Beyond NDCG: behavioral testing of recommender systems with RecList

Patrick John Chia*
Coveo
Canada
pchia@coveo.com

Jacopo Tagliabue
Coveo Labs
United States
jtagliabue@coveo.com

Federico Bianchi
Bocconi University
Italy
f.bianchi@unibocconi.it

Chloe He
Stanford University
United States
chloehe@stanford.edu

Brian Ko
KOSA AI
United States
sangwoo@kosa.ai

ABSTRACT

As with most Machine Learning systems, recommender systems are typically evaluated through performance metrics computed over held-out data points. However, real-world behavior is undoubtedly nuanced: *ad hoc* error analysis and deployment-specific tests must be employed to ensure the desired quality in actual deployments. In this paper, we propose `RecList`, a behavioral-based testing methodology. `RecList` organizes recommender systems by use case and introduces a general plug-and-play procedure to scale up behavioral testing. We demonstrate its capabilities by analyzing known algorithms and black-box commercial systems, and we release `RecList` as an open source, extensible package for the community.

CCS CONCEPTS

• **Software and its engineering** → *Acceptance testing*; • **Information systems** → **Recommender systems**.

KEYWORDS

recommender systems, behavioral testing, open source

ACM Reference Format:
Patrick John Chia, Jacopo Tagliabue, Federico Bianchi, Chloe He, and Brian Ko. 2018. Beyond NDCG: behavioral testing of recommender systems with RecList. In *Lion '22: The Web Conference, June 03–05, 2022, Lion, France.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

> "A QA engineer walks into a bar. Orders a beer. Orders 0 beers. Orders 99999999999 beers. Orders a lizard. Orders -1 beers. Orders a ueicbksjdhd." – B. Keller (random tweet).

In recent years, recommender systems (hence **RSs**) have played an indispensable role in providing personalized digital experiences to users, by fighting information overload and helping with navigating inventories often made of millions of items [4, 8, 25, 35, 38]. RSs' ability to generalize, both in industry and academia, is often evaluated through some accuracy score over a held-out dataset: however, performance given by a single score often fails to give developers and stakeholders a rounded view of the expected performances of the system "in the wild". For example, as industry seems to recognize more than academia, not all inputs are created equal, and not all mistakes are uniformly costly; while these considerations are crucial to real-world success, reporting NDCG alone fails to capture these nuances. This is particularly important in the world of RSs, given both the growing market for RSs[1] and the role of RSs in shaping (often, narrowing [1]) user preferences with potential harmful consequences [15].

Following the lead of [27] in Natural Language Processing, we propose a behavioral-based framework to test RSs across a variety of industries, focusing on the peculiarities of horizontal use cases (e.g. substitute vs complementary items) more than vertical domains. We summarize our main contributions as follows:

- we argue for the importance of a well-rounded and more nuanced evaluation of RSs and discuss the importance of scaling up testing effort through automation;
- we release an open-source package to the community – `RecList`. `RecList` comes with ready-made behavioral tests and connectors for important public datasets (*Coveo Data Challenge* [31], *MovieLens* [13], *Spotify* [39]) and an extensible interface for custom use cases;
- we demonstrate our methodology by analyzing standard models and SaaS offerings over a cart recommendation task.

While we developed `RecList` out of the very practical necessities involved in scaling RSs to hundreds of organizations across many industries[2], as researchers, we also believe this methodology to be widely applicable in error analysis and thorough evaluation of new models: as much as we like to read about a new SOTA score on *MovieLens*, we would also like to understand what that score tells us about the capabilities and shortcomings of the model.

*Patrick, Jacopo and Federico originally conceived and designed `RecList` together, and they contributed equally to the paper. Chloe and Brian added important capabilities to the package, and greatly helped in improving the paper as well.

[1] E-commerce alone – arguably the biggest market for recommendations – is estimated to turn into a > 4 trillion industry by the end of 2021 [29].

[2] *Coveo* is a *multi-tenant* provider of A.I. services, with a network of hundreds of deployments for customer service, e-commerce and enterprise search use cases.

