OmniRec: The All-In-One Solution for Reproducible and Interoperable Recommender Systems Experimentation

Lukas Wegmeth[1][0000−0001−8848−9434], Moritz Baumgart[1,2][0009−0007−1322−1450], Philipp Meister[1,2][0009−0008−6814−9668], Bela Gipp[2][0000−0001−6522−3019], and Joeran Beel[1,3][0000−0002−4537−5573]

[1] University of Siegen, Germany
{FirstName.LastName}@uni-siegen.de
[2] University of Göttingen, Germany
{FirstName.LastName}@uni-goettingen.de
[3] Recommender-Systems.com, Germany

Abstract. Recommender systems researchers rely heavily on general-purpose libraries that facilitate data preprocessing, model training, and evaluation. However, existing frameworks often suffer from fragmented data handling, inconsistent preprocessing, limited interoperability, and poor dataset referencing, which hinder reproducibility and comparability between studies. We present OmniRec, an open-source Python library designed to address these limitations. OmniRec provides standardized access to more than 230 datasets, a unified and flexible preprocessing pipeline, and seamless integration with multiple state-of-the-art recommender system frameworks, including RecPack, RecBole, Lenskit, and Elliot. Its modular architecture allows researchers to easily integrate new datasets, customize preprocessing steps, and external model interfaces. By combining ease of use, transparency, and reproducibility, OmniRec simplifies experimentation and fosters a more open and collaborative ecosystem for recommender systems research and practice.

Keywords: Recommender Systems · Framework · Reproducibility · Evaluation · Benchmarking.

1 Introduction

The growth of the number and size of datasets, models, and evaluation protocols for recommender systems has created both opportunities and challenges. Concurrently, recommender systems research has relied on general-purpose libraries that simplify the development and evaluation of models. While powerful frameworks exist to train and evaluate recommendation models, the community still struggles with fragmented data handling, inconsistent preprocessing, and limited interoperability between toolkits [17].

Frameworks such as RecBole [29], Lenskit [6], Lenskit-Auto [22], Elliot [2], RecPack [18], Surprise [10], Auto-Surprise [1], ClayRS [16], DaisyRec [21], Microsoft Recommenders [8], RecStudio [15], RecList [5], Spotlight [13], Cornac

[19], LightFM [12], TorchRec [11], and ReChorus [14] have become integral to both academic and applied work, providing implementations of data preprocessing, model training, and evaluation protocols [28,23,17].

While such frameworks often support multiple input formats, pre-filtering options, and a variety of dataset splitting strategies, recent work has highlighted substantial shortcomings in current practice [17]. Pipelines vary drastically across frameworks, making comparisons opaque and hindering reproducibility due to three primary problems: (1) fragmented and incompatible data handling, (2) inadequate dataset referencing and versioning, and (3) inconsistent implementation of core methods. First, data handling is often fragmented, with each framework implementing its own incompatible preprocessing logic, preventing datasets from being shared or used across different frameworks. Second, poor dataset referencing and versioning, including missing citations, undocumented modifications, and broken download links, undermine applicability. Finally, preprocessing and splitting methods, such as feedback conversion or k-core filtering, are implemented inconsistently or absent altogether, and many frameworks remain closed in design, lacking standard export formats or integration APIs.

These limitations create two major problems: reduced accessibility for newcomers and a lack of scientific rigor. Reduced accessibility is evident in the steep entry barriers from a lack of standardization. Preparing a dataset for a simple baseline requires substantial, error-prone preprocessing that is frequently duplicated across research groups, diverting time from novel development. The absence of a standardized preparation process inhibits fair benchmarking and reduces scientific rigor, as variations in filtering and splitting create incompatible evaluations even with the same nominal dataset [9,7]. Missing data provenance on source and transformations compromises reproducibility and ethical compliance, and the absence of cross-framework export mechanisms forces error-prone reimplementation of pipelines to compare models from different toolkits [20].

In this paper, we present OmniRec, a novel open-source Python library designed to address these issues through a comprehensive and robust all-in-one approach. OmniRec provides a unified interface for loading more than 230 datasets, coupled with a simple preprocessing API for cleaning and transformation. A key advantage is that this preprocessing pipeline is defined only once and can then be seamlessly executed across all integrated frameworks, eliminating redundant effort and ensuring consistency. To ensure reliability, its core design leverages modern Python typing to create strict interfaces, enabling developers to catch errors during code development and avoid costly, late-stage runtime crashes.

