The Challenging Reproducibility Task in Recommender Systems Research between Traditional and Deep Learning Models

Discussion Paper

Vito Walter Anelli[1], Alejandro Bellogín[2], Antonio Ferrara[1], Daniele Malitesta[1], Felice Antonio Merra[4], Claudio Pomo[1], Francesco Maria Donini[3], Eugenio Di Sciascio[1] and Tommaso Di Noia[1]

[1]*Politecnico di Bari, via Orabona, 4, 70125 Bari, Italy*
[2]*Universidad Autónoma de Madrid, Ciudad Universitaria de Cantoblanco, 28049 Madrid, Spain*
[3]*Università degli Studi della Tuscia, via Santa Maria in Gradi, 4, 01100 Viterbo, Italy*
[4]*Amazon Science Berlin, Invalidenstraße 75, 10557 Berlin, Germany*

Abstract

Recommender Systems have shown to be a useful tool for reducing over-choice and providing accurate, personalized suggestions. The large variety of available recommendation algorithms, splitting techniques, assessment protocols, metrics, and tasks, on the other hand, has made thorough experimental evaluation extremely difficult. Elliot is a comprehensive framework for recommendation with the goal of running and reproducing a whole experimental pipeline from a single configuration file. The framework uses a variety of ways to load, filter, and divide data. Elliot optimizes hyper-parameters for a variety of recommendation algorithms, then chooses the best models, compares them to baselines, computes metrics ranging from accuracy to beyond-accuracy, bias, and fairness, and does statistical analysis. The aim is to provide researchers with a tool to ease all the experimental evaluation phases (and make them reproducible), from data reading to results collection. Elliot is freely available on GitHub at https://github.com/sisinflab/elliot.

Keywords

Recommender Systems, Reproducibility, Adversarial Learning, Visual Recommenders, Knowledge Graphs

1. Introduction

In the last decade, Recommendation Systems (RSs) have gained momentum as the pivotal choice for personalized decision-support systems. Recommendation is essentially a retrieval task where a catalog of items is ranked in a personalized way and the top-scoring items are presented to the user [1]. Once the RSs ability to provide personalized items to clients had been demonstrated, both academia and industry began to devote attention to them [2]. This collective effort resulted in an impressive number of recommendation algorithms, ranging from memory-based [3] to latent factor-based [4, 5], as well as deep learning-based methods [6]. At the same time, the RS research community realized that focusing only on the accuracy of results could be detrimental,

SEBD 2022: The 30th Italian Symposium on Advanced Database Systems, June 19-22, 2022, Tirrenia (PI), Italy
vitowalter.anelli@poliba.it (V. W. Anelli); claudio.pomo@poliba.it (C. Pomo)

and started exploring beyond-accuracy evaluation [7]. As *accuracy* was recognized as insufficient to guarantee users' satisfaction [8], *novelty* and *diversity* [9, 10] came into play as new dimensions to be analyzed when comparing algorithms. However, this was only the first step in the direction of a more comprehensive evaluation. Indeed, more recently, the presence of *biased* [11] and *unfair* recommendations towards user groups and item categories [12] has been widely investigated. The abundance of possible choices has generated confusion around choosing the correct baselines, conducting the hyperparameter optimization and the experimental evaluation [13], and reporting the details of the adopted procedure. Consequently, two major concerns have arisen: unreproducible evaluation and unfair comparisons [14].

The advent of various frameworks over the last decade has improved the research process, and the RS community has gradually embraced the emergence of recommendation, assessment, and even hyperparameter tweaking frameworks. Starting from 2011, Mymedialite [15], LensKit [16], LightFM [17], RankSys [9], and Surprise [18], have formed the basic software for rapid prototyping and testing of recommendation models, thanks to an easy-to-use model execution and the implementation of standard accuracy, and beyond-accuracy, evaluation measures and splitting techniques. However, the outstanding success and the community interest in Deep Learning (DL) recommendation models, raised the need for novel instruments. LibRec [19], Spotlight [1], and OpenRec [20] are the first open-source projects that made DL-based recommenders available – with less than a dozen of available models but, unfortunately, without filtering, splitting, and hyper-optimization tuning strategies. An important step towards more exhaustive and up-to-date set of model implementations have been released with RecQ [21], DeepRec [22], and Cornac [23] frameworks. However, they do not provide a general tool for extensive experiments on the pre-elaboration and the evaluation of a dataset. Indeed, after the reproducibility hype [24, 25], DaisyRec [14] and RecBole [26] raised the bar of framework capabilities, making available both large set of models, data filtering/splitting operations and, above all, hyper-parameter tuning features. Unfortunately, even though these frameworks are a great help to researchers, facilitating reproducibility or extending the provided functionality would typically depend on developing bash scripts or programming on whatever language each framework is written.