Lion '22, June 03–05, 2022, Lion, France

ACM ISBN 978-1-4503-XXXX-X/22/06...$15.00
https://doi.org/10.1145/1122445.1122456

2 AN INDUSTRY PERSPECTIVE ON RECSYS

While quantitative metrics over standardized datasets are indispensable in providing an objective pulse on where the field is going, our experiences both as researchers and practitioners are that NDCG tells only one part of the performance story. As a very concrete example, while model performance depends mostly on what happens with frequent items, the final user experience may be ruined by poor outcomes in the long-tail [2]. Metrics such as coverage, serendipity, and bias [16, 18, 22] have been therefore proposed to capture other aspects of the behaviors of RSs, but they still fall short of what is needed to debug RSs in production, or provide any guarantee that a model will be reliable when released.

When developing `RecList`, we started from popular use cases that represent the most widely adopted strategies for recommendation systems:

(1) **similar items**: when shown running shoes, users may want to browse for another pair of running shoes – in other words, they are looking for substitutable products. This type of recommendation is also a common pattern in entertainment [21, 25], to suggest content similar to a previous or current viewing, and in comparison recommenders [8];
(2) **complementary items**: when a TV has been added to the cart, shoppers may want to buy complementary products such as an HDMI cable. This type of recommendation is more typical of e-commerce scenarios and exhibits a characteristic asymmetry (Figure 1);
(3) **session-based recommendations**: real-time behavior has been recently exploited to provided session-based personalization [6, 12, 14, 36], which captures both preferences from recent sessions and real-time intent; a typical session-based RS ingests the latest item interactions for a user and predicts the probable next interaction(s).

From these use cases, we identified three main areas of behavioral intervention:

(1) **enforce per-task invariants**: irrespective of the target deployment, complementary and similar items satisfy formal relations which are different in nature. In particular, similar items need to be interchangeable, while complementary items may have a natural ordering. We operationalize these insights by joining predictions with metadata: for example, we can use price information to check for asymmetry constraints;
(2) **being less wrong**: if the ground truth item for a movie recommendation is "When Harry met Sally", hit-or-miss metrics won't be able to distinguish between model A that predicts "Terminator" and model B that suggests "You've got mail"[3]. In other words, while both are "wrong", they are not wrong in the same way: one is a reasonable mistake, the other is a terrible suggestion. RSs are a major factor in boosting user experience (which translates to revenues, loyalty etc.): in a recent survey, 38% of shoppers said they would stop shopping at a retailer showing non-relevant recommendations [20];

[3]In case the reader is too young to know better, model A is way worse than model B.

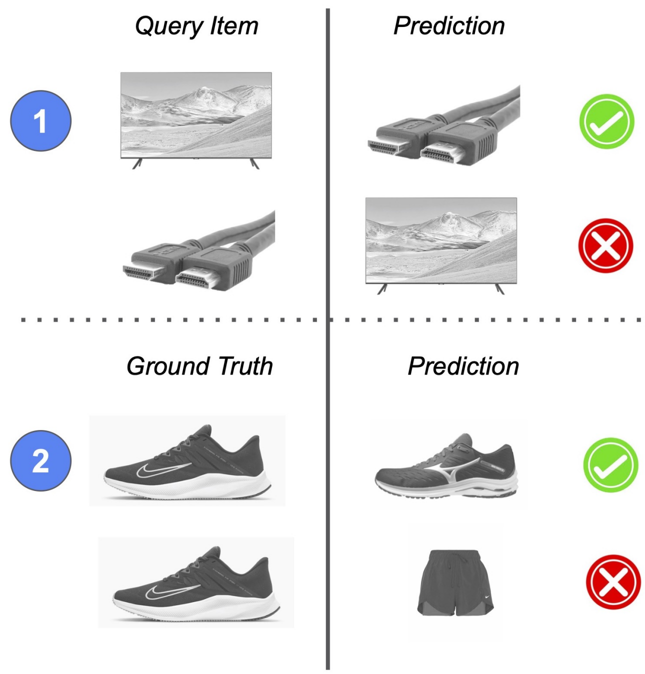

Figure 1: Examples of behavioral principles for RSs: in (1) we observe the asymmetry desired when recommending complementary items, while (2) exemplifies that model mistakes (i.e. missing the ground truth item) may degrade the shopping experience in different ways.