A cornerstone of OmniRec's interoperability is its seamless integration with multiple state-of-the-art recommendation frameworks. Our initial release integrates RecPack, RecBole, Lenskit, and Elliot. We developed custom API adapters for each framework, which we coupled with an automated environment management system that creates and maintains isolated virtual Python environments for each library. Thereby, OmniRec elegantly solves the critical problem of dependency conflicts between research frameworks.

OmniRec emphasizes extensibility, allowing developers to add new datasets, preprocessing operations, external frameworks, and evaluation metrics with minimal effort. For example, integrating a new recommender framework requires only a few hours of work implementing OmniRec's accessible interfaces. In terms of maintenance, framework updates that preserve a framework's API are trivial, i.e., a version bump, while those with breaking API changes require effort similar to adding a new framework. Using OmniRec, all processed datasets are fully documented, with complete provenance and exportable in common formats, enabling fair benchmarking and cross-framework experimentation. By addressing robustness, dependency management, and interoperability, OmniRec lowers the barrier to entry, safeguards reproducibility, and fosters a transparent research ecosystem. Unlike other frameworks, it provides a truly end-to-end, multi-framework workflow through a flexible, unified pipeline.

2 OmniRec

OmniRec is an open-source project, with source code on GitHub[4], comprehensive documentation[5], and a live demonstration[6]. Figure 1 illustrates the main components of OmniRec, which are organized into four interconnected modules: **Data Loader**, **Preprocessing Pipeline**, **Recommender Interface**, and **Evaluator**. This architecture enables a flexible, end-to-end workflow that spans from dataset loading to model evaluation.

The **Data Loader** loads registered datasets via specialized interfaces. It includes preprocessing operations, i.e., optional removal of duplicate user-item interactions and normalization of identifiers to incrementing integers, to ensure consistency across datasets. In addition, it exposes dataset statistics, which can be used for both exploratory analysis and reproducibility reporting.

The **Preprocessing Pipeline** applies user-specified preprocessing steps. Currently supported operations include subsampling, filtering by time or rating, core pruning, feedback conversion, and several splitting strategies, such as time-based holdout, random holdout, user-based holdout, and user-based cross-validation. The pipeline design is extensible, allowing developers to add new preprocessing functions by implementing a pre-defined interface.

The **Recommender Interface** enables seamless export of preprocessed datasets to the widely used recommender systems frameworks Lenskit, RecPack, RecBole, and Elliot. Through this interface, users can select algorithms, fit models, and generate predictions within the target framework, all while maintaining a single, unified preprocessing pipeline.

The **Evaluator** module provides a standardized interface for assessing model performance across different frameworks and datasets. It supports the computation of ranking- and rating-based metrics, ensuring that evaluation metrics are

[4] http://code.isg.beel.org/OmniRec
[5] https://omnirec.recommender-systems.com/
[6] https://youtu.be/fr4Gxo0sTwE

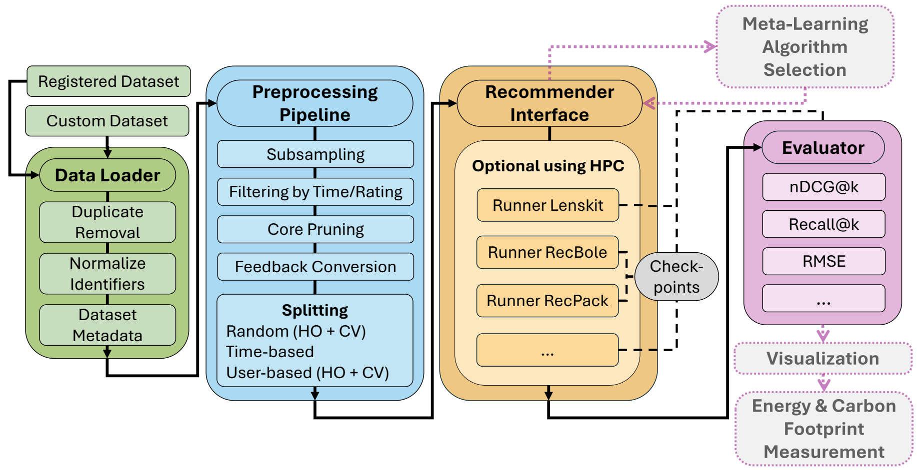

Fig. 1. Diagram of the OmniRec architecture, depicting four main components with their respective features and the relationships between them. Semi-transparent grey boxes denote features planned for future integration.

directly comparable regardless of the underlying framework. By centralizing evaluation logic, the module eliminates discrepancies caused by framework-specific metric implementations and promotes transparent, reproducible reporting. Evaluation outputs can be stored in addition to the dataset and preprocessing metadata, enabling complete experiment traceability and facilitating long-term reproducibility in recommender systems research.

