Sturgeon and the Cool Kids: Problems with Random Decoys for Top-N Recommender Evaluation

Michael D. Ekstrand[1,2] and Vaibhav Mahant[2]

[1] People and Information Research Team
Dept. of Computer Science, Boise State University
Boise, ID USA
michaelekstrand@boisestate.edu

[2] Department of Computer Science
Texas State University
San Marcos, TX USA
vaibhav.mahant@gmail.com

Abstract

Top-N evaluation of recommender systems, typically carried out using metrics from information retrieval or machine learning, has several challenges. Two of these challenges are *popularity bias*, where the evaluation intrinsically favors algorithms that recommend popular items, and *misclassified decoys*, where items for which no user relevance is known are actually relevant to the user, but the evaluation is unaware and penalizes the recommender for suggesting them. One strategy for mitigating the misclassified decoy problem is the one-plus-random evaluation strategy and its generalization, which we call *random decoys*. In this work, we explore the random decoy strategy through both a theoretical treatment and an empirical study, but find little evidence to guide its tuning and show that it has complex and deleterious interactions with popularity bias.

Introduction

Offline evaluation of recommender system performance relies on two families of metrics: error metrics that assess the recommender's ability to predict unknown ratings, and Top-*N* metrics that judge its ability to identify and retrieve items relevant to the user. In recent years, top-*N* metrics have gained dominance as they more accurately reflect the tasks that recommenders most often perform. They are used both to assess recommender quality in offline evaluation and to train and pre-validate recommenders for online applications. The common metrics, such as precision & recall, AUC, nDCG, MAP, and MRR, come from information retrieval (IR) and machine learning (ML) contexts (Gunawardana and Shani 2009). However, the high data sparsity in recommender applications – and in many modern IR and ML settings – violate key assumptions these metrics make about the coverage of available relevance data. The extent and impact of these violations has not been thoroughly explored.

While the basic structure of offline recommender evaluation is well-known, for the purposes of establishing useful terminology we rehearse it as follows:

1. Partition data –records of users rating, purchasing, or clicking on items – into *training* and *test* sets. For simplicity, we will call these *ratings* when the user provides an explicit preference judgement of an item, and *purchases* when the user takes an action to adopt the item from which preference is inferred (implicit feedback).
2. Train an algorithm and its model on the training data.
3. Use the trained model to produce recommendations for users with test data. Recommendations are produced from a *candidate set*. The exact constitution of the candidate set varies (Bellogin, Castells, and Cantador 2011); for the purposes of our discussion, we consider it to be the union of two disjoint sets of items: the *test items*, which the user has rated or purchased in the test data, and the *decoy items*, which are items the user has not rated.
4. Measure the recommender's quality using standard metrics from IR and ML. When rating data is available, metrics such as nDCG and NDPM allow measurement of the consistency between the recommender's rankings and those induced by user ratings; when considering unary purchase data, the metrics effectively assess the recommender's ability to separate test items from decoy items. Items the user has purchased are assumed relevant; the decoy items, about which no information is available, are assumed to be irrelevant.

These metrics work best on a *fully-coded corpus*, where relevance judgements are available for all items, or a probabilistic approximation of such a corpus built with techniques such as pooling (Buckley and Voorhees 2004). However, recommender data sets such as those from MovieLens (Harper and Konstan 2015) and BookCrossing

(Ziegler et al. 2005), commonly used as benchmark data sets for recommender system evaluation, are missing relevance judgements for most user-item pairs. Further, due to the highly subjective nature of relevance in recommendation tasks, most methods used in IR to expand the data set's coverage do not apply and even 'more complete' data such as that collected in commercial settings will have similar problems. When we apply IR metrics to recommender systems, therefore, we are violating the assumptions necessary to make them correct and meaningful.

This assumption failure results in at least two specific problems with recommender evaluation:

- **Popularity bias**, where evaluations favor recommenders that tend to recommend popular items. While popularity is a useful signal in recommendation, an excessive emphasis on it may penalize recommenders that produce more novel, but equally satisfactory, recommendations.
- **Misclassified decoys**, where one or more of the decoy items is of interest to the user. If the user did not rate or purchase the item because they did not know about it, then not only is the recommendation relevant, it could be a *better* recommendation than one of the test items because it would help the user find something they like that they didn't know about before, whereas they did know about the test item somehow.