(3) **data slices**: in real-world RSs, not all inputs are created equal. In particular, we may tolerate a small decrease in overall accuracy if a subset of users we care about is happier; or, conversely, we may want to improve general performance *provided that* the experiences of some groups do not degrade too much. For a practical example, consider a multi-brand retailer promoting the latest Nike shoes with a marketing campaign: other things being equal, this retailer would want to make sure the experiences of users landing on Nike product pages are particularly well curated. Aside from horizontal cases (e.g. *cold-start* items), the most interesting slices are often context-dependent, which is an important guiding principle for our library.

Figure 1 exemplifies these concepts in an e-commerce setting. Building `RecList` requires to both operationalize behavioral principles in code whenever possible, and provide an extensible interface when domain knowledge and custom logic are required (Section 4).

3 RELATED WORK

This work sits at the intersection of several themes in the research and industrial communities. We were initially inspired by behavioral testing for NLP pioneered by [27]: from this seminal work we took two lessons: *first*, that black-box testing [3] is a source of great insights in addition to standard quantitative metrics; *second*, that this methodology goes hand-in-hand with software tools, as creating, maintaining, and analyzing behavioral tests by manual curation is a very time-consuming process. On the other hand, `RecList` needs to consider the peculiarities of RSs, as compared to NLP: in particular, the concept of generic large-scale model does not apply to RSs, which are deployed in different shops, with a specific target distribution: the same pair of running shoes can be popular in *Shop X* and not *Shop Y*, and categorized as *sneakers* in one case, *running shoes* in the other.

From the A/B testing literature [17], we take the important lesson that not all test cases are created equal: in particular, just as a careful A/B test cares both about the aggregate effect of treatment and the individual effects on specific data slices, a careful set of RS testing should worry about the overall accuracy as well as the accuracy in specific subgroup-of-interests: in ML systems, as in life, gains and losses are not always immediately interchangeable.

The RS literature exploited already insights contained in `RecList`, typically as part of error analysis [28], or as performance boost for specific datasets [11]. For example, "being less wrong" is discussed in [33], while cold start performance is often highlighted for methods exploiting content-based features [34]. Our work builds on top of this scattered evidence, and aims to be the one-stop shop for behavioral analysis of RSs: `RecList` provides practitioners with both a common lexicon and working code for scalable, in-depth error analysis.

Finally, as far as standard quantitative metrics go, the literature is pretty consistent: a quick scan through recent editions of RecSys and SIGIR highlights the use of *MRR*, *ACCURACY*, *HITS*, *NDCG* as the main metrics [7, 19, 24, 26, 37]. To ease the comparison with research papers on standard KPIs, we made sure that these metrics are computed by `RecList` as well, together with behavioral results.

4 RECLIST (A.K.A. `CHECKLIST` FOR RECS)

`RecList` is behavioral testing applied to RSs, and available as a plug-and-play open-source package that can be easily extended to proprietary datasets and models. Following [27], we decouple testing from implementation: our framework treats RSs as a black box (through an extensible programming interface), allowing us to test RSs for which no source code is available (e.g. SaaS models). To strengthen our exposition of the methodology, we offer here a high-level view of the logical architecture and capabilities of `RecList` as a package. However, please note the code is actively evolving as a community project: the reader is encouraged to check out our repository[4] for up-to-date documentation and examples.

4.1 Abstractions

`RecList` is a Python package built over these main abstractions:

- *RecTask*: the recommendation use case (Section 2).
- *RecModel*: the model we are testing – as long as a simple prediction-based interface can be implemented, any model can be represented in `RecList`. For example, a SaaS model would make an API call to a service and let `RecList` handle the analysis.
- *RecDataset*: the dataset we are using – the class provides standard access to train/test splits *and* item metadata. `RecList` comes with ready-made connectors for popular datasets.
- *RecList*[5]: the actual set of tests we are running, given a *RecTask*, *RecModel* and *RecDataset*. A *RecList* is made of *RecTests*.

When running a *RecList*, the package automatically versions the relevant metadata: a web application is provided to analyze test reports, and visually compare the performance of different models (Appendix A).