This is where Elliot comes to the stage. It is a novel kind of recommendation framework, aimed to overcome these obstacles by proposing a fully declarative approach (by means of a configuration file) to the set-up of an experimental setting. It analyzes the recommendation problem from the researcher's perspective as it implements the whole experimental pipeline, from dataset loading to results gathering in a principled way. The main idea behind Elliot is to keep an entire experiment reproducible and put the user (in our case, a researcher or RS developer) in control of the framework. To date, according to the recommendation model, Elliot allows for choosing among 27 similarity metrics, defining of multiple neural architectures, and choosing 51 hyperparameter tuning combined approaches, unleashing the full potential of the HyperOpt library [27]. To enable evaluation for the diverse tasks and domains, Elliot supplies 36 metrics (including Accuracy, Error-based, Coverage, Novelty, Diversity, Bias, and Fairness metrics), 13 splitting strategies, and 8 prefiltering policies.

[1] https://github.com/maciejkula/spotlight

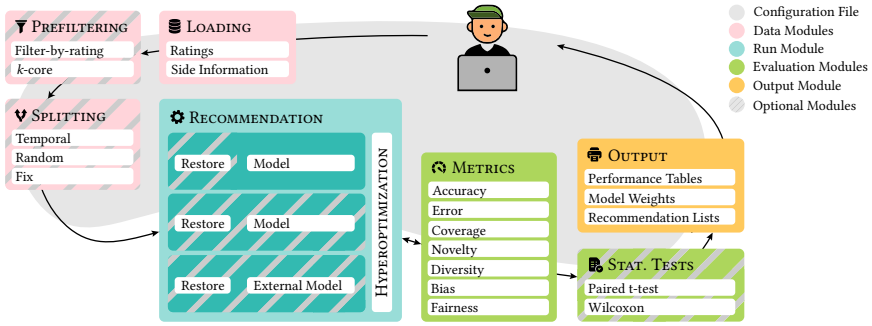

Figure 1: Overview of Elliot.

2. Framework

Elliot is an extendable framework made up of eight functional modules, each of which is in charge of a different phase in the experimental suggestion process. The user is only required to submit human-level experimental flow information via a customisable configuration file, so what happens beneath the hood (Figure 1) is transparent to them. As a result, Elliot constructs the whole pipeline. What follows presents each of Elliot's modules and how to create a configuration file.

2.1. Data Preparation

The *Data* modules are in charge of handling and organizing the experiment's input, as well as providing a variety of supplementary data, such as item characteristics, visual embeddings, and pictures. The input data is taken over by the *Prefiltering* and *Splitting* modules after being loaded by the *Loading* module, whose techniques are described in Sect.2.1.2 and 2.1.3 respectively.

2.1.1. Loading

Different data sources, such as user-item feedback or side information, such as the item visual aspects, may be required for RSs investigations. Elliot comes with a variety of *Loading* module implementations to meet these requirements. Furthermore, the user may create computationally intensive prefiltering and splitting operations that can be saved and loaded to save time in the future. Additional data, such as visual characteristics and semantic features generated from knowledge graphs, can be handled through data-driven extensions. When a side-information-aware *Loading* module is selected, it filters out items that lack the needed information to provide a fair comparison.

