
- **GRU4Rec** [13], a RNN model that uses gated recurrent units (GRUs) [38] to improve prediction accuracy.

We chose these two models because they have demonstrated exceptional performance in numerous benchmarks and are widely cited in the academic literature. Moreover, since one model is based on attention mechanisms and the other on RNNs, their different network operations make it particularly interesting to evaluate their behaviour.

Table 1
Dataset statistics after pre-processing; users and items not having at least 5 interactions are removed. Avg. and Med. refer to the Average and Median of $\frac{\text{Actions}}{\text{User}}$, respectively.

Name	Users	Items	Interactions	Density	Avg.	Med.
FS-NYC	1,083	9,989	179,468	1.659	165	116
FS-TKY	2,293	15,177	494,807	1.421	215	146
ML-100k	943	1,349	99,287	7.805	105	64
ML-1M	6,040	3,416	999,611	4.845	165	96

3.4. Datasets

We use four different datasets:

MovieLens [14]: Frequently utilized to evaluate recommender systems, this benchmark dataset is employed in our study using both the 100K and 1M versions.

Foursquare [15]: This dataset includes check-in data from New York City and Tokyo, collected over a span of roughly ten months.

The statistics for all the datasets are shown in Table 1. Our pre-processing technique adheres to recognised principles, such as treating ratings as implicit, using all interactions without regard to the rating value, and deleting users and things with fewer than 5 interactions [12, 26]. For testing, as in [26, 12], we keep the most recent interaction for each user, while for validation, we keep the second to last action. The remaining interactions are added to the training set, which is the only one affected by the fake users perturbation.

We focus exclusively on genres in the ML dataset, as it is the only dataset that contains category information. Specifically, we select only those categories that represent more than 5% of the total items in the dataset.

3.5. Evaluation

We only carry out the evaluation on the part of the real users. To evaluate the performance of the models, we employ traditional evaluation metrics used for Sequential Recommendation: Precision, Recall, MAP and NDCG. Moreover, to investigate the stability of the recommendation models, we employ the Rank List Sensitivity (RLS) [33]: it compares two lists of rankings $\mathcal{X}$ and $\mathcal{Y}$, one derived from the model trained under standard conditions and the other derived from the model trained with perturbed data.

Given these two rankings, and a similarity function sim between them, we can formalise the RLS measure as

$$\mathbf{RLS} = \frac{1}{|\mathcal{X}|} \sum_{k=1}^{|\mathcal{X}|} \text{sim}(R^{X_k}, R^{Y_k}) \tag{1}$$

where X_k and Y_k represent the k-th ranking inside $\mathcal{X}$ and $\mathcal{Y}$ respectively.

RLS's similarity measure can be chosen from two possible options:

- **Jaccard Similarity (JAC)** [39] is a normalized measure of the similarity of the contents of two sets. A model is stable if its Jaccard score is close to 1.

$$\mathbf{JAC(X, Y)} = \frac{|X \cap Y|}{|X \cup Y|} \tag{2}$$

- **Finite-Rank-Biased Overlap (FRBO)** [32] measures the similarity of orderings between two rank lists. Higher values indicate that the items in the two lists are arranged similarly:

$$\mathbf{FRBO(X, Y)@k} = \frac{1-p}{1-p^k} \sum_{d=1}^{k} p^{d-1} \frac{|X[1:d] \cap Y[1:d]|}{d}$$

All metrics are computed "@k", meaning that we use just the first k recommended items in the output ranking, with $k \in \{10, 20\}$.

3.6. Experimental Setup

All experiments were performed on a single NVIDIA RTX A6000 with 10752 CUDA cores and 48 GB of RAM. We train the models for 500 epochs, fixing the batch size to 128 and by using the Adam optimizer [40] with a lr of 10^{-3}. To run our experiments, we use the EasyRec library [41].

