Fairness-aware Recommendation with librec-auto

Nasim Sonboli
University of Colorado, Boulder
Boulder, CO, USA
nasim.sonboli@colorado.edu

Robin Burke
University of Colorado, Boulder
Boulder, CO, USA
robin.burke@colorado.edu

Zijun Liu
University of Colorado, Boulder
Boulder, CO, USA
zijun.liu@colorado.edu

Masoud Mansoury
Eindhoven University of Technology
Eindhoven, the Netherlands
m.mansoury@tue.nl

ABSTRACT

Comparative experimentation is important for studying reproducibility in recommender systems. This is particularly true in areas without well-established methodologies, such as fairness-aware recommendation. In this paper, we describe fairness-aware enhancements to our recommender systems experimentation tool `librec-auto`. These enhancements include metrics for various classes of fairness definitions, extension of the experimental model to support result re-ranking and a library of associated re-ranking algorithms, and additional support for experiment automation and reporting. The associated demo will help attendees move quickly to configuring and running their own experiments with `librec-auto`.

CCS CONCEPTS

• **Information systems → Recommender systems**; • **Software and its engineering → Software libraries and repositories**.

KEYWORDS

Recommender Systems Frameworks; Experimentation; Librec; Fairness; Reranking

ACM Reference Format:
Nasim Sonboli, Robin Burke, Zijun Liu, and Masoud Mansoury. 2020. Fairness-aware Recommendation with librec-auto. In *Fourteenth ACM Conference on Recommender Systems (RecSys '20), September 22–26, 2020, Virtual Event, Brazil.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3383313.3411525

1 INTRODUCTION

Despite progress in recent years, reproducibility remains a challenge in recommender systems research [3]. Minor differences in parameters and experimental settings can yield incompatible results, which make it difficult to provide definitive answers about the relative properties of different algorithms. Progress in this area is supported by providing platforms on which comparative experiments can be conducted using declarative experimental configuration (so that experimental settings can be easily shared), with pre-implemented methodological workflows, and with a large library of algorithms for rapid benchmarking.

In [15], we introduced `librec-auto`, an open-source command-line Python package providing a wrapper for the LibRec 2.0 recommender systems algorithm library[1]. Key advantages of `librec-auto` were its ability to support typical research workflows, to offer a declaration configuration system, and to supplement experiment execution with the scripted production of human-friendly outputs including visualizations.

We have now extended this platform in a number of ways, particularly to support research in fairness-aware recommendation. `librec-auto` now supports the current 3.0 version of the LibRec library and can take advantage of the new algorithms (including deep learning algorithms) found there. The `librec-auto` project has also enhanced LibRec with a suite of metrics for measuring the fairness of recommendation outcomes. Most significantly, the tool now supports recommendation re-ranking, a common approach to enhancing fairness, diversity, and other non-accuracy properties of recommendation outcomes.

2 CORE FEATURES

LibRec 3.0 is a Java-based recommendation generation platform. It has been available to the recommender systems community since 2015 [11], and has large library of implemented recommendation algorithms (more than 70 as of this writing). The platform supports a variety of evaluation metrics and evaluation methodologies. However, our experience indicates that for practical experimentation and reproducibility research, LibRec by itself is not sufficient. For example, intermediate computational outputs, such as recommendation results, cannot be reused as input for new evaluation metrics, requiring the re-execution of potentially lengthy experiment executions.

We developed `librec-auto`[2] to retain the benefits of working with LibRec while adding support for experimentation. A sketch of the functionality of `librec-auto` is provided in Figure 1. As the figure indicates, LibRec is encapsulated and its various component elements are used to execute particular portions of the experimental workflow. In addition, the optional re-ranking component allows for the study of re-ranking algorithms not within the scope of LibRec's design. The key inputs are data and an XML-based configuration file. A particular study may consist of multiple experiments, all of