2.1.2. Prefiltering

Elliot provides data filtering procedures using two different techniques after data loading. *Filter-by-rating* is the first method implemented in the *Prefiltering* module, which removes a user-item interaction if the preference score falls below a certain level. It can be a *Numerical* value, such as 3.5, a *Distributional* information, such as the worldwide rating average value, or a user-based

distributional (*User Dist.*) value, such as the user's average rating value. The k-*core* prefiltering approach eliminates people, objects, or both if there are less than k documented interactions. The k-*core* technique can be used to both users and things repeatedly (Iterative k-*core*) until the k-*core* filtering requirement is fulfilled, i.e., all users and items have at least k recorded interactions. Since reaching such condition might be intractable, ELLIOT allows specifying the maximum number of iterations (*Iter-*n*-rounds*). Finally, the *Cold-Users* filtering feature allows retaining cold-users only.

2.1.3. Splitting

ELLIOT implements three splitting strategies: (i) *Temporal*, (ii) *Random*, and (iii) *Fix*. The *Temporal* method divides user-item interactions depending on the transaction timestamp, either by setting the timestamp, selecting the best one [28, 29], or using a hold-out (*HO*) mechanism. Hold-out (*HO*), K-repeated hold-out (*K-HO*), and cross-validation (*CV*) are all part of the *Random* methods. Finally, the *Fix* approach leverages an already split dataset.

2.2. Recommendation Models

After data loading and pre-elaborations, the *Recommendation* module (Figure 1) provides the functionalities to train (and restore) both ELLIOT's state-of-the-art recommendation models and custom user-implemented models, with the possibility to find the best hyper-parameter setting.

2.2.1. Implemented Models

To date, ELLIOT integrates around 50 recommendation models grouped into two sets: (i) *popular* models implemented in at least two of the other reviewed frameworks, and (ii) other well-known state-of-the-art recommendation models which are less common in the reviewed frameworks, such as autoencoder-based, e.g., [6], graph-based, e.g., [30], visually-aware [31], e.g., [32], adversarially-robust, e.g., [33], and content-aware, e.g., [34, 35].

2.2.2. Hyper-parameter Tuning

According to Rendle et al. [25], Anelli et al. [36], hyper-parameter optimization has a significant impact on performance. ELLIOT supplies *Grid Search*, *Simulated Annealing*, *Bayesian Optimization*, and *Random Search*, supporting four different traversal techniques in the search space. *Grid Search* is automatically inferred when the user specifies the available hyper-parameters.

2.3. Performance Evaluation

After the training phase, ELLIOT continues its operations, evaluating recommendations. Figure 1 indicates this phase with two distinct evaluation modules: Metrics and Statistical Tests.

2.3.1. Metrics

Elliot provides a set of 36 evaluation metrics, partitioned into seven families: *Accuracy*, *Error*, *Coverage*, *Novelty*, *Diversity*, *Bias*, and *Fairness*. It is worth mentioning that Elliot is the framework that exposes both the largest number of metrics and the only one considering bias and fairness measures. Moreover, the practitioner can choose any metric to drive the model selection and the tuning.

2.3.2. Statistical Tests

All other cited frameworks do not support statistical hypothesis tests, probably due to the need for computing fine-grained (e.g., per-user or per-partition) results and retaining them for each recommendation model. Conversely, Elliot helps computing two statistical hypothesis tests, i.e., *Wilcoxon* and *Paired t-test*, with a flag in the configuration file.

2.4. Framework Outcomes

When the training of recommenders is over, Elliot uses the *Output* module to gather the results. Three types of output files can be generated: (i) *Performance Tables*, (ii) *Model Weights*, and (iii) *Recommendation Lists*. Performance Tables come in the form of spreadsheets, including all the metric values generated on the test set for each recommendation model given in the configuration file. Cut-off-specific and model-specific tables are included in a final report (i.e., considering each combination of the explored parameters). Statistical hypothesis tests are also presented in the tables, as well as a JSON file that summarizes the optimal model parameters. Optionally, Elliot stores the model weights for the sake of future re-training.

2.5. Preparation of the Experiment

Elliot is triggered by a single configuration file written in YAML (e.g., refer to the toy example `sample_hello_world.yml`). The first section details the data loading, filtering, and splitting information defined in Section 2.1. The `models` section represents the recommendation models' configuration, e.g., Item-kNN. Here, the model-specific hyperparameter optimization strategies are specified, e.g., the grid-search. The `evaluation` section details the evaluation strategy with the desired metrics, e.g., nDCG in the toy example. Finally, `save_recs` and `top_k` keys detail, for example, the *Output* module abilities described in Section 2.4.