4. Results

Our experiments aim to address the following research questions:

- **RQ1**: How does the value of standard metrics such as NDCG change for real users depending on the type and increasing number of fake users?
- **RQ2**:How do recommendation lists for real users differ from those generated without fake users?
- **RQ3**: Are more or less popular items favoured by the presence of fake users with certain types of interactions?

4.1. RQ1: Impact of Fake Users on Standard Metrics for Real Users

In Figure 1 the results for all datasets considered are shown for both models using the standard metrics.

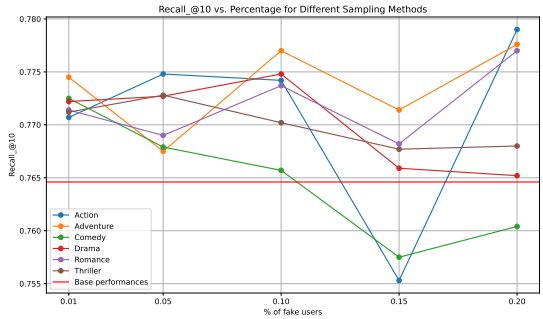
Regarding the SASRec shown in Figure 1d for the FS-NYC dataset, we observe that the performance tends to improve slightly for the unpopular scenario for the NDCG@20 metric, while for the popular and random interaction there is a gradual but consistent decline in performance. Regarding genre interactions in the ML-1M dataset, shown in Figure 1a, all genres appear to positively impact the NDCG metric. A more detailed analysis using RLS metrics is presented in Section 4.2.

In the case of GRU4Rec figs. 1b and 1c, there is a slow but steady decline in performance for the ML-100k and FS-TKY datasets, with the decline occurring in a predictable manner for both metrics considered, as the percentage of fake users increase.

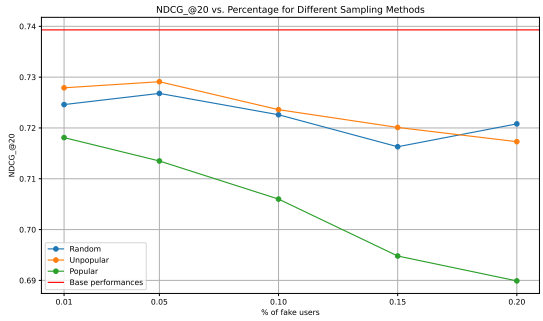
4.2. RQ2: Analysis of Recommendation Lists Generated for Real Users

In Figure 2 we present the RLS metrics for all datasets considered, comparing the performance of the two models. These metrics are derived from predictions made by the standard model - without fake users - and predictions made after training with fake users.

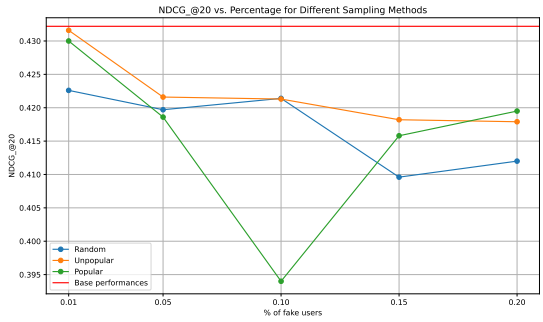
When analysing the SASRec model on the ML-100k dataset (fig. 2a), SASRec shows minimal performance degradation. Conversely, the FS-TKY dataset gives less favourable results, with significantly worse performance and a Jaccard



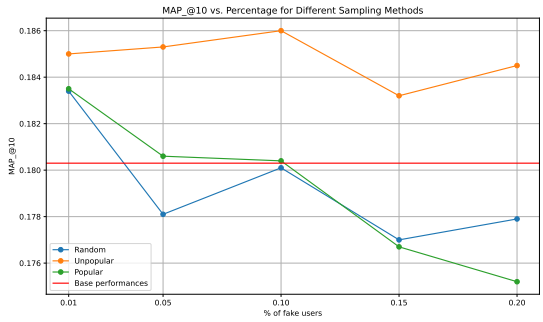
(a) NDCG@20 ML-1M SASRec



(b) MAP@10 FS-TKY GRU4Rec



(c) NDCG@20 ML-100k GRU4Rec



(d) NDCG@20 FS-NYC SASRec

Figure 1: Plots of various metrics for all the datasets considered as the percentage of fake users increases. The baseline is shown as a horizontal solid line, while other lines show the metrics as the percentage of fake users changes for the three scenarios considered.

index close to 0, indicating that the generated lists have almost no overlap with the original lists (fig. 2b).