2.1 Evolution and Future Plans

OmniRec is the product of a multi-year development and validation process within our lab[7]. It has served as the experimental basis for our research published in the top recommender systems venues, including ACM RecSys, ECIR [23,4,26,25], and more [27,3,24]. OmniRec has also demonstrated its accessibility through its adoption in numerous undergraduate and graduate theses.

This work accompanies its inaugural public release. Designed for extensibility and growth, OmniRec will continue to evolve according to the following post-release roadmap, which is also illustrated in Figure 1. Our immediate priorities include: (1) adding a visualization component to simplify the creation of plots for datasets and evaluations; (2) introducing an algorithm selection module that suggests suitable models based on dataset characteristics [25], and (3) incorporating energy consumption monitoring to provide insights into the computational cost and environmental impact of experiments [27].

[7] https://isg.beel.org/

References

1. Anand, R., Beel, J.: Auto-surprise: An automated recommender-system (autorecsys) library with tree of parzens estimator (tpe) optimization. In: Proceedings of the 14th ACM Conference on Recommender Systems. pp. 585–587 (2020)
2. Anelli, V.W., Bellogín, A., Ferrara, A., Malitesta, D., Merra, F.A., Pomo, C., Donini, F.M., Di Noia, T.: Elliot: A comprehensive and rigorous framework for reproducible recommender systems evaluation. In: Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval. pp. 2405–2414 (2021)
3. Baumgart, M., Wegmeth, L., Vente, T., Beel, J.: e-fold cross-validation for recommender-system evaluation. In: Boratto, L., De Filippo, A., Lex, E., Ricci, F. (eds.) Recommender Systems for Sustainability and Social Good. pp. 90–97. Springer Nature Switzerland, Cham (2025)
4. Beel, J., Wegmeth, L., Michiels, L., Schulz, S.: Informed dataset selection with 'algorithm performance spaces'. In: Proceedings of the 18th ACM Conference on Recommender Systems. p. 1085–1090. RecSys '24, Association for Computing Machinery, New York, NY, USA (2024). https://doi.org/10.1145/3640457.3691704, https://doi.org/10.1145/3640457.3691704
5. Chia, P.J., Tagliabue, J., Bianchi, F., He, C., Ko, B.: Beyond ndcg: behavioral testing of recommender systems with reclist. In: Companion Proceedings of the Web Conference 2022. pp. 99–104 (2022)
6. Ekstrand, M.D.: Lenskit for python: Next-generation software for recommender systems experiments. In: Proceedings of the 29th ACM international conference on information & knowledge management. pp. 2999–3006 (2020)
7. Ferrari Dacrema, M., Boglio, S., Cremonesi, P., Jannach, D.: A troubling analysis of reproducibility and progress in recommender systems research. ACM Trans. Inf. Syst. **39**(2) (Jan 2021). https://doi.org/10.1145/3434185, https://doi.org/10.1145/3434185
8. Graham, S., Min, J.K., Wu, T.: Microsoft recommenders: tools to accelerate developing recommender systems. In: Proceedings of the 13th ACM Conference on Recommender Systems. pp. 542–543 (2019)
9. Hidasi, B., Czapp, Á.T.: The effect of third party implementations on reproducibility. In: Proceedings of the 17th ACM Conference on Recommender Systems. pp. 272–282 (2023)
10. Hug, N.: Surprise: A python library for recommender systems. Journal of Open Source Software **5**(52), 2174 (2020)
11. Ivchenko, D., Van Der Staay, D., Taylor, C., Liu, X., Feng, W., Kindi, R., Sudarshan, A., Sefati, S.: Torchrec: a pytorch domain library for recommendation systems. In: Proceedings of the 16th ACM Conference on Recommender Systems. p. 482–483. RecSys '22, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3523227.3547387, https://doi.org/10.1145/3523227.3547387
12. Kula, M.: Metadata embeddings for user and item cold-start recommendations. In: Bogers, T., Koolen, M. (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. CEUR Workshop Proceedings, vol. 1448, pp. 14–21. CEUR-WS.org (2015), http://ceur-ws.org/Vol-1448/paper4.pdf
13. Kula, M.: Spotlight. https://github.com/maciejkula/spotlight (2017)