3. Conclusion and Future Work

Elliot is a framework that looks at the recommendation process from the eyes of an RS researcher. To undertake a thorough and repeatable experimental assessment, the user only has to generate a flexible configuration file. Several loading, prefiltering, splitting, hyperparameter optimization, recommendation models, and statistical hypothesis testing are included in the framework. Elliot reports may be evaluated and used directly into research papers. We evaluated the RS assessment literature, putting Elliot in the context of the other frameworks and highlighted its benefits and drawbacks. Following that, we looked at the framework's design

and how to create a functional (and repeatable) experimental benchmark. Elliot is the only recommendation framework we're aware of that supports a full multi-recommender experimental pipeline from a single configuration file. We intend to expand the framework in the near future to incorporate sequential recommendation scenarios, adversarial attacks, reinforcement learning-based recommendation systems, differential privacy facilities, sampling assessment, and distributed recommendation, among other things.

References

[1] W. Krichene, S. Rendle, On sampled metrics for item recommendation, in: R. Gupta, Y. Liu, J. Tang, B. A. Prakash (Eds.), KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020, ACM, 2020, pp. 1748–1757. URL: https://dl.acm.org/doi/10.1145/3394486.3403226.

[2] J. Bennett, S. Lanning, The netflix prize, in: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Jose, California, USA, August 12-15, 2007, ACM, 2007.

[3] B. M. Sarwar, G. Karypis, J. A. Konstan, J. Riedl, Item-based collaborative filtering recommendation algorithms, in: V. Y. Shen, N. Saito, M. R. Lyu, M. E. Zurko (Eds.), WWW 2001, ACM, 2001, pp. 285–295. doi:10.1145/371920.372071.

[4] Y. Koren, R. M. Bell, Advances in collaborative filtering, in: F. Ricci, L. Rokach, B. Shapira (Eds.), Recommender Systems Handbook, Springer, 2015, pp. 77–118. doi:10.1007/978-1-4899-7637-6_3.

[5] S. Rendle, Factorization machines, in: G. I. Webb, B. Liu, C. Zhang, D. Gunopulos, X. Wu (Eds.), ICDM 2010, The 10th IEEE International Conference on Data Mining, Sydney, Australia, 14-17 December 2010, IEEE Computer Society, 2010, pp. 995–1000. doi:10.1109/ICDM.2010.127.

[6] D. Liang, R. G. Krishnan, M. D. Hoffman, T. Jebara, Variational autoencoders for collaborative filtering, in: P. Champin, F. L. Gandon, M. Lalmas, P. G. Ipeirotis (Eds.), WWW 2018, ACM, 2018, pp. 689–698. doi:10.1145/3178876.3186150.

[7] S. Vargas, P. Castells, Rank and relevance in novelty and diversity metrics for recommender systems, in: B. Mobasher, R. D. Burke, D. Jannach, G. Adomavicius (Eds.), RecSys 2011, ACM, 2011, pp. 109–116. URL: https://dl.acm.org/citation.cfm?id=2043955.

[8] S. M. McNee, J. Riedl, J. A. Konstan, Being accurate is not enough: how accuracy metrics have hurt recommender systems, in: G. M. Olson, R. Jeffries (Eds.), Extended Abstracts Proceedings of the 2006 Conference on Human Factors in Computing Systems, CHI 2006, Montréal, Québec, Canada, April 22-27, 2006, ACM, 2006, pp. 1097–1101. doi:10.1145/1125451.1125659.

[9] S. Vargas, Novelty and diversity enhancement and evaluation in recommender systems and information retrieval, in: S. Geva, A. Trotman, P. Bruza, C. L. A. Clarke, K. Järvelin (Eds.), SIGIR 2014, ACM, 2014, p. 1281. doi:10.1145/2600428.2610382.