Figures Figures 2c and 2d show the performance on the ML-100k dataset for genre sampling and the ML-1M dataset for the other sampling methods. On the ML-1M dataset, the performance is relatively good, although the Jaccard index remains low at around 0.35 (fig. 2c). For ML-100k and genre interactions, the degradation in performance is consistent across all genres, with the degradation worsening as the number of fake users increases.

The evaluation metrics for Foursquare show a significant drop in performance compared to other datasets, highlighting the limitations of the dataset [42].

An additional observation is that as the number of fake users increases, the performance of the model generally deteriorates. This suggests that while adding more fake users tends to reduce the effectiveness of the lists generated, managing a higher number of fake users becomes increasingly difficult.

4.3. RQ3: Influence of Fake User Interactions on Popular and Unpopular Items

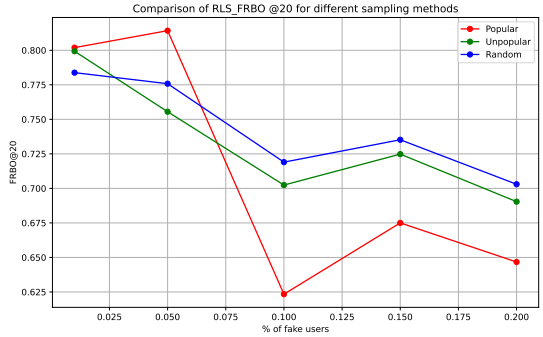
We investigated whether popular and unpopular items were favoured in recommendation lists by analysing the percentage of the top 20 items recommended to each user. Our results show that unpopular items were consistently underrepresented in these lists. This suggests that more users, a wider range of items, or consideration of a larger number of top positions (e.g. top 100 items) may be necessary to gain a better understanding. On the other hand, in the ML-100k dataset, the percentage of popular items in the recommendation lists without any user-specific adjustments is 5.73%. The introduction of popular users barely affects this percentage (5.68%), while the inclusion of non-popular users slightly reduces it to 5.45%.

These results suggest significant opportunities for future research, such as focusing on specific categories of items to either improve or reduce recommendation performance.

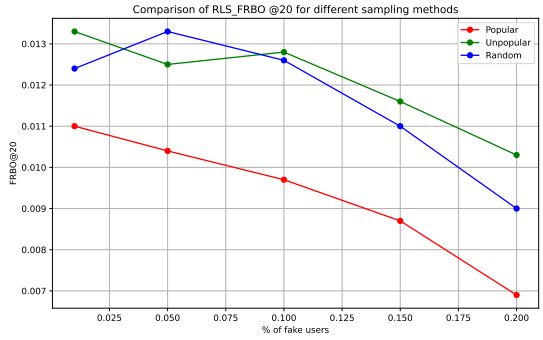
5. Conclusion

In this work we investigated the impact of fake users on real users. These fake users can have random interactions, interact with popular or unpopular items, and are only added to the training set at different percentages of the total dataset. The results showed that although the standard metrics were not significantly affected, with random perturbations causing the most significant degradation in performance, the output lists generated under these perturbations were significantly different from the standard lists trained without any perturbations. These differences, measured using ranking list sensitivity metrics, in particular Jaccard and FRBO, showed that in the case of MovieLens about half of the list elements were shared, whereas in the case of Foursquare almost no elements were considered. Furthermore, the proportion of popular and unpopular items in recommendations for real users was not affected by the presence of fake users.

