Estimating Error and Bias in Offline Evaluation Results

Mucun Tian
People & Information Research Team
Dept. of Computer Science, Boise State University
Boise, Idaho, USA
mucuntian@u.boisestate.edu

Michael D. Ekstrand
People & Information Research Team
Dept. of Computer Science, Boise State University
Boise, Idaho, USA
michaelekstrand@boisestate.edu

ABSTRACT

Offline evaluations of recommender systems attempt to estimate users' satisfaction with recommendations using static data from prior user interactions. These evaluations provide researchers and developers with first approximations of the likely performance of a new system and help weed out bad ideas before presenting them to users. However, offline evaluation cannot accurately assess *novel*, *relevant* recommendations, because the most novel items were previously unknown to the user, so they are missing from the historical data and cannot be judged as relevant.

We present a simulation study to estimate the error that such missing data causes in commonly-used evaluation metrics in order to assess its prevalence and impact. We find that missing data in the rating or observation process causes the evaluation protocol to systematically mis-estimate metric values, and in some cases erroneously determine that a popularity-based recommender outperforms even a perfect personalized recommender. Substantial breakthroughs in recommendation quality, therefore, will be difficult to assess with existing offline techniques.

CCS CONCEPTS

• **Information systems → Evaluation of retrieval results**.

KEYWORDS

simulation, offline evaluation

ACM Reference Format:
Mucun Tian and Michael D. Ekstrand. 2020. Estimating Error and Bias in Offline Evaluation Results. In *2020 Conference on Human Information Interaction and Retrieval (CHIIR '20), March 14–18, 2020, Vancouver, BC, Canada.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3343413.3378004

1 INTRODUCTION

Offline evaluation protocols for recommendation algorithms are designed to estimate how effective an algorithm is at delivering recommendations that users will find relevant. Significant work has gone in to refining experimental protocols and metrics to better measure recommender performance, but offline evaluation still faces a fundamental difficulty: most of the data we need for a robust evaluation — user response to various items — is missing. In particular, we only have data on items to which the user was somehow exposed, through an existing recommender system or other discovery mechanisms. Offline evaluation cannot accurately measure the effectiveness of truly novel recommendations: if a recommender algorithm reliably finds items the user has never heard of, but would enjoy, the evaluation protocol will either ignore those recommendations or judge them to be irrelevant.

The existence of this problem (and related problems with missing data in recommender evaluation) is well-documented [3, 6, 7, 10]. However, we do not yet understand the *impact* of this missing data: how frequently, and by how much, does it lead recommender system evaluations astray?

In this paper, we present a simulation study that estimates the extent of errors caused by missing relevance data. We simulate a data generation process consisting of complete preference data followed by observations (in the form of purchases or other consumption activities), and use these to compare evaluation results on observable data with results on complete data. One outcome of this process is an estimate of how the offline evaluation would treat a perfect recommender, implemented as an oracle with access to the true preference data. By varying the models and parameters in this process, we assess the sensitivity of our results to assumptions about preference and observation. We use public data sets in multiple domains to calibrate the simulation to produce realistic observation data. Our analysis addresses the following questions:

RQ1 Which models produce more realistic simulations?
RQ2 What error does missing data cause in evaluation metrics?
RQ3 How do assumptions about data affect metric error?
RQ4 What incorrect decisions does missing data cause?

Our here is to model the underlying process assumed by offline evaluation and estimate its error on its own terms. Other work will need to estimate errors resulting from those assumptions.

2 RELATED WORK

Several existing techniques attempt to measure and/or correct problems with offline evaluation. One approach is to *change the experimental protocol.* Bellogín [2] proposed data splitting and analysis strategies to address popularity bias; these methods affect absolute metric values, but not necessarily the relative performance of algorithms [3]. Using random subsets of the item space as candidates for recommendation may reduce the impact of unknown relevant items [7], but it relies on unrealistically strong assumptions and likely exacerbates popularity bias [10].