Misclassified decoys are particularly pernicious because they make it near-impossible to evaluate the recommender's effectiveness at exposing the user to new material. If we were able to develop a perfect recommender that could accurately identify items the user would like but is unaware of, such a recommender would likely lose in accuracy metrics because it prefers such excellent items over test items.

Bellogin explores a number of solutions to the popularity bias problem (Bellogin 2012), such as popularity-stratified sampling for test items. For the misclassified decoy problem, there are a number of solutions in the information retrieval literature (Yilmaz and Aslam 2006) in addition to the one-plus-random protocol (Cremonesi, Koren, and Turrin 2010; Koren 2008). Of the IR solutions we have been able to identify, however, only rank effectiveness is applicable to recommender systems, as pooling requires an objective concept of relevance and inference begs the question by effectively requiring a perfect recommender in order to evaluate a recommender.

In this paper, we examine the *random decoys* protocol, a generalization of the one-plus-random protocol. In this experimental protocol, the candidate set consists of the test items plus a randomly-selected set of N decoy items, rather than the typical choice of considering all unseen items to be candidates (one-plus-random is an iterated version of random decoys where the user's test items are considered one at a time, each with a random set of decoys). The key idea is that there are probably not very many unknown-but-relevant items, so a randomly-selected set of items will probably not result in misclassified decoys. 1000 is a common size for the decoy set (Cremonesi, Koren, and Turrin 2010; Koren 2008); the optimal size depends on the size of the data set and the estimated prevalence of unknown relevant items.

We started this work believing that the random decoy strategy is a promising technique for mitigating misclassified decoys, and our goal was to gain insight into how to select a good decoy set size. We found instead that the distribution of item goodness required to avoid misclassified decoys with reasonable probability is unreasonable, there is a serious discrepancy between the theoretical and observed behavior of the random decoy strategy with respect to the popularity bias problem, and saw no clear points to investigate as potential decoy set sizes. Together, these findings cast doubt on the usefulness of the random decoy protocol.

In the remainder of this paper, we provide a theoretical treatment of the statistical infeasibility of the random decoy protocol as a remedy for the misclassified decoy problem, an experiment demonstrating it exacerbates popularity bias, and conclude with some thoughts on best practices for evaluation and future work.

Related Work

There is a long line of work investigating various means of evaluating recommender systems using publicly-available data (Breese, Heckerman, and Kadie 1998; Herlocker et al. 2004; Gunawardana and Shani 2009; Bellogin, Castells, and Cantador 2011; Cremonesi, Koren, and Turrin 2010), and an even longer history of such offline evaluations in information retrieval (Goffman 1964; Cleverdon 1967; Voorhees 2001).

Of particular relevance to our work, recommender system evaluation work has identified the popularity bias problem (Bellogin 2012) and attempted to mitigate misclassified decoys (Cremonesi, Koren, and Turrin 2010). Other lines of research have identified deficiencies in accuracy-only pictures of recommender performance (McNee, Riedl, and Konstan 2006) and the connection – or lack thereof – between offline measures and user response in an online setting (Ekstrand et al. 2014; Rossetti, Stella, and Zanker 2016).

The information retrieval community has worked on addressing these challenges in their own data; three particular approaches have had good impact:

- *Pooling* (Buckley and Voorhees 2004), where multiple retrieval algorithms are used to identify candidate items for human judgement; the remaining items can be assumed irrelevant with higher confidence.
- *Rank effectiveness* (the 'TestItems' strategy of Bellogin et al. (2011)), where unknown items are ignored

and the results are evaluated solely on their rank consistency with available relative preference judgements. This requires ratings or relative preference.

- *Relevance inference* (Aslam and Yilmaz 2007), where relevance of unknown items is algorithmically inferred based on similarity with known documents.