RecSys '20, September 22–26, 2020, Virtual Event, Brazil

ACM ISBN 978-1-4503-7583-2/20/09.
https://doi.org/10.1145/3383313.3411525

[1] www.librec.net

[2] github.com/that-recsys-lab/librec-auto

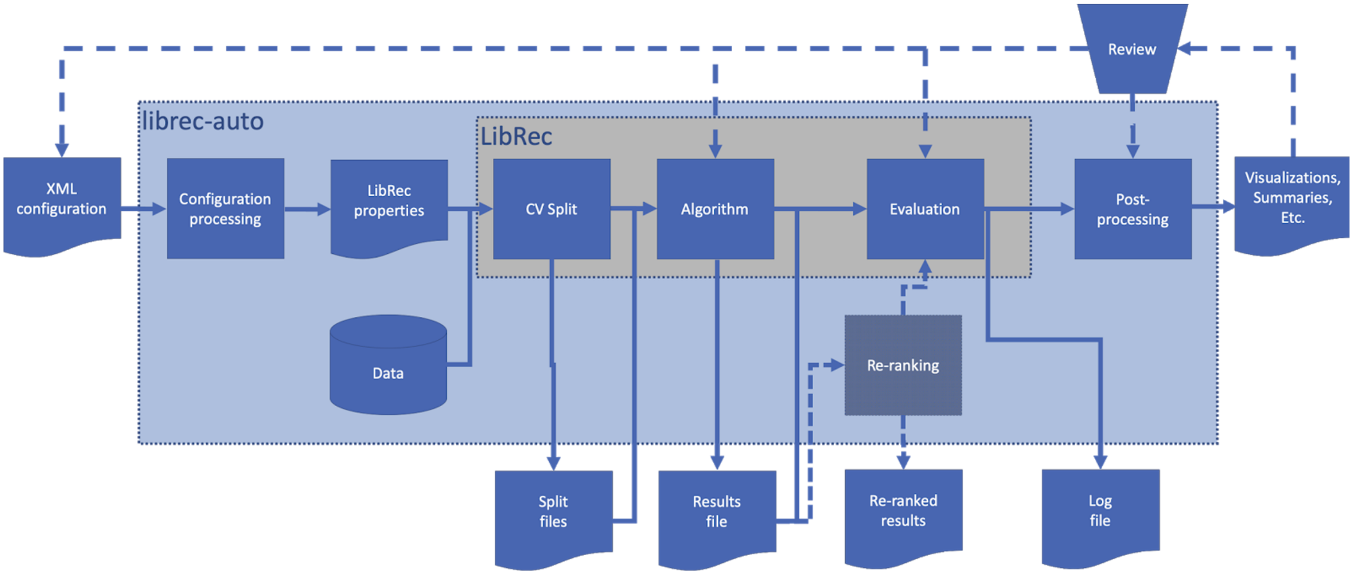

Figure 1: Schematic of experimentation workflow with librec-auto. The LibRec library (Java, shown in grey) is encapsulated by librec-auto (Python, shown in blue), which manages configuration, experimental outputs and post-processs. Added from [15] is the new re-ranking module shown in dark blue.

which are configured at the same time in the configuration file. Configuration files are modular, so that, for example, multiple studies can share the same methodology elements, preventing inadvertent misconfiguration.

Although the figure indicates a straight-line of execution, parallelism is built into librec-auto at the level of experiment execution. Because experiments can have lengthy execution times, the post-processing phase allows for integration with messaging platforms, including Slack, so that experimenters are notified when their tasks are complete. These messages can include visualizations of experimental output, to provide a quick overview of results.

3 FAIRNESS-AWARE EXTENSIONS

Although librec-auto has been under development since 2018, the latest release incorporates several key advances that specifically support common tasks in the study of recommendation fairness. These advances are (1) new evaluation metrics that report on fairness aspects of recommendation output, (2) an optional re-ranking step in the experiment pipeline, to support what is one of the most common category of fairness enhancing techniques, and (3) additional support for working with user (demographic) and item (content) features in algorithms and metrics. With these features, librec-auto now can support a wide range of research activities in fairness-aware recommendation, and we will be adding additional capabilities in future releases.