Another approach is to seek metrics that admit *statistically unbiased estimators* with observable data. If ratings for relevant items are missing at random, recall [19] and unnormalized DCG [17] are unbiased. But these results limit choice of metrics and depend on

CHIIR '20, March 14–18, 2020, Vancouver, BC, Canada

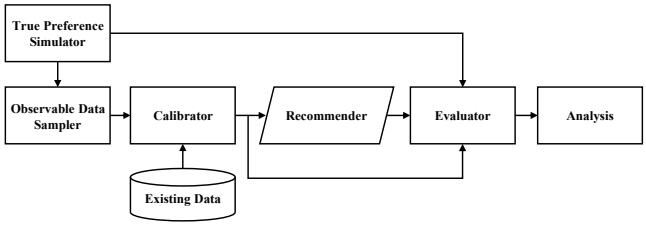

Figure 1: Simulation architecture

assumptions unlikely to hold in actual use, as relevance is not the only influence on users' choice of items to consume or rate.

Counterfactual evaluation [5, 11, 20] uses causal inference techniques — often inverse propensity scoring — to estimate how users would have responded to a different recommender algorithm. However, it is difficult to apply to commonly-used data sets and does not yield insight into the reliability of existing evaluations. It also cannot address the fundamental problem that concerns us in this work: if the user was never exposed to an item under the logging policy, the historical log data contains no information on its relevance. Such items are precisely where a recommender system can produce the most benefit in many discovery-oriented applications.

Simulation is a promising technique for studying evaluation procedures. Simulations can produce complete ground truth and corresponding observations in a controlled manner, subject to assumptions about the structure of the data generation process. Cañamares and Castells [6] used probabilistic models to better understand the impact of popularity bias, finding relationships between popularity bias and structural assumptions about the underlying data and inversions in the relative performance of collaborative filtering algorithms between complete and observable data in some cases.

3 SIMULATION METHODS

To estimate error distributions, we simulate the entire recommender system data collection and evaluation process. Figure 1 shows our simulation architecture; it models the user interaction that is fundamentally assumed by offline evaluation: users like items, and consume items that they (usually) like, producing records such as views or ratings that can be used to train and evaluate recommender systems. We begin by generating (binary) user-item relevance data, then sample purchase observations from this complete truth. We tune simulation parameters to mimic key statistics of existing public data sets, ensuring realism of simulated data.[1]

We then split simulated observations into training and test data, generate recommendations for the simulated users, and measure the quality of these recommendations using both the observed test data and the underlying true preferences as ground truth. This is the key capability our simulations enable: with access to ground truth data, because it is generated by the simulator, we can measure what the precision or reciprocal rank of a recommendation list would be if the data set were not missing data and compare it to the metric obtained from the observable data an experiment would ordinarily employ. We can therefore compute a first approximation of the error in experimental results where we cannot access unbiased truth. The simulated true preferences also enable us to test experimental protocols on an oracle recommender that omnisciently returns the most relevant items, irrespective of observation process.

3.1 True Preference Models

We use two models to simulate items being relevant to users. Both simulation strategies produce a set U of users and sets $\tilde{I}_u \subseteq I$ of items relevant to each user. We abbreviate here for space; the appendix documents full parameterizations. The first is preferential attachment, implemented as the three-parameter generalized Indian buffet process (IBP) [21]. This model is capable of producing data exhibiting power law behavior, unlike a traditional IBP. The IBP model assumes that users like items independently; the users who like item i are independent from those users who like the item j. This property allows us to scale up the simulation size through parallelism at the expense of realism.

The second is a correlated preference model, implemented as symmetric latent Dirichlet allocation (LDA) [4]. Exploiting correlations between items is fundamental to many recommendation techniques; latent feature models such as LDA provide a way to simulate such correlations.

3.2 Observation Process

We employ two models to produce a set of observable rated items $I_u \subseteq \tilde{I}_u$ for each user from their simulated preferences. Both models start with n_u, the number of items a user will consume; since each observable user rates at least one item (or some larger number, such as 20 for MovieLens data sets) and user activity levels follow a heavy-tailed distribution, we draw n_u from a truncated Pareto distribution rounded to an integer in the range $[1, |\tilde{I}_u|]$.[2]

Uniform samples n_u items uniformly at random from $\tilde{I}_u$. This implements the assumption that observations of relevant items are *missing-at-random* (MAR), so we can to compare simulation results with unbiased estimators relying on this assumption [19].