There is significant variance in the accuracy metrics reported by different experimental platforms (Said and Bellogin 2014) and between different papers. Part of this is due to a lack of best practices in recommender evaluation (Konstan and Adomavicius 2013). Previous studies have found many top-*N* evaluation metrics to be mostly rank-consistent (Herlocker et al. 2004; Bellogin, Castells, and Cantador 2011) – ordering algorithms consistently with respect to accuracy – but they are not always so, and even when rank is consistent the differing values of accuracy metrics impact statistical significance tests and other valuable analyses. We believe that a lack of in-depth understanding of the implications of different variations in evaluation strategy makes it difficult to resolve this situation. In this paper, we report on one of our attempts to illuminate these issues in the interest of promoting more rigorous, comparable, and interpretable recommender systems research.

The Data

For our analysis and experiments, we use the MovieLens data sets (Harper and Konstan 2015). These data sets contain 5-star movie ratings provided by the users of the MovieLens movie recommendation service. Table 1 summarizes the data sets, and Figure 1 shows the distribution of rating counts per item.

Set	Ratings	Users	Items	Density
100K	100K	943	1682	6.30%
1M	1M	6040	3706	4.47%
10M	10M	69878	10677	1.34%
20M	20M	138493	26744	0.54%

Table 1: Data Sets

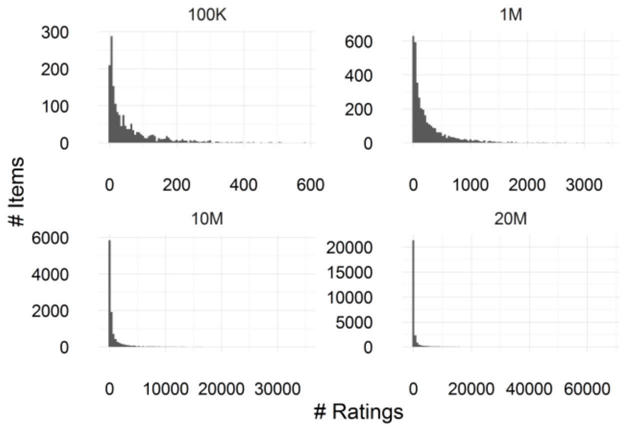

Figure 1: Distribution of item rating counts.

Random Decoys in Theory

The underlying principle of random decoys can be derived from Sturgeon's Law (Sturgeon 1958): '90% of everything is crud'. This has been further refined to the rough approximation that only 1% of anything is 'really good' (Miller 1960). If most items are low-quality, and a further large proportion of them are not of interest to a particular user, then we can assume that a randomly selected item is probably not interesting to the user. Some users will enjoy items that would generally fall into the 'crud' category – for example, B-movie aficionados – but that does not substantially change our analysis. We can focus on the *a priori* probability $g = P(i \in G_u)$ of a user thinking an item is 'good'. For mathematical simplicity, we assume that goodness of each item is independent.

Probability of Misclassified Decoys

For a decoy set size N, we can reason about the probability that the set contains at least one good item: $P(L_N \cap G_u \neq \emptyset) = P(\forall i \in L_N . i \notin G_u) = P(i \notin G_u)^N = (1-g)^N$. If we desire 95% certainty that our decoy set does not contain a good item, then we require $1 - g = 0.95^{1/N}$; for the common size of 1000, $g = 0.00001$: we must expect that only 1 in 10K items is relevant to a user's desires. Given that users rate tens to hundreds of movies in a 25K-movie set, this seems far too low. A decoy set size of 100 requires that no more than 1 in 1000 items be relevant, which also seems unrealistically high.

We can consider other probabilities, such as the likelihood of having more than m or f% misclassified decoys. However, if the decoy set contains even one relevant item, then it is the recommender's job to find that item, and conceivably to prioritize it above all known items (depending on the recommender application).

Popularity Bias and Random Decoys

Popularity bias (Bellogin, Castells, and Cantador 2011) is caused by the fact that popular items are more likely than unpopular items to be rated, including in the test set, so picking the most popular items in the candidate set is a fairly reliable means of separating test items from decoys. But popular items may not be the best recommendations for the same reason that test items may not be the best: if the recommender's goal is to help users discover new items, then recommending ones they already know is not helpful. Therefore, top-*N* evaluation protocols seem to overemphasize the value of popularity as a recommendation signal.