4.2 Capabilities

While we refer readers to our repository for an up-to-date list of available *RecLists*, *RecModels* and *RecDatasets*, we wish to highlight some key capabilities:

- **leveraging representation learning**: word embeddings for behavioral testing in NLP are replaced by representational learning *per dataset*. By unifying access to items and metadata (for example, *brands* for products, *labels* for music), `RecList` provides a scalable, unsupervised flow to obtain latent representation of target entities [23], and uses them to generate new test pairs, or supply similarity judgment when needed (Figure 2).
- **merging metadata and predictions**: `RecList`'s tests provide a functional interface that can be applied to any dataset by supplying the corresponding entities. For example, asymmetry tests can be applied to any feature exhibiting the desired behavior (e.g. *price* for complementary items); in the same vein, data slices can be specified with arbitrary partitioning functions, allowing seamless reporting on important subsets;
- **injecting domain knowledge when needed**: `RecList` allows to easily swap default similarity metrics with custom ones (or, of course, write entirely new tests): for example, a very accurate taxonomy could be used to define a new distance between predictions and labels, supplementing out-of-the-box unsupervised similarity metrics.

Table 1: Results for a complementary *RecList*.

Test	P2V	GOO	S1
HR@10	0.197	**0.199**	0.939
MRR@10	0.0913	**0.102**	0.0692
Coverage@10	1.01e-2	**1.99-e2**	3.00e-3
Popularity Bias@10	**9.91e-5**	1.41e-4	1.20e-4
Cos Distance (Brand)	**0.411**	0.483	0.540
Cos Distance (Misses)	0.564	**0.537**	0.577
Path Length (Category)	1.13	1.59	**1.91**

5 A WORKED-OUT EXAMPLE: CART RECS

To showcase `RecList` in a real-world setting, we test three RSs on a complementary items task: a *prod2vec*-based recommender [5] (hence **P2V**); Google Recommendation APIs (**GOO**) [9]; and one

[4] https://github.com/jacopotagliabue/reclist

[5] Note that we use *RecList* to indicate the class or its instances, and `RecList` to indicate the package as a whole.

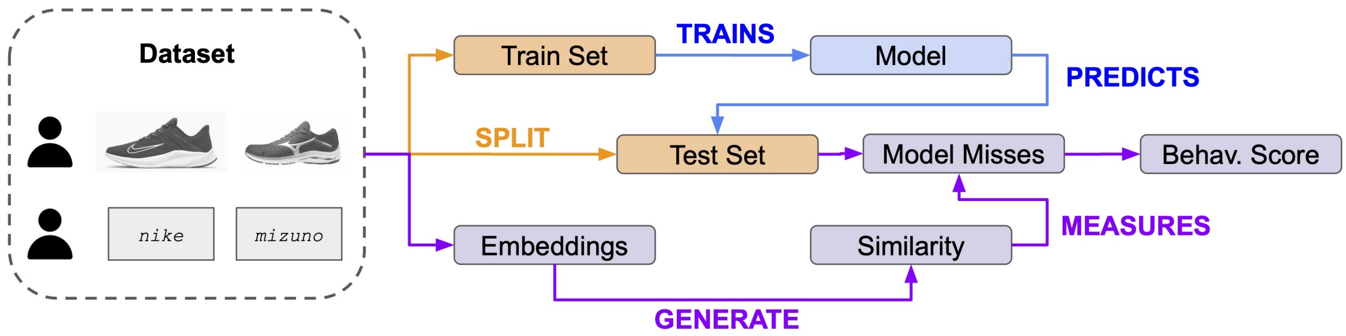

Figure 2: Sample workflow for behavioral tests. Starting with shopping data (left), the dataset split (orange) and model training (blue) mimic the usual training loop. `RecList` creates a latent space, which it uses to measure the relationships between inputs, ground truths and predictions, such as how far misses are from ground truths (violet). Since a session can be viewed as a sequence of items or features (brands), `RecList` can use the same method to create embeddings for different tests.

popular SaaS model (**S1**)[6]. We use data from a "reasonable scale" [30] e-commerce in the sport apparel industry, where 1M product interactions have been sampled for training from a period of 3 months in 2019, and 2.5K samples from a disjoint time period for testing[7]. We first assess the models with various standard aggregate metrics (Table 1): based on HR@10 and MRR@10, **GOO** and **P2V** are close and they outperform **S1**. For reason of space, we discuss our insights from three *RecTests*:

- **Product Popularity**: we compare the distribution of hits across product click-frequency (i.e. how accurate the prediction is, conditional on its target being very / mildly / poorly popular). **P2V** can be seen to perform better on rare items (clicked ~ 10^2 times) by 40% over **GOO**. On the other hand **GOO** outperforms **P2V** by 200% on the most frequently-viewed items (clicked ~ 10^5 times).
- **"Being Less Wrong"**: we compute the distance between input product to ground truth, and input product to prediction *for missed predictions*: cosine-distance over a *prod2vec* space is our distance measure (Figure 3). We observe that despite having equivalent HR@10/MRR@10 as **P2V**, **GOO**'s prediction distribution better matches the label distribution, suggesting that its predictions are more aligned to the complementary nature of the cart recommendation task. This highlights another difference between **GOO** and **P2V** which HR/MRR alone were unable to capture[8].
- **Slice-by-Brand**: we measure hits across various brands. We find that while **P2V** and **GOO** have very similar performance, **P2V** is particularly performant on `asics`, compensating for a slightly lower result on `nike`: without behavioral testing, this bias in **P2V** would have been hard or time-consuming to catch.

[6]Due to monetary and legal limitations, a perfect comparison on our private dataset was impossible. The goal of this analysis is *not* to rank these models, but to demonstrate how the methodology provides insights about their behavior.
[7]Performance on hashed, public datasets can be obtained with the package: we used private data here to develop our intuition on behavioral tests.
[8]Qualitative checks confirmed that P2V often predicted products from the same category as input, whereas GOO exhibited greater prediction variety

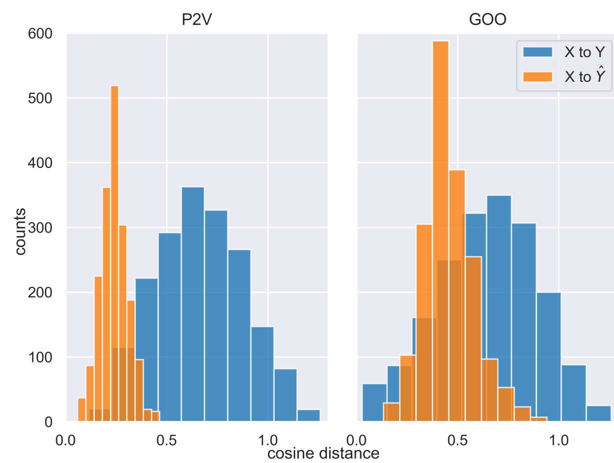

Figure 3: Distribution of cosine distances for input to label (X to Y) and input to prediction (X to $\hat{Y}$).

The lower half of Table 1 contains aggregate results for other *RecTests*. We show other ways "Being Less Wrong" can be operationalized to demonstrate the flexibility of `RecList`. *Cos Distance (Brand)* trains brand embeddings and measures the distance between label and prediction in this space for misses. "Less Wrong" in this case might mean that presenting an Adidas product over Lacoste if a Nike product is in the basket. Similarly, *Cos Distance (Misses)* measures the same distance but over a prod2vec space instead. Conversely, *Path Length* goes for a discrete approach and measures distance as the path length between input and prediction based on a product category tree (longer suggests greater diversity).

6 CONCLUSION

We introduced `RecList`, a package for behavioral testing in recommender systems: `RecList` *alpha* version already supports popular datasets and plug-and-play tests for common use cases. However, behavioral testing needs to continuously evolve as our understanding of RSs improves, and their limitations, capabilities and reach change: by open sourcing `RecList` we hope to help the field go beyond "leaderboard chasing", and to empower practitioners with better tools for analysis, debugging, and decision-making.

ACKNOWLEDGMENTS

Authors wish to thank Andrea Polonioli for feedback on previous drafts of this work and Jean-Francis Roy for his constant support in this project (and many others as well). Finally, it is worth mentioning that this is our first community-sourced (5 cities, 4 time zones) scholarly work, with Chloe and Brian joining the *Coveo* line-up through a thread on Discord (thanks Chip Huyen for creating that amazing place!). While it is too early to say how successful this model will be for a company of our size, we are proud of what we achieved so far.