Previously in the literature, many re-ranking algorithms have been proposed to achieve a balance between diversity and accuracy. The following methods try to achieve a fair representation between groups by penalizing the score of over-represented groups or reinforcing the score of the under-represented groups: (1) **FAR**, defined in [14], combines a personalization-induced and fairness-induced scores with hyper-parameter λ; (2) **PFAR**, from [14], adds a personalized weight to FAR, calculated based on item-features in user profile, representing the tolerance of the user for diverse resultsl and (3) **OFAiR** incorporates similar personalization and allows fine-grained control of protected group promotion when there are multiple protected groups [20]. By contrast, (4) **FA*IR** [25] builds a queues of protected and unprotected items and draws from each queue to build the final re-ranked list. We also include two more general diversity-enhancing re-rankers first promoted in the information retrieval literature: (5) **MMR** diversifies result lists by greedily adding items with maximal marginal relevance [8], and (6) **XQuAD** defined in [18] has similar goal to MMR algorithm, but it enhances diversity with respect to specific aspects. Finally, we include (7) **Calibrated Recommendations**, an algorithm closely tied to the Calibration metric above, which re-ranks recommendations to ensure a close match to the user's distribution of interests in item features [21]. The re-ranking methods are part of librec-auto and are implemented in Python.

Recommendation fairness and associated fairness metrics can be defined from the perspective of two main stakeholders: providers and consumers [7]. Additionally, both provider-side and consumer-side metrics come in two basic varieties: exposure-based and hit-based. *Exposure* metrics focus on the the appearance of protected items [19] in a ranked list and *hit-based* metrics take into account the suitability of the target user [1]. Many metrics have been offered to measure recommendation fairness [4, 6, 9, 13, 21–24]. We implement the following metrics in librec-auto and where possible, both consumer-side and item-side versions of the metric are available: (1) **Discounted Proportional Fairness** (DPF), a hit-based fairness metric similar to the metric offered in [9] where it measures the ranking utility (nDCG) of the protected group with respect to the other groups. (2) **Calibration** [21], a distribution-based metric that uses KL-Divergence to measure the difference in item category distribution between the preferences of users and their respective recommendation lists. (3) **Statistical parity**, based on the ideas discussed in [17, 26], measuring the difference in outcomes between protected and unprotected groups relative to various recommendation outcomes. Both ranking and prediction accuracy measures are supported. (4) **P-Percent-Rule** (PPR) discussed in [5], is a two-sided extension of statistical parity [2]. (5) **Error-based** metrics proposed in Yao et al. [24] including value-unfairness, absolute unfairness, underestimation unfairness, overestimation unfairness, and non-parity unfairness by Kamishima et al. [12]. Additionally, we offer the following diversity-based metrics (6) **Intra-list distance (ILD)** [28], a pairwise distance between all the item features in each user's recommendation list, and (7) **Gini Index** calculated over the exposure of all the present groups in the recommendation list. All metrics are implemented in Java and integrated with the LibRec code base.

We plan future releases of `librec-auto` to include integration with additional recommendation libraries, including LKPY [10], LibFM [16], and DeepRec [27].

ACKNOWLEDGMENTS

Authors Burke and Sonboli were supported in part by the National Science Foundation under Grant Number: 1911025.

REFERENCES

[1] Himan Abdollahpouri, Gediminas Adomavicius, Robin Burke, Ido Guy, Dietmar Jannach, Toshihiro Kamishima, Jan Krasnodebski, and Luiz Pizzato. 2020. Multistakeholder recommendation: Survey and research directions. *User Modeling and User-Adapted Interaction* 30, 1 (2020), 127–158.

[2] Solon Barocas and Andrew D Selbst. 2016. Big data's disparate impact. *Calif. L. Rev.* 104 (2016), 671.

[3] Joeran Beel, Corinna Breitinger, Stefan Langer, Andreas Lommatzsch, and Bela Gipp. 2016. Towards reproducibility in recommender-systems research. *User modeling and user-adapted interaction* 26, 1 (2016), 69–101.

[4] Alex Beutel, Jilin Chen, Tulsee Doshi, Hai Qian, Li Wei, Yi Wu, Lukasz Heldt, Zhe Zhao, Lichan Hong, Ed H Chi, et al. 2019. Fairness in recommendation ranking through pairwise comparisons. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* 2212–2220.

[5] Dan Biddle. 2006. *Adverse impact and test validation: A practitioner's guide to valid and defensible employment testing.* Gower Publishing, Ltd.

[6] Asia J Biega, Krishna P Gummadi, and Gerhard Weikum. 2018. Equity of attention: Amortizing individual fairness in rankings. In *The 41st international acm sigir conference on research & development in information retrieval.* 405–414.

[7] Robin Burke. 2017. Multisided Fairness for Recommendation. In *Workshop on Fairness, Accountability and Transparency in Machine Learning (FATML).* Halifax, Nova Scotia.