[10] P. Castells, N. J. Hurley, S. Vargas, Novelty and diversity in recommender systems, in: F. Ricci, L. Rokach, B. Shapira (Eds.), Recommender Systems Handbook,

Springer, 2015, pp. 881–918. URL: https://doi.org/10.1007/978-1-4899-7637-6_26. doi:10.1007/978-1-4899-7637-6_26.

[11] Z. Zhu, Y. He, X. Zhao, Y. Zhang, J. Wang, J. Caverlee, Popularity-opportunity bias in collaborative filtering, in: WSDM 2021, ACM, 2021. doi:https://doi.org/10.1145/3437963.3441820.

[12] Y. Deldjoo, V. W. Anelli, H. Zamani, A. Bellogin, T. Di Noia, A flexible framework for evaluating user and item fairness in recommender systems, User Modeling and User-Adapted Interaction (2020) 1–47.

[13] A. Said, A. Bellogín, Comparative recommender system evaluation: benchmarking recommendation frameworks, in: A. Kobsa, M. X. Zhou, M. Ester, Y. Koren (Eds.), RecSys 2014, ACM, 2014, pp. 129–136. doi:10.1145/2645710.2645746.

[14] Z. Sun, D. Yu, H. Fang, J. Yang, X. Qu, J. Zhang, C. Geng, Are we evaluating rigorously? benchmarking recommendation for reproducible evaluation and fair comparison, in: R. L. T. Santos, L. B. Marinho, E. M. Daly, L. Chen, K. Falk, N. Koenigstein, E. S. de Moura (Eds.), RecSys 2020, ACM, 2020, pp. 23–32. doi:10.1145/3383313.3412489.

[15] Z. Gantner, S. Rendle, C. Freudenthaler, L. Schmidt-Thieme, Mymedialite: a free recommender system library, in: B. Mobasher, R. D. Burke, D. Jannach, G. Adomavicius (Eds.), RecSys 2011, ACM, 2011, pp. 305–308. doi:10.1145/2043932.2043989.

[16] M. D. Ekstrand, Lenskit for python: Next-generation software for recommender systems experiments, in: M. d'Aquin, S. Dietze, C. Hauff, E. Curry, P. Cudré-Mauroux (Eds.), CIKM 2020, ACM, 2020, pp. 2999–3006. doi:10.1145/3340531.3412778.

[17] M. Kula, Metadata embeddings for user and item cold-start recommendations, in: T. Bogers, M. Koolen (Eds.), Proceedings of the 2nd Workshop on New Trends on Content-Based Recommender Systems co-located with 9th ACM Conference on Recommender Systems (RecSys 2015), Vienna, Austria, September 16-20, 2015., volume 1448 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2015, pp. 14–21. URL: http://ceur-ws.org/Vol-1448/paper4.pdf.

[18] N. Hug, Surprise: A python library for recommender systems, J. Open Source Softw. 5 (2020) 2174. doi:10.21105/joss.02174.

[19] G. Guo, J. Zhang, Z. Sun, N. Yorke-Smith, Librec: A java library for recommender systems, in: A. I. Cristea, J. Masthoff, A. Said, N. Tintarev (Eds.), Posters, Demos, Late-breaking Results and Workshop Proceedings of the 23rd Conference on User Modeling, Adaptation, and Personalization (UMAP 2015), Dublin, Ireland, June 29 - July 3, 2015, volume 1388 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2015.

[20] L. Yang, E. Bagdasaryan, J. Gruenstein, C. Hsieh, D. Estrin, Openrec: A modular framework for extensible and adaptable recommendation algorithms, in: Y. Chang, C. Zhai, Y. Liu, Y. Maarek (Eds.), WSDM 2018, ACM, 2018, pp. 664–672. doi:10.1145/3159652.3159681.

[21] J. Yu, M. Gao, H. Yin, J. Li, C. Gao, Q. Wang, Generating reliable friends via adversarial training to improve social recommendation, in: J. Wang, K. Shim, X. Wu (Eds.), 2019 IEEE International Conference on Data Mining, ICDM 2019, Beijing, China, November 8-11, 2019, IEEE, 2019, pp. 768–777. doi:10.1109/ICDM.2019.00087.