Popular operationalizes the idea that users are more likely to rate items that they are exposed to, and that they are more likely to be exposed to popular items. This is one way in which observed data may violate typical MAR assumptions. This strategy samples items with probability proportional to $|\tilde{U}_i|$ (where $\tilde{U}_i$ is the set of users who like item i in the true preference data).

3.3 Data Sets

We set the simulation parameters by comparing their output to three data sets from different domains, summarized in Table 2. ML1M [13] contains 1M ratings of 3,706 movies from 6,040 users, where each user has at least 20 ratings. AZM5 [14] contains 65K 5-star ratings of 3.6K digital music albums from 5.5K users; each user and item has at least 5 ratings (the "5-core"). STMV1 [18] contains 5M purchases of 11K video games by 71K Australian users of the Steam game distribution service. MovieLens and Amazon data sets are pruned [1], but the Steam data is — to our knowledge — unpruned and accurately represents user and item distributions.

[1] Full experiment code is in supplementary materials. In this exposition we use the notation proposed by Ekstrand and Konstan [9].

[2] We also tested the truncated beta-binomial distribution; its performance is similar to the truncated Pareto. Rejection-sampling when $n_u > |\tilde{I}_u|$ produced slightly better simulations than clamping at substantial computational expense.

Table 1: Simulation calibration performance, as $D_{\mathrm{KL}}(\mathrm{obs}\|\mathrm{sim})$ for each statistic, and relative loss.

Data	Model	I-I Sim		Item Pop		U-U Sim		User Act		Avg. Loss
ML1M	IBP-Unif	0.170	8400.00%	0.973	112.45%	0.227	3683.33%	0.373	42.37%	3059.54%
	IBP-Pop	0.200	9900.00%	1.793	291.48%	0.170	2733.33%	0.388	48.09%	3243.23%
	LDA-Unif	0.137	6750.00%	0.737	60.92%	0.082	1266.67%	0.669	155.34%	2058.23%
	LDA-Pop	0.050	2400.00%	2.063	350.44%	0.291	4750.00%	1.084	313.74%	**1953.54%**
AZM5	IBP-Unif	0.024	500.00%	4.639	1007.16%	0.020	1900.00%	0.100	203.03%	902.55%
	IBP-Pop	0.016	300.00%	4.170	895.23%	0.020	1900.00%	0.144	336.36%	857.90%
	LDA-Unif	0.015	275.00%	1.029	145.58%	0.002	100.00%	0.114	245.45%	**191.51%**
	LDA-Pop	0.035	775.00%	0.485	15.75%	0.008	700.00%	0.045	36.36%	381.78%
STMV1	IBP-Unif	2.103	1402.14%	1.299	248.26%	1.719	2132.47%	4.503	1175.64%	1239.63%
	IBP-Pop	0.366	161.43%	2.458	558.98%	0.770	900.00%	2.486	604.25%	**556.16%**
	LDA-Unif	4.883	3387.86%	2.636	606.70%	6.052	7759.74%	3.100	778.19%	3133.12%
	LDA-Pop	4.876	3382.86%	2.431	551.74%	6.041	7745.45%	3.251	820.96%	3125.25%

Table 2: Data set summary statistics.

Datasets	Users	Items	Pairs	Density
ML1M	6,040	3,706	1,000,209	4.47%
AZM5	5,541	3,568	64,706	0.33%
STMV1	70,912	10,978	5,094,082	0.65%

3.4 Calibrating Simulations

We compare synthetic data from our simulator to our reference data sets using the K-L divergence [16] between synthetic and observed distributions of four key properties: item popularity, user activity, item-item correlation, and user-user correlation.

We compute item popularity and user activity distributions by counting the frequency of each popularity or activity level (profile size) in the observed data set. We compute item (and user) correlation distributions by sampling 1M unique item (user) pairs, with at least 5 ratings each, from the observed data and computing the cosine similarity between their 0, 1 rating vectors. The 5-rating limit is to reduce the impact of uncorrelated pairs; when we did not do this, K-L divergence was dominated by the zeroes and did not meaningfully compare the distribution of nonzero correlations.

We used Gaussian process minimziation as implemented by scikit-optimize [15] to find model parameters that minimize the K-L divergence for a particular distribution and target data set. We optimized parameters for all four models (each combination of preference and observation model) using each distributions on each of the data sets, yielding a total of 48 tuned models.