[8] Jaime Carbonell and Jade Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval.* 335–336.

[9] Carlos Castillo. 2019. Fairness and transparency in ranking. In *ACM SIGIR Forum*, Vol. 52. ACM New York, NY, USA, 64–71.

[10] Michael D Ekstrand. 2018. The LKPY package for recommender systems experiments: Next-generation tools and lessons learned from the LensKit project. *arXiv preprint arXiv:1809.03125* (2018).

[11] Guibing Guo, Jie Zhang, Zhu Sun, and Neil Yorke-Smith. 2015. LibRec: A Java Library for Recommender Systems.. In *UMAP Workshops*, Vol. 4.

[12] Toshihiro Kamishima, Shotaro Akaho, and Jun Sakuma. 2011. Fairness-aware learning through regularization approach. In *2011 IEEE 11th International Conference on Data Mining Workshops.* IEEE, 643–650.

[13] Caitlin Kuhlman, MaryAnn VanValkenburg, and Elke Rundensteiner. 2019. FARE: Diagnostics for Fair Ranking using Pairwise Error Metrics. In *The World Wide Web Conference.* 2936–2942.

[14] Weiwen Liu, Jun Guo, Nasim Sonboli, Robin Burke, and Shengyu Zhang. 2019. Personalized Fairness-Aware Re-Ranking for Microlending. In *Proceedings of the 13th ACM Conference on Recommender Systems* (Copenhagen, Denmark) *(RecSys '19).* Association for Computing Machinery, New York, NY, USA, 467–471. https://doi.org/10.1145/3298689.3347016

[15] Masoud Mansoury, Robin Burke, Aldo Ordonez-Gauger, and Xavier Sepulveda. 2018. Automating recommender systems experimentation with librec-auto. In *Proceedings of the 12th ACM Conference on Recommender Systems.* 500–501.

[16] Steffen Rendle. 2012. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST)* 3, 3 (2012), 1–22.

[17] Ya'acov Ritov, Yuekai Sun, and Ruofei Zhao. 2017. On conditional parity as a notion of non-discrimination in machine learning. *arXiv preprint arXiv:1706.08519* (2017).

[18] Rodrygo LT Santos, Jie Peng, Craig Macdonald, and Iadh Ounis. 2010. Explicit search result diversification through sub-queries. In *European conference on information retrieval.* Springer, 87–99.

[19] Ashudeep Singh and Thorsten Joachims. 2018. Fairness of exposure in rankings. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* 2219–2228.

[20] Nasim Sonboli, Farzad Eskandanian, Robin Burke, Weiwen Liu, and Bamshad Mobasher. 2020. Opportunistic Multi-aspect Fairness through Personalized Reranking. In *Proceedings of the 28th ACM Conference on User Modeling, Adaptation and Personalization.* to appear.

[21] Harald Steck. 2018. Calibrated recommendations. In *Proceedings of the 12th ACM conference on recommender systems.* 154–162.

[22] Virginia Tsintzou, Evaggelia Pitoura, and Panayiotis Tsaparas. 2018. Bias disparity in recommendation systems. *arXiv preprint arXiv:1811.01461* (2018).

[23] Ke Yang and Julia Stoyanovich. 2017. Measuring fairness in ranked outputs. In *Proceedings of the 29th International Conference on Scientific and Statistical Database Management.* 1–6.

[24] Sirui Yao and Bert Huang. 2017. Beyond parity: Fairness objectives for collaborative filtering. In *Advances in Neural Information Processing Systems.* 2921–2930.

[25] Meike Zehlike, Francesco Bonchi, Carlos Castillo, Sara Hajian, Mohamed Megahed, and Ricardo Baeza-Yates. 2017. Fa* ir: A fair top-k ranking algorithm. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management.* 1569–1578.

[26] Rich Zemel, Yu Wu, Kevin Swersky, Toni Pitassi, and Cynthia Dwork. 2013. Learning fair representations. In *International Conference on Machine Learning.* 325–333.

[27] Shuai Zhang, Yi Tay, Lina Yao, Bin Wu, and Aixin Sun. 2019. Deeprec: An open-source toolkit for deep learning based recommendation. *arXiv preprint arXiv:1905.10536* (2019).

[28] Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. 2005. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web.* 22–32.