REFERENCES

[1] Panagiotis Adamopoulos and Alexander Tuzhilin. 2014. On Over-Specialization and Concentration Bias of Recommendations: Probabilistic Neighborhood Selection in Collaborative Filtering Systems. In *Proceedings of the 8th ACM Conference on Recommender Systems* (Foster City, Silicon Valley, California, USA) *(RecSys '14)*. Association for Computing Machinery, New York, NY, USA, 153–160. https://doi.org/10.1145/2645710.2645752

[2] Nidhi Arora, Daniel Ensslen, Lars Fiedler, Wei Wei Liu, Kelsey Robinson, Eli Stein, and Gustavo Schüler. 2021. *The value of getting personalization right—or wrong—is multiplying.* Retrieved November 15, 2021 from https://www.mckinsey.com/business-functions/marketing-and-sales/our-insights/the-value-of-getting-personalization-right-or-wrong-is-multiplying

[3] B. Beizer and J. Wiley. 1996. Black Box Testing: Techniques for Functional Testing of Software and Systems. *IEEE Software* 13, 5 (1996), 98–. https://doi.org/10.1109/MS.1996.536464

[4] Rahul Bhagat, Srevatsan Muralidharan, Alex Lobzhanidze, and Shankar Vishwanath. 2018. Buy It Again: Modeling Repeat Purchase Recommendations. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (London, United Kingdom) *(KDD '18)*. Association for Computing Machinery, New York, NY, USA, 62–70. https://doi.org/10.1145/3219819.3219891

[5] Federico Bianchi, J. Tagliabue, Bingqing Yu, Luca Bigon, and Ciro Greco. 2020. Fantastic Embeddings and How to Align Them: Zero-Shot Inference in a Multi-Shop Scenario. *ArXiv* abs/2007.14906 (2020).

[6] Federico Bianchi, Bingqing Yu, and Jacopo Tagliabue. 2021. BERT Goes Shopping: Comparing Distributional Models for Product Representations. In *Proceedings of The 4th Workshop on e-Commerce and NLP*. Association for Computational Linguistics, Online, 1–12. https://doi.org/10.18653/v1/2021.ecnlp-1.1

[7] Renqin Cai, Jibang Wu, Aidan San, Chong Wang, and Hongning Wang. 2021. Category-Aware Collaborative Sequential Recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, Canada) *(SIGIR '21)*. Association for Computing Machinery, New York, NY, USA, 388–397. https://doi.org/10.1145/3404835.3462832

[8] Patrick John Chia, Bingqing Yu, and Jacopo Tagliabue. 2021. "Are you sure?": Preliminary Insights from Scaling Product Comparisons to Multiple Shops. *ArXiv* abs/2107.03256 (2021).

[9] Google Cloud. 2021. *Implementing Recommendations AI.* Retrieved November 17, 2021 from https://cloud.google.com/retail/recommendations-ai/docs/overview

[10] dbt Labs. 2021. *dbt docs.* Retrieved November 14, 2021 from https://docs.getdbt.com/reference/commands/cmd-docs

[11] Gabriel de Souza Pereira Moreira, Sara Rabhi, Ronay Ak, Md Yasin Kabir, and Even Oldridge. 2021. Transformers with multi-modal features and post-fusion context for e-commerce session-based recommendation. *ArXiv* abs/2107.05124 (2021).

[12] Lei Guo, Hongzhi Yin, Qinyong Wang, Tong Chen, Alexander Zhou, and Nguyen Quoc Viet Hung. 2019. Streaming Session-Based Recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Anchorage, AK, USA) *(KDD '19)*. Association for Computing Machinery, New York, NY, USA, 1569–1577. https://doi.org/10.1145/3292500.3330839

[13] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.* 5, 4, Article 19 (Dec. 2015), 19 pages. https://doi.org/10.1145/2827872

[14] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks. *CoRR* abs/1511.06939 (2016).

[15] Kartik Hosanagar, Daniel Fleder, Dokyun Lee, and Andreas Buja. 2014. Will the Global Village Fracture Into Tribes? Recommender Systems and Their Effects on Consumer Fragmentation. *Management Science* 60 (04 2014), 805–823. https://doi.org/10.1287/mnsc.2013.1808

[16] Dietmar Jannach and Malte Ludewig. 2017. When recurrent neural networks meet the neighborhood for session-based recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*. 306–310.