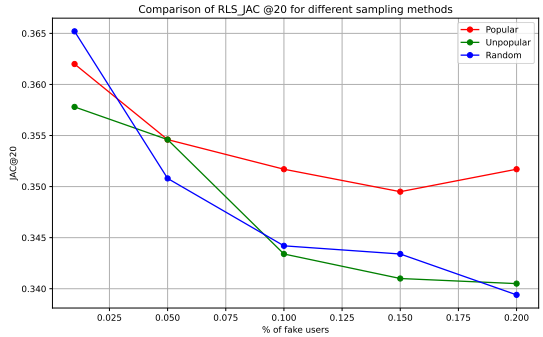
This study opens up future research directions in a number of ways. First, it would be valuable to compare the number of recommended items - categorised as popular, unpopular and genre-specific - using a standard training model with those generated by a model trained on fake users. This comparison could reveal better significant differences in recommendation patterns. Second, the creation of a set of fake users could allow to systematically elevate or downgrade certain categories over time. Third, studying datasets with shorter interaction sequences, such as those from Amazon [43], could provide new insights into user



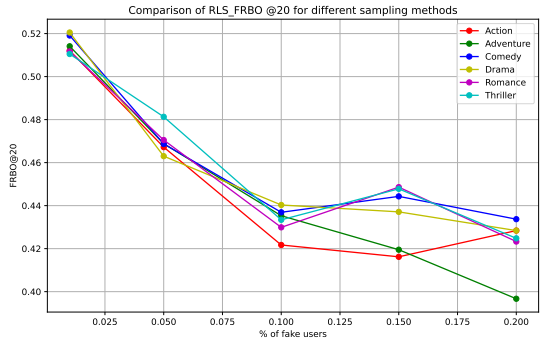
(a) RLS-FRBO ML100k SASRec



(b) RLS-FRBO FS-TKY SASRec



(c) RLS-JAC ML-1M GRU4Rec



(d) RLS-FRBO ML-100k GRU4Rec

Figure 2: Plots of RLS metrics for all the datasets considered as the percentage of fake users increases. The metrics are shown as the percentage of fake users changes for the three scenarios considered.

behaviour and recommendation effectiveness. Finally, research should focus on building resilient models for these types of perturbations: the solution could lie in different training strategies[44], robust loss functions [45, 46], or different optimisation objectives [47].

References

[1] G. Adomavicius, A. Tuzhilin, Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions, IEEE transactions on knowledge and data engineering 17 (2005) 734–749.

[2] S. Zhang, L. Yao, A. Sun, Y. Tay, Deep learning based recommender system: A survey and new perspectives, ACM Comput. Surv. 52 (2019). URL: https://doi.org/10.1145/3285029. doi:10.1145/3285029.

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

[4] S. Wang, L. Hu, Y. Wang, L. Cao, Q. Z. Sheng, M. Orgun, Sequential recommender systems: challenges, progress and prospects, arXiv preprint arXiv:2001.04830 (2019).

[5] H. Hwangbo, Y. S. Kim, K. J. Cha, Recommendation system development for fashion retail e-commerce, Electronic Commerce Research and Applications 28 (2018) 94–101.

[6] D. Afchar, A. Melchiorre, M. Schedl, R. Hennequin, E. Epure, M. Moussallam, Explainability in music recommender systems, AI Magazine 43 (2022) 190–208.

[7] E. Ferrara, O. Varol, C. Davis, F. Menczer, A. Flammini, The rise of social bots, Communications of the ACM 59 (2016) 96–104.

[8] M. Daniels, Amazon says its stopped 700k counterfeiters from making accounts last year, 2024. URL: https://www.modernretail.co/technology/amazon-says-its-stopped-700k-counterfeiters-from-making-accounts-last

[9] M. Mendoza, M. Tesconi, S. Cresci, Bots in social and interaction networks: detection and impact estimation, ACM Transactions on Information Systems (TOIS) 39 (2020) 1–32.