The random decoy protocol makes this worse. Because popularity follows a heavily skewed distribution, as can be seen for the MovieLens data in Figure 1, most items are not popular. Therefore, while the full set of items contains a number of items of similar popularity to the test items that the recommender should distinguish the test items from, the random decoy set probably does not contain these items, making popularity an even better predictor of whether an item appears in the user's test ratings. While the exact probabilities will depend on user behavior and the relationship between relevance and popularity, we do not expect random decoys to improve popularity's excessive advantage.

Evaluations of recommenders that take into account the desire to explore the long tail of worthwhile but less-popular items often augment accuracy with additional novelty measures. However, the difficult-to-untangle link of popularity with available assessments of 'goodness' makes these evaluations difficult to start with, and deepening that entanglement through our evaluation strategy will not help.

Random Decoys in Practice

Having examined the theoretical dynamics of the random decoy protocol, we now turn to our experiment on the impact of decoy set size on recommender evaluation results.

Experimental Method

To empirically examine the effect of decoy set size on recommender evaluation, we tested several recommender algorithms with different selections of decoy set sizes. We used a pre-release version of LensKit version 3.0 (Ekstrand et al. 2011) and the MovieLens data sets (Harper and Konstan 2015) for our experiment.

We tested the following algorithms:

- *Popular* recommends the most often-rated movies.
- *PersMean* recommends items with the highest average rating.
- *Item-Item* is an item-based collaborative filter (Sarwar et al. 2001) configured to use 20 neighbors, cosine similarity, and item-mean normalization.
- *FunkSVD* is a gradient descent matrix factorization technique (Funk 2006; Paterek 2007) with 40 latent features and 125 training iterations per feature.

For each data set, we generated 5 disjoint sets of test users; for 100K and 1M, these sets encompassed all users, while for 10M and 20M they each contained 5000 users. For each test user, we randomly selected 5 ratings as test ratings; the user's remaining ratings, along with all ratings from users not in their test set, formed the training data.

We then produced 10-, 25-, and 100-item recommendation lists, varying the number of decoy items selected, and measured precision, recall, MRR, MAP, and nDCG. For precision metrics, we considered an item 'relevant' if the user rated it. We report results primarily on nDCG; other metrics yield similar patterns (see Figure 6 for MRR).

Full source code to reproduce the experiment and analysis is available in the Supplemental Files accompanying this paper, and online at:
https://works.bepress.com/michael-ekstrand/19/

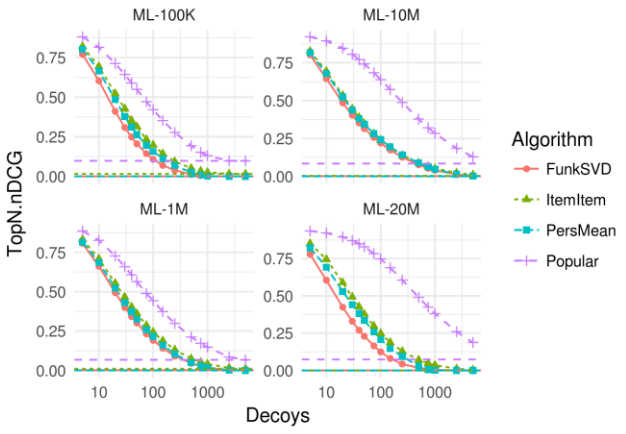

Figure 2: nDCG on 10-item lists

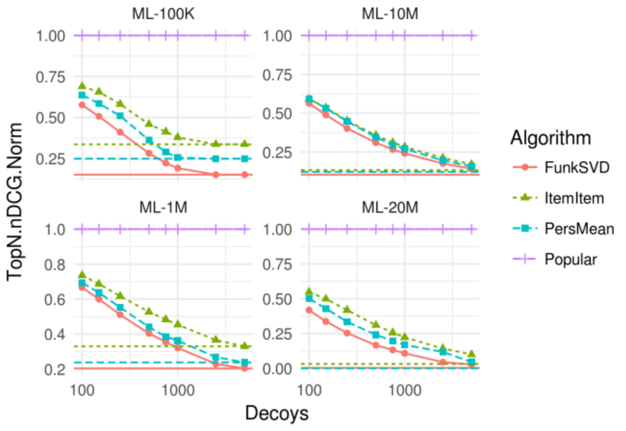

Figure 3: nDCG on 10-item lists (fraction of Popular score)