We then retuned our models to simultaneously optimize all 4 distributions for a data set by minimizing the *average relative loss* across the four distributions. We define relative loss for a distribution by comparing the K-L divergence on that distribution to the best K-L divergence achieved on that data set with the single-objective optimizations in the previous step. We computed mean relative loss across the four distributions to create a single integrated metric, weighting loss across all properties equally.

3.5 Evaluation Experiments

To simulate recommender evaluation, we held out 20% of each user's observed items as testing data, generated 50-item recommendation lists, and computed commonly-used recommendation accuracy metrics using LensKit [8]. We computed each metric two ways: once with the held-out observable test data as ground truth, and again with the simulated true preference data. We repeated each experiment, including data generation, 100 times.

The Oracle recommender uses relevant item sets to generate perfect recommendations. Popular recommends the most popular items using popularity statistics from the observable training data. Random recommends random unpurchased items for each user.

4 RESULTS

This section presents the results of running our simulations.

4.1 Calibration Results

Table 1 shows the performance of tuned models with respect to each reference data set, averaged over 20 runs. We show K-L divergence and relative loss for each characteristic distribution; models with the best average loss for that data set are in bold.

LDA-based models fit better for ML1M and AMZM5, while the IBP-Pop fit best STMV1. The high average loss on ML1M is because single-statistic optimizations were able to obtain extremely low divergence on similarity distributions (the optimal item-item K-L divergence on ML1M was 0.002), but required significant tradeoffs on similarity distributions in order to match popularity and activity level distributions. Since the ML1M and AMZ5 are pruned, it is possible that item correlations are more important for simulating such data sets than they are for natural data sets.

4.2 Simulation Results

Our evaluation simulations address RQ2 and RQ3. We report the *error* of each metric, defined as $M^{\mathrm{obs}} - M^{\mathrm{truth}}$ where M^{obs} is the metric value using observable test data as ground truth and M^{truth} is the value over true preference data. Figure 2 shows these errors.

With the uniform observation sampler, recall has no bias (error is symmetrically distributed about 0), consistent with its status as an unbiased estimator [19]. When observations are popularity-biased, however, observed data leads to overestimates of true recall.

Observed data generally underestimates both precision and MRR, often substantially. Since we do not simulate users consuming irrelevant items, the set of relevant items in the true data is a superset