[10] M. Mazza, S. Cresci, M. Avvenuti, W. Quattrociocchi, M. Tesconi, Rtbust: Exploiting temporal patterns for botnet detection on twitter, in: Proceedings of the 10th ACM conference on web science, 2019, pp. 183–192.

[11] H. Li, S. Di, L. Chen, Revisiting injective attacks on recommender systems, Advances in Neural Information Processing Systems 35 (2022) 29989–30002.

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

[13] B. Hidasi, A. Karatzoglou, L. Baltrunas, D. Tikk, Session-based recommendations with recurrent neural networks, 2016. arXiv:1511.06939.

[14] F. M. Harper, J. A. Konstan, The movielens datasets: History and context, ACM Trans. Interact. Intell. Syst. 5 (2015). URL: https://doi.org/10.1145/2827872. doi:10.1145/2827872.

[15] D. Yang, D. Zhang, V. W. Zheng, Z. Yu, Modeling user activity preference by leveraging user spatial temporal characteristics in lbsns, IEEE Transactions on Systems, Man, and Cybernetics: Systems 45 (2014) 129–142.

[16] J. B. Schafer, J. A. Konstan, J. Riedl, E-commerce recommendation applications, Data Mining and Knowledge Discovery 5 (2001) 115–153.

[17] I. Guy, N. Zwerdling, I. Ronen, D. Carmel, E. Uziel, Social media recommendation based on people and tags,

in: Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '10, Association for Computing Machinery, New York, NY, USA, 2010, p. 194–201. URL: https://doi.org/10.1145/1835449.1835484. doi:10.1145/1835449.1835484.

[18] F. Amato, V. Moscato, A. Picariello, G. Sperlí, Recommendation in social media networks, in: 2017 IEEE Third International Conference on Multimedia Big Data (BigMM), IEEE, 2017, pp. 213–216.

[19] M. Schedl, P. Knees, B. McFee, D. Bogdanov, M. Kaminskas, Music recommender systems, Recommender Systems Handbook (2015) 453–492.

[20] M. Schedl, H. Zamani, C.-W. Chen, Y. Deldjoo, M. Elahi, Current challenges and visions in music recommender systems research, International Journal of Multimedia Information Retrieval 7 (2018) 95–116.

[21] F. Fouss, A. Pirotte, M. Saerens, A novel way of computing similarities between nodes of a graph, with application to collaborative recommendation, in: The 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05), IEEE, 2005, pp. 550–556.

[22] 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.

[23] T. Donkers, B. Loepp, J. Ziegler, Sequential user-based recurrent neural network recommendations, in: Proceedings of the Eleventh ACM Conference on Recommender Systems, RecSys '17, Association for Computing Machinery, New York, NY, USA, 2017, p. 152–160. URL: https://doi.org/10.1145/3109859.3109877. doi:10.1145/3109859.3109877.

[24] M. Quadrana, A. Karatzoglou, B. Hidasi, P. Cremonesi, Personalizing session-based recommendations with hierarchical recurrent neural networks, in: Proceedings of the Eleventh ACM Conference on Recommender Systems, RecSys '17, Association for Computing Machinery, New York, NY, USA, 2017, p. 130–137. URL: https://doi.org/10.1145/3109859.3109896. doi:10.1145/3109859.3109896.

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Advances in neural information processing systems 30 (2017).

[26] 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.

[27] J. Chang, C. Gao, Y. Zheng, Y. Hui, Y. Niu, Y. Song, D. Jin, Y. Li, Sequential recommendation with graph neural networks, in: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '21, Association for Computing Machinery, New York, NY, USA, 2021, p. 378–387. URL: https://doi.org/10.1145/3404835.3462968. doi:10.1145/3404835.3462968.

[28] Z. Fan, Z. Liu, J. Zhang, Y. Xiong, L. Zheng, P. S. Yu, Continuous-time sequential recommendation with temporal graph collaborative transformer, in: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, CIKM '21, Association for Computing Machinery, New York, NY, USA, 2021, p. 433–442. URL: https://doi.org/10.1145/3459637.3482242. doi:10.1145/3459637.3482242.