[17] Ron Kohavi, Alex Deng, Brian Frasca, Roger Longbotham, Toby Walker, and Ya Xu. 2012. Trustworthy Online Controlled Experiments: Five Puzzling Outcomes Explained. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Beijing, China) *(KDD '12)*. Association for Computing Machinery, New York, NY, USA, 786–794. https://doi.org/10.1145/2339530.2339653

[18] Denis Kotkov, Jari Veijalainen, and Shuaiqiang Wang. 2016. Challenges of Serendipity in Recommender Systems. In *WEBIST*.

[19] Pigi Kouki, Ilias Fountalis, Nikolaos Vasiloglou, Xiquan Cui, Edo Liberty, and Khalifeh Al Jadda. 2020. From the Lab to Production: A Case Study of Session-Based Recommendations in the Home-Improvement Domain. In *Fourteenth ACM Conference on Recommender Systems* (Virtual Event, Brazil) *(RecSys '20)*. Association for Computing Machinery, New York, NY, USA, 140–149. https://doi.org/10.1145/3383313.3412235

[20] Krista Garcia. 2018. *The Impact of Product Recommendations.* Retrieved November 9, 2021 from https://www.emarketer.com/content/the-impact-of-product-recommendations

[21] Sudarshan Lamkhede and Christoph Kofler. 2021. Recommendations and Results Organization in Netflix Search. In *RecSys '21: Fifteenth ACM Conference on Recommender Systems*.

[22] Malte Ludewig and Dietmar Jannach. 2018. Evaluation of session-based recommendation algorithms. *User Modeling and User-Adapted Interaction* 28, 4-5 (2018), 331–390.

[23] Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *ICLR*.

[24] Théo Moins, Daniel Aloise, and Simon J. Blanchard. 2020. RecSeats: A Hybrid Convolutional Neural Network Choice Model for Seat Recommendations at Reserved Seating Venues. In *Fourteenth ACM Conference on Recommender Systems* (Virtual Event, Brazil) *(RecSys '20)*. Association for Computing Machinery, New York, NY, USA, 309–317. https://doi.org/10.1145/3383313.3412263

[25] Houssam Nassif, Kemal Oral Cansizlar, Mitchell Goodman, and SVN Vishwanathan. 2018. Diversifying Music Recommendations. In *ICML '16 Workshop*.

[26] Ahmed Rashed, Shayan Jawed, Lars Schmidt-Thieme, and Andre Hintsches. 2020. MultiRec: A Multi-Relational Approach for Unique Item Recommendation in Auction Systems. *Fourteenth ACM Conference on Recommender Systems* (2020).

[27] Marco Túlio Ribeiro, Tongshuang (Sherry) Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond Accuracy: Behavioral Testing of NLP Models with CheckList. In *ACL*.

[28] Mohammad Saberian and Justin Basilico. 2021. *RecSysOps: Best Practices for Operating a Large-Scale Recommender System.* Association for Computing Machinery, New York, NY, USA, 590–591. https://doi.org/10.1145/3460231.3474620

[29] Statista Research Department. 2020. *Global retail e-commerce sales 2014-2023.* Retrieved November 29, 2020 from https://www.statista.com/statistics/379046/worldwide-retail-e-commerce-sales/

[30] Jacopo Tagliabue. 2021. *You Do Not Need a Bigger Boat: Recommendations at Reasonable Scale in a (Mostly) Serverless and Open Stack.* Association for Computing Machinery, New York, NY, USA, 598–600. https://doi.org/10.1145/3460231.3474604

[31] Jacopo Tagliabue, Ciro Greco, Jean-Francis Roy, Federico Bianchi, Giovanni Cassani, Bingqing Yu, and Patrick John Chia. 2021. SIGIR 2021 E-Commerce Workshop Data Challenge. In *SIGIR eCom 2021*.

[32] Jacopo Tagliabue, Ville H. Tuulos, Ciro Greco, and Valay Dave. 2021. DAG Card is the new Model Card. *ArXiv* abs/2110.13601 (2021).