Results

Figures 2 and 3 show the performance of the algorithms we tested on nDCG with 10-item recommendation lists; Figure 2 shows raw nDCG performance, and Figure 3 shows nDCG normalized to the fraction of Popular's performance each algorithm achieves. Figures 4 and 5 show the same for 100-item lists. In all of these charts, the horizontal lines are the performance with a full decoy set (the candidates set consists of all unrated items).

We observe a few things in these plots:

- For short lists, the absolute difference in algorithm performance increases – with Popular gaining an advantage – for a moderate range of decoy set sizes including 1000. It is low for very small decoy sets – where there are not many non-decoys to choose from – and for large decoy sets, approaching the traditional evaluation protocol without random decoys.
- The proportional difference in performance – fraction of Popular's performance achieved by a rating-based recommender – increases fairly consistently as the decoy set size decreases.

- The discrepancy between absolute and proportional performance differences introduces further room for inconsistency in experimental protocols and analysis. Also, many papers report '% improvement' in their error metrics, but statistical tests typically rely on absolute differences unless error values are normalized.
- Inflection points do not seem to be connected to the fraction of the items in the decoy set.

These results are not consistent with the theoretically-expected effect that randomly subsetting decoys would increase popularity bias. The change in absolute difference of metric values also implies that statistical significance can easily depend on the choice of decoy set size, a clearly undesirable situation.

We also note, though, that random decoys do not change the performance ordering of any algorithms on the data sets and metrics we have run. While a more careful comparison would need to account for tuning parameters with varying decoy strategies, if random decoys do not affect the decisions made as a result of a recommender evaluation experiment, then they do not seem to provide much value.

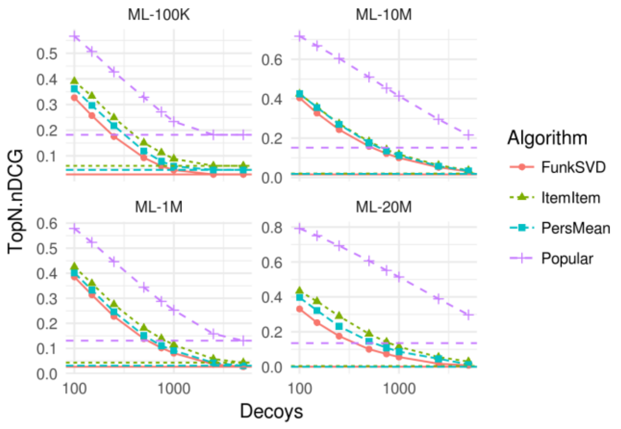

Figure 4: nDCG over 100-item lists

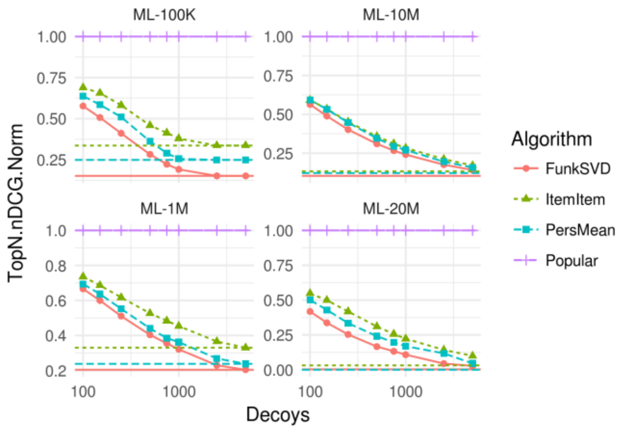

Figure 5: nDCG over 100-item lists (fraction of Popular score)