[29] S. Wu, F. Sun, W. Zhang, X. Xie, B. Cui, Graph neural networks in recommender systems: a survey, ACM Computing Surveys 55 (2022) 1–37.

[30] A. Purificato, G. Cassarà, P. Liò, F. Silvestri, Sheaf neural networks for graph-based recommender systems, arXiv preprint arXiv:2304.09097 (2023).

[31] A. Purificato, F. Silvestri, Eco-aware graph neural networks for sustainable recommendations, arXiv preprint arXiv:2410.09514 (2024).

[32] F. Betello, F. Siciliano, P. Mishra, F. Silvestri, Investigating the robustness of sequential recommender systems against training data perturbations, in: European Conference on Information Retrieval, Springer, 2024, pp. 205–220.

[33] 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, CIKM '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 1584–1594. URL: https://doi.org/10.1145/3511808.3557425. doi:10.1145/3511808.3557425.

[34] M. Yin, Y. Xu, M. Fang, N. Z. Gong, Poisoning federated recommender systems with fake users, in: Proceedings of the ACM on Web Conference 2024, 2024, pp. 3555–3565.

[35] G. Trappolini, V. Maiorca, S. Severino, E. Rodolà, F. Silvestri, G. Tolomei, Sparse vicious attacks on graph neural networks, IEEE Transactions on Artificial Intelligence 5 (2024) 2293–2303. doi:10.1109/TAI.2023.3319306.

[36] Z. Chen, F. Silvestri, J. Wang, Y. Zhang, G. Tolomei, The dark side of explanations: Poisoning recommender systems with counterfactual examples, in: Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '23, Association for Computing Machinery, New York, NY, USA, 2023, p. 2426–2430. URL: https://doi.org/10.1145/3539618.3592070. doi:10.1145/3539618.3592070.

[37] Y. Wang, Y. Liu, Q. Wang, C. Wang, Clusterpoison: Poisoning attacks on recommender systems with limited fake users, IEEE Communications Magazine (2024).

[38] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014. arXiv:1406.1078.

[39] P. Jaccard, The distribution of the flora in the alpine zone. 1, New Phytologist 11 (1912) 37–50.

[40] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2017. arXiv:1412.6980.

[41] F. Betello, A. Purificato, F. Siciliano, G. Trappolini, A. Bacciu, N. Tonellotto, F. Silvestri, A reproducible analysis of sequential recommender systems, IEEE Access (2024).

[42] A. Klenitskiy, A. Volodkevich, A. Pembek, A. Vasilev, Does it look sequential? an analysis of datasets for evaluation of sequential recommendations, arXiv preprint arXiv:2408.12008 (2024).

[43] Y. Hou, J. Li, Z. He, A. Yan, X. Chen, J. McAuley, Bridging language and items for retrieval and recommendation, arXiv preprint arXiv:2403.03952 (2024).

[44] A. Petrov, C. Macdonald, Effective and efficient training for sequential recommendation using recency sam-

pling, in: Proceedings of the 16th ACM Conference on Recommender Systems, 2022, pp. 81–91.
[45] M. S. Bucarelli, L. Cassano, F. Siciliano, A. Mantrach, F. Silvestri, Leveraging inter-rater agreement for classification in the presence of noisy labels, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 3439–3448.
[46] F. A. Wani, M. S. Bucarelli, F. Silvestri, Learning with noisy labels through learnable weighting and centroid similarity, in: 2024 International Joint Conference on Neural Networks (IJCNN), IEEE, 2024, pp. 1–9.
[47] A. Bacciu, F. Siciliano, N. Tonellotto, F. Silvestri, Integrating item relevance in training loss for sequential recommender systems, in: Proceedings of the 17th ACM Conference on Recommender Systems, 2023, pp. 1114–1119.