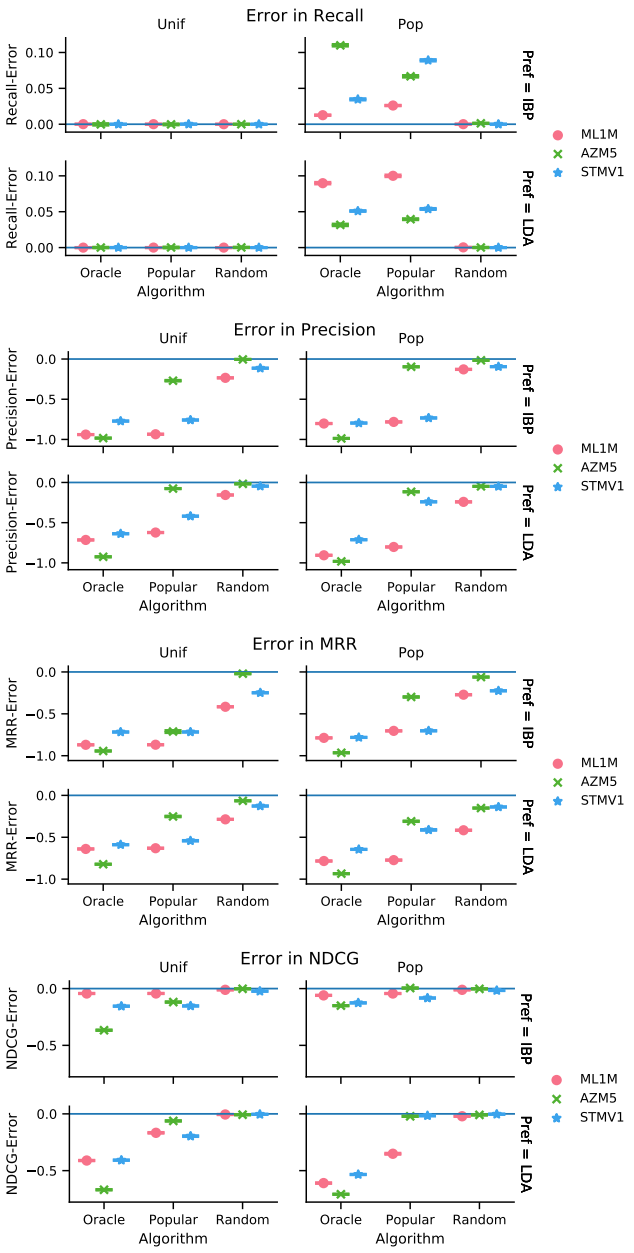

Figure 2: Error of evaluation metrics ($M^{\text{obs}} - M^{\text{truth}}$).

of the observed items, increasing opportunity for recommendations to be "correct". For all models, observed data underestimates these metrics more for the Oracle recommender than for Popular; this calls into question an experiment's ability to assess the performance of algorithms relative to each other, particularly when one performs extremely well. An experiment would correctly conclude that Oracle beats Popular, but would *over*-estimate the effect size.

nDCG is also biased, but its particular biases vary more between experimental conditions and recommenders. This inconsistency suggests it would be more difficult to correct for missing data errors in evaluations using nDCG.

Table 3: Percentage of runs where Oracle beats Popular.

Data	Pref	Obs	P@50	Recall	MRR	NDCG
ML1M	IBP	Unif	87	58	47	65
		Pop	0	0	0	0
	LDA	Unif	100	100	100	100
		Pop	100	100	89	100
AZM5	IBP	Unif	100	100	100	100
		Pop	100	100	99	100
	LDA	Unif	100	100	100	100
		Pop	100	100	100	100
STMV1	IBP	Unif	100	100	86	100
		Pop	100	5	0	85
	LDA	Unif	100	100	100	100
		Pop	100	100	100	100

4.3 Algorithm Ranking

To address RQ4, we looked at the prevalence of *rank inversions*: how often would an experiment erroneously conclude that the Popular recommender is more effective than Oracle? Table 3 shows these results. While the outcomes were usually correct, in several cases they were reliably wrong. Because this happened under IBP with Popular observations, it may be due to popularity bias severely fooling the evaluator on the observable data. LDA did not produce this effect, indicating that extent of this error is sensitive to assumptions.

However, the most realistic scenario we examined — the best-performing model on the unpruned Steam data set — reliably *failed* to select the oracle model over the popular one.

5 CONCLUSIONS AND FUTURE WORK

We have simulated user preference for items and resulting consumption observations in order to estimate error and bias in the results of offline evaluations of recommender systems. To maximize realism, we calibrated our simulation models to match distributions of key characteristics of public existing data sets.

With the exception of recall in the cases where it is already known to be an unbiased estimator, we find substantial error — usually underestimation — in evaluation metrics (RQ2, RQ3). Most concerningly, we find that the *degree* of error varies between algorithms in the same data and experimental condition, undermining estimates of relative differences in algorithm performance using offline evaluation protocols. Evaluations are also sometimes fooled into misranking algorithms. This effect is sensitive to assumptions, but in what we judge to be the most realistic scenario in our study, the evaluation is unable to reliably favor an oracle recommender over a popular-item recommender on multiple metrics (RQ4).

Simulation is a promising tool for understanding recommender evaluations and characterizing their failure modes. The simulations we present here are simple. They do not account for rating values or relative preference and do not reflect users consuming the occasional item they do not like. In future work we hope to extend this study to capture a wider array of user and algorithm behavior.

ACKNOWLEDGMENTS

This work partly supported by the National Science Foundation under grant CHS 17-51278. We thank the People and Information Research Team and Mucun's M.S. thesis committee, Sole Pera and Hoda Mehrpouyan, for their support.

REFERENCES

[1] Joeran Beel and Victor Brunel. 2019. Data Pruning in Recommender Systems Research: Best-Practice or Malpractice?. In *ACM RecSys 2019 Late-Breaking Results*, Vol. 2431. CEUR, 5.