[33] Jacopo Tagliabue, Bingqing Yu, and Federico Bianchi. 2020. The Embeddings That Came in From the Cold: Improving Vectors for New and Rare Products with Content-Based Inference. In *Fourteenth ACM Conference on Recommender Systems* (Virtual Event, Brazil) *(RecSys '20)*. Association for Computing Machinery, New York, NY, USA, 577–578. https://doi.org/10.1145/3383313.3411477

[34] Flavian Vasile, Elena Smirnova, and Alexis Conneau. 2016. Meta-Prod2Vec: Product Embeddings Using Side-Information for Recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems* (Boston, Massachusetts, USA) *(RecSys '16)*. Association for Computing Machinery, New York, NY, USA, 225–232. https://doi.org/10.1145/2959100.2959160

[35] Menghan Wang, Yujie Lin, Guli Lin, Keping Yang, and Xiao-Ming Wu. 2020. M2GRL: A Multi-task Multi-view Graph Representation Learning Framework for Web-scale Recommender Systems. *CoRR* abs/2005.10110 (2020). arXiv:2005.10110 https://arxiv.org/abs/2005.10110

[36] Shoujin Wang, Longbing Cao, and Yan Wang. 2019. A Survey on Session-based Recommender Systems. *ACM Computing Surveys (CSUR)* 54 (2019), 1 – 38.

[37] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*. 165–174.

[38] Yixin Wang, Dawen Liang, Laurent Charlin, and David M. Blei. 2020. Causal Inference for Recommender Systems. In *recsys2020*.

[39] Hamed Zamani, Markus Schedl, Paul Lamere, and Ching-Wei Chen. 2019. An Analysis of Approaches Taken in the ACM RecSys Challenge 2018 for Automatic Music Playlist Continuation. *ACM Trans. Intell. Syst. Technol.* 10, 5, Article 57

(Sept. 2019), 21 pages. https://doi.org/10.1145/3344257

A TRACKING AND VISUALIZATION

Running a *RecList*, given a *RecDataset* and *RecModel*, automatically produces a report, versioned by the list name, the model name and the timestamp of the run. Running different models (or the same one, trained with different parameters) will therefore produce separate reports, which can be listed and inspected with a small web application bundled with the package (Figure 4).

Selecting multiple reports results in out-of-the-box comparative tables and charts; being a static HTML, the page can be easily shared among all stakeholders. Since reporting and visualization are completely decoupled, it is possible for practitioners to either extend the visualization to encompass new capabilities, or to send the metrics as a machine-friendly payload to downstream systems.

Please note that the *alpha* version of `RecList` re-uses styling and code from the original Dag Cards [32].

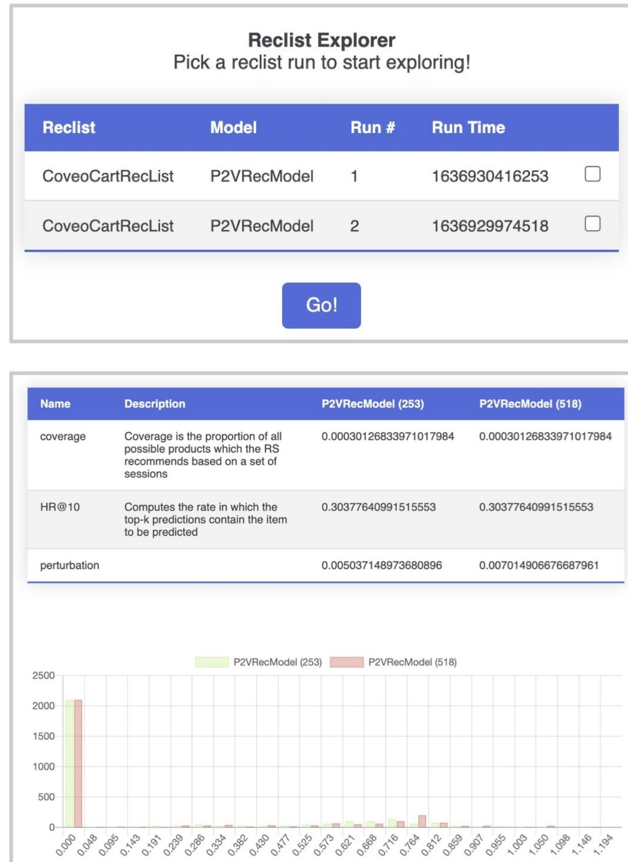

Figure 4: Built-in comparison and visualization app. Inspired by modern MLOps tools [10], `RecList` ships with a small web-app that prompts the user to select test runs (top), and then easily compare them through tables and distributions (bottom).