14. Li, J., Li, H., He, Z., Ma, W., Sun, P., Zhang, M., Ma, S.: Rechorus2. 0: A modular and task-flexible recommendation library. In: Proceedings of the 18th ACM Conference on Recommender Systems. pp. 454–464 (2024)
15. Lian, D., Huang, X., Chen, X., Chen, J., Wang, X., Wang, Y., Jin, H., Fan, R., Liu, Z., Wu, L., et al.: Recstudio: Towards a highly-modularized recommender system. In: Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 2890–2900 (2023)
16. Lops, P., Polignano, M., Musto, C., Silletti, A., Semeraro, G.: Clayrs: An end-to-end framework for reproducible knowledge-aware recommender systems. Information Systems **119**, 102273 (2023)
17. Mancino, A.C.M., Bufi, S., Di Fazio, A., Ferrara, A., Malitesta, D., Pomo, C., Di Noia, T.: Datarec: A python library for standardized and reproducible data management in recommender systems. In: Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 3478–3487 (2025)
18. Michiels, L., Verachtert, R., Goethals, B.: Recpack: An (other) experimentation toolkit for top-n recommendation using implicit feedback data. In: Proceedings of the 16th ACM Conference on Recommender Systems. pp. 648–651 (2022)
19. Salah, A., Truong, Q.T., Lauw, H.W.: Cornac: A comparative framework for multimodal recommender systems. Journal of Machine Learning Research **21**(95), 1–5 (2020), http://jmlr.org/papers/v21/19-805.html
20. Schmidt, M., Nitschke, J., Prinz, T.: Evaluating the performance-deviation of itemknn in recbole and lenskit (2024), https://arxiv.org/abs/2407.13531
21. Sun, Z., Fang, H., Yang, J., Qu, X., Liu, H., Yu, D., Ong, Y.S., Zhang, J.: Daisyrec 2.0: Benchmarking recommendation for rigorous evaluation. IEEE Transactions on Pattern Analysis and Machine Intelligence **45**(7), 8206–8226 (2022)
22. Vente, T., Ekstrand, M., Beel, J.: Introducing lenskit-auto, an experimental automated recommender system (autorecsys) toolkit. In: Proceedings of the 17th ACM Conference on Recommender Systems. pp. 1212–1216 (2023)
23. Vente, T., Wegmeth, L., Said, A., Beel, J.: From clicks to carbon: The environmental toll of recommender systems. In: Proceedings of the 18th ACM Conference on Recommender Systems. p. 580–590. RecSys '24, Association for Computing Machinery, New York, NY, USA (2024). https://doi.org/10.1145/3640457.3688074, https://doi.org/10.1145/3640457.3688074
24. Wegmeth, L., Beel, J.: CaMeLS: Cooperative meta-learning service for recommender systems. In: Proceedings of the Perspectives on the Evaluation of Recommender Systems Workshop 2022. CEUR-WS (Sep 2022), https://ceur-ws.org/Vol-3228/paper2.pdf
25. Wegmeth, L., Vente, T., Beel, J.: Recommender systems algorithm selection for ranking prediction on implicit feedback datasets. In: Proceedings of the 18th ACM Conference on Recommender Systems. p. 1163–1167. RecSys '24, Association for Computing Machinery, New York, NY, USA (2024). https://doi.org/10.1145/3640457.3691718, https://doi.org/10.1145/3640457.3691718
26. Wegmeth, L., Vente, T., Purucker, L.: Revealing the hidden impact of top-n metrics on optimization in recommender systems. In: Advances in Information Retrieval: 46th European Conference on Information Retrieval, ECIR 2024, Glasgow, UK, March 24–28, 2024, Proceedings, Part I. p. 140–156. Springer-Verlag, Berlin, Heidelberg (2024). https://doi.org/10.1007/978-3-031-56027-9_9, https://doi.org/10.1007/978-3-031-56027-9_9

27. Wegmeth, L., Vente, T., Said, A., Beel, J.: Emers: Energy meter for recommender systems. In: Boratto, L., De Filippo, A., Lex, E., Ricci, F. (eds.) Recommender Systems for Sustainability and Social Good. pp. 83–89. Springer Nature Switzerland, Cham (2025)
28. Zangerle, E., Bauer, C.: Evaluating recommender systems: survey and framework. ACM computing surveys **55**(8), 1–38 (2022)
29. Zhao, W.X., Mu, S., Hou, Y., Lin, Z., Chen, Y., Pan, X., Li, K., Lu, Y., Wang, H., Tian, C., et al.: Recbole: Towards a unified, comprehensive and efficient framework for recommendation algorithms. In: proceedings of the 30th acm international conference on information & knowledge management. pp. 4653–4664 (2021)