[2] Alejandro Bellogín. 2012. *Recommender System Performance Evaluation and Prediction: An Information Retrieval Perspective.* Ph.D. Dissertation. Universidad Autónoma de Madrid, Madrid, Spain.

[3] Alejandro Bellogin, Pablo Castells, and Ivan Cantador. 2011. Precision-oriented Evaluation of Recommender Systems: An Algorithmic Comparison. In *Proceedings of the Fifth ACM Conference on Recommender Systems (RecSys '11).* ACM, New York, NY, USA, 333–336. https://doi.org/10.1145/2043932.2043996

[4] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research* 3 (March 2003), 993–1022. http://dl.acm.org/citation.cfm?id=944919.944937

[5] Léon Bottou, Jonas Peters, Joaquin Quiñonero Candela, Denis X. Charles, D. Max Chickering, Elon Portugaly, Dipankar Ray, Patrice Simard, and Ed Snelson. 2013. Counterfactual Reasoning and Learning Systems: The Example of Computational Advertising. *Journal of Machine Learning Research* 14, 1 (Jan. 2013), 3207–3260. http://dl.acm.org/citation.cfm?id=2567709.2567766

[6] Rocío Cañamares and Pablo Castells. 2018. Should I Follow the Crowd?: A Probabilistic Analysis of the Effectiveness of Popularity in Recommender Systems. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR '18).* ACM, New York, NY, USA, 415–424. https://doi.org/10.1145/3209978.3210014

[7] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. 2010. Performance of Recommender Algorithms on Top-n Recommendation Tasks. In *Proceedings of the Fourth ACM Conference on Recommender Systems (RecSys '10).* ACM, New York, NY, USA, 39–46. https://doi.org/10.1145/1864708.1864721

[8] Michael D. Ekstrand. 2018. The LKPY Package for Recommender Systems Experiments: Next-Generation Tools and Lessons Learned from the LensKit Project. *CoRR* abs/1809.03125 (2018). arXiv:1809.03125 http://arxiv.org/abs/1809.03125

[9] Michael D. Ekstrand and Joseph A. Konstan. 2019. Recommender Systems Notation: Proposed Common Notation for Teaching and Research. *CoRR* abs/1902.01348 (2019). arXiv:1902.01348 http://arxiv.org/abs/1902.01348

[10] Michael D. Ekstrand and Vaibhav Mahant. 2017. Sturgeon and the Cool Kids: Problems with Random Decoys for Top-N Recommender Evaluation. In *Proceedings of the Thirtieth International Florida Artificial Intelligence Research Society Conference.* Association for the Advancement of Artificial Intelligence, 639–644. https://aaai.org/ocs/index.php/FLAIRS/FLAIRS17/paper/view/15534

[11] Alexandre Gilotte, Clément Calauzènes, Thomas Nedelec, Alexandre Abraham, and Simon Dollé. 2018. Offline A/B Testing for Recommender Systems. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining (WSDM '18).* ACM, New York, NY, USA, 198–206. https://doi.org/10.1145/3159652.3159687

[12] Thomas L. Griffiths and Zoubin Ghahramani. 2011. The Indian Buffet Process: An Introduction and Review. *Journal of Machine Learning Research* 12 (July 2011), 1185–1224. http://dl.acm.org/citation.cfm?id=1953048.2021039

[13] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems* 5, 4 (Dec. 2015), 19:1–19:19. https://doi.org/10.1145/2827872

[14] Ruining He and Julian McAuley. 2016. Ups and Downs: Modeling the Visual Evolution of Fashion Trends with One-Class Collaborative Filtering. In *Proceedings of the 25th International Conference on World Wide Web (WWW '16).* International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 507–517. https://doi.org/10.1145/2872427.2883037

[15] Tim Head, MechCoder, Gilles Louppe, Iaroslav Shcherbatyi, fcharras, ZÃŁ VinÃŋcius, cmmalone, Christopher SchrÃűder, nel215, Nuno Campos, Todd Young, Stefano Cereda, Thomas Fan, rene rex, Kejia (KJ) Shi, Justus Schwabedal, carlosdanielcsantos, Hvass-Labs, Mikhail Pak, SoManyUsernamesTaken, Fred Callaway, LoÃŕc EstÃĺve, Lilian Besson, Mehdi Cherti, Karlson Pfannschmidt, Fabian Linzberger, Christophe Cauet, Anna Gut, Andreas Mueller, and Alexander Fabisch. 2018. scikit-optimize/scikit-optimize: v0.5.2. https://doi.org/10.5281/zenodo.1207017