[22] U. Gupta, S. Hsia, V. Saraph, X. Wang, B. Reagen, G. Wei, H. S. Lee, D. Brooks, C. Wu, Deeprecsys: A system for optimizing end-to-end at-scale neural recommendation inference, in: 47th ACM/IEEE Annual International Symposium on Computer Architecture, ISCA 2020, Valencia, Spain, May 30 - June 3, 2020, IEEE, 2020, pp. 982–995.

doi:10.1109/ISCA45697.2020.00084.

[23] A. Salah, Q. Truong, H. W. Lauw, Cornac: A comparative framework for multimodal recommender systems, J. Mach. Learn. Res. 21 (2020) 95:1–95:5. URL: http://jmlr.org/papers/v21/19-805.html.

[24] M. F. Dacrema, P. Cremonesi, D. Jannach, Are we really making much progress? A worrying analysis of recent neural recommendation approaches, in: RecSys, ACM, 2019, pp. 101–109.

[25] S. Rendle, W. Krichene, L. Zhang, J. R. Anderson, Neural collaborative filtering vs. matrix factorization revisited, in: RecSys, ACM, 2020, pp. 240–248.

[26] W. X. Zhao, S. Mu, Y. Hou, Z. Lin, K. Li, Y. Chen, Y. Lu, H. Wang, C. Tian, X. Pan, Y. Min, Z. Feng, X. Fan, X. Chen, P. Wang, W. Ji, Y. Li, X. Wang, J. Wen, Recbole: Towards a unified, comprehensive and efficient framework for recommendation algorithms, CoRR abs/2011.01731 (2020). URL: https://arxiv.org/abs/2011.01731. arXiv:2011.01731.

[27] J. Bergstra, D. Yamins, D. D. Cox, Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures, in: Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013, volume 28 of *JMLR Workshop and Conference Proceedings*, JMLR.org, 2013, pp. 115–123. URL: http://proceedings.mlr.press/v28/bergstra13.html.

[28] V. W. Anelli, T. D. Noia, E. D. Sciascio, A. Ragone, J. Trotta, Local popularity and time in top-n recommendation, in: ECIR (1), volume 11437 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 861–868.

[29] A. Bellogín, P. Sánchez, Revisiting neighbourhood-based recommenders for temporal scenarios, in: RecTemp@RecSys, volume 1922 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2017, pp. 40–44.

[30] X. Wang, X. He, M. Wang, F. Feng, T. Chua, Neural graph collaborative filtering, in: B. Piwowarski, M. Chevalier, É. Gaussier, Y. Maarek, J. Nie, F. Scholer (Eds.), SIGIR 2019, ACM, 2019, pp. 165–174. doi:10.1145/3331184.3331267.

[31] V. W. Anelli, A. Bellogín, A. Ferrara, D. Malitesta, F. A. Merra, C. Pomo, F. M. Donini, T. D. Noia, V-elliot: Design, evaluate and tune visual recommender systems, in: RecSys, ACM, 2021, pp. 768–771.

[32] R. He, J. J. McAuley, VBPR: visual bayesian personalized ranking from implicit feedback, in: D. Schuurmans, M. P. Wellman (Eds.), Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA, AAAI Press, 2016, pp. 144–150. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/11914.

[33] J. Tang, X. Du, X. He, F. Yuan, Q. Tian, T. Chua, Adversarial training towards robust multimedia recommender system, IEEE Trans. Knowl. Data Eng. 32 (2020) 855–867. doi:10.1109/TKDE.2019.2893638.

[34] V. W. Anelli, T. D. Noia, E. D. Sciascio, A. Ragone, J. Trotta, How to make latent factors interpretable by feeding factorization machines with knowledge graphs, in: ISWC (1), volume 11778 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 38–56.

[35] V. W. Anelli, T. D. Noia, E. D. Sciascio, A. Ferrara, A. C. M. Mancino, Sparse feature factorization for recommender systems with knowledge graphs, in: RecSys, ACM, 2021, pp. 154–165.

[36] V. W. Anelli, T. D. Noia, E. D. Sciascio, C. Pomo, A. Ragone, On the discriminative power of hyper-parameters in cross-validation and how to choose them, in: RecSys, ACM, 2019, pp. 447–451.