[16] S. Kullback. 1997. *Information Theory and Statistics.* Dover Publications. https://books.google.com/books?id=05LwShwkhFYC

[17] Daryl Lim, Julian McAuley, and Gert Lanckriet. 2015. Top-N Recommendation with Missing Implicit Feedback. In *Proceedings of the 9th ACM Conference on Recommender Systems (RecSys '15).* ACM, New York, NY, USA, 309–312. https://doi.org/10.1145/2792838.2799671

[18] Apurva Pathak, Kshitiz Gupta, and Julian McAuley. 2017. Generating and Personalizing Bundle Recommendations on Steam. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '17).* ACM, New York, NY, USA, 1073–1076. https://doi.org/10.1145/3077136.3080724

[19] Harald Steck. 2010. Training and Testing of Recommender Systems on Data Missing Not at Random. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '10).* ACM, New York, NY, USA, 713–722. https://doi.org/10.1145/1835804.1835895

[20] Adith Swaminathan and Thorsten Joachims. 2015. Batch Learning from Logged Bandit Feedback Through Counterfactual Risk Minimization. *J. Mach. Learn. Res.* 16, 1 (Jan. 2015), 1731–1755. http://dl.acm.org/citation.cfm?id=2789272.2886805

[21] Yee Whye Teh and Dilan Görür. 2009. Indian Buffet Processes with Power-law Behavior. In *Proceedings of the 22nd International Conference on Neural Information Processing Systems (NIPS'09).* Curran Associates Inc., USA, 1838–1846. http://dl.acm.org/citation.cfm?id=2984093.2984299

A MODEL AND PARAMETER DETAILS

A.1 Preferential Attachment Preference (IBP)

The IBP model with parameters $\alpha > 0$, $\sigma \in [0, 1)$, and $c > -\sigma$ is defined as follows [21]:

(1) The first user likes Poisson(α) items.
(2) User $(n + 1)$ likes previously-known item i with probability $\frac{m_i - \sigma}{n+c}$ (where m_i is the number of users who like item i) and likes Poisson($\alpha \frac{\Gamma(1+c)\Gamma(n+c+\sigma)}{\Gamma(n+1+c)\Gamma(c+\sigma)}$) new items.

c controls how likely the user is to rate new vs. old items. σ governs the power0law behavior of the generated preference matrix; $\sigma = 0$ yields a traditional IBP, with larger values yielding stronger power-law distributions of item popularity. α controls the density of the generated preference matrix. When $\sigma > 0$, the process generates on average $\alpha|U|^\sigma$ items; when $\sigma = 0$ and $c = 1$, it generates approximately $\alpha(\log|U| + \gamma)$ items on average [21], where γ is Euler's constant [12].

A.2 Correlated Preference (LDA)

The LDA generation process [4] with K latent features operates as follows:

(1) Draw K feature-item vectors $\vec{\phi}_k \in [0, 1]^{|I|}$ from Dirichlet(β).
(2) For each user:
   (a) Draw a latent feature vector $\vec{\theta}_u \in [0, 1]^K$ from Dirichlet(α).
   (b) Draw n_u (the number of items) from Poisson(λ).
   (c) Draw items $i_1, \ldots, i_{n_u}$ liked by user u by drawing feature $k_x \sim$ Multinomial($\vec{\theta}_u$) and i_x from Multinomial($\vec{\phi}_{k_x}$).
(3) De-duplicate user-item pairs to produce implicit user preference samples.

To reduce the number of parameters for fitting efficiency, we use symmetric LDA, where α is a constant vector with all values equal to $a > 0$, and likewise β is constant $b > 0$. These parameters a and b control the breadth of user preferences; when $a < 1$, the values of $\vec{\theta}_u$ concentrate on a few of the K dimensions, making the userâĂŹs preferences concentrate on a few items if $b < 1$. The parameter λ controls the average number of items each user likes. The parameter K controls the size of the latent feature space, affecting the diversity of user-item preference patterns in the whole true preference data.