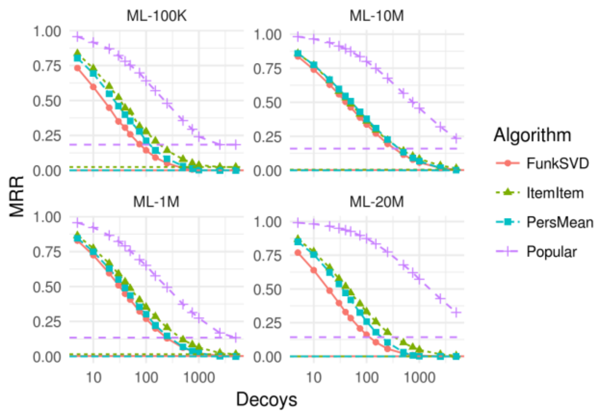

Figure 6: Mean Reciprocal Rank for 10-item lists.

Conclusion

While randomly-selected decoys, on their face, seem like a promising method for dealing with the relevant decoy problem, our experiments failed to identify a means of selecting an appropriate number of decoy items, and further provided evidence that the random decoy strategy interacts in complex ways with popularity bias; in theory, it should exacerbate popularity bias, but empirical data find that randomly restricting the decoy set decreases the relative performance advantage of Popular over collaborative filtering algorithms. We also noted a dependency of the absolute different in performance on the decoy set size, with adverse implications for statistical significance testing.

Since there is no known means for determining an effective number of decoys (the choice of which Koren also acknowledges as arbitrary (Koren 2008)), and there is a complex relationship with other known recommender evaluation problems that still needs to be fully characterized, we suggest that the random decoy strategy is not, at present, an effective means of evaluating recommender quality.

Rather, randomly selecting decoys introduces additional variance into evaluation that harms reproducibility and comparability of research results, for a benefit that cannot currently be quantified and may not be achieved.

It is still unknown just how pervasive the popularity bias and misclassified decoy problems are and the full impact they have on recommender evaluation outcomes. We hope in future work to quantify the extent and impact of these problems and perhaps to devise better strategies for mitigating them, as well as to explore our empirical results on additional data sets.

Until then, given the existence and unknown nature of the problems we discuss, the most reliable means of assessing top-*N* recommendation quality when ratings or other relative preference data is available seems to be a *rank effectiveness* strategy, called 'TestItems' by Bellogin, Castells, and Cantador (2011): limit the candidate set to the test items and assess the recommender on its ability to order these consistent with the user's expressed preferences. This approach tests the algorithm's ability to rank while avoiding many of the sparsity pitfalls causing top-N evaluation to be unreliable, but is little comfort to those working with implicit feedback data such as clicks and purchases. It also does not test the recommender's ability to identify interesting items from a large pool, but the misclassified decoy problem casts doubt on the ability of any offline top-N evaluation to measure this in a way that is consistent with the user experience desired for most recommender applications.

References

Aslam, J. A.; and Ylimaz, E. 2007. "Inferring Document Relevance from Incomplete Information." In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, 633–642. CIKM '07. New York, NY, USA: ACM. doi:10.1145/1321440.1321529.

Bellogin, A. 2012. "Performance Prediction and Evaluation in Recommender Systems: An Information Retrieval Perspective." Ph.D diss., Universidad Autónoma de Madrid, Madrid, Spain. http://ir.ii.uam.es/~alejandro/thesis/.

Bellogin, A.; Castells, P.; and Cantador, I. 2011. "Precision-Oriented Evaluation of Recommender Systems: An Algorithmic Comparison." In *Proceedings of the Fifth ACM Conference on Recommender Systems*, 333–336. New York, NY, USA: ACM.

Breese, J. S.; Heckerman, D.; and Kadie, C. 1998. "Empirical Analysis of Predictive Algorithms for Collaborative Filtering." In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, 43–52.

Buckley, C.; and Voorhees, E.M.. 2004. "Retrieval Evaluation with Incomplete Information." In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 25–32. New York, NY, USA: ACM.

Cleverdon, C. 1967. "The Cranfield Tests on Index Language Devices." *Aslib Proceedings* 19 (6): 173–94.

Cremonesi, P.; Koren, Y.; and Turrin, R. 2010. "Performance of Recommender Algorithms on Top-N Recommendation Tasks." In *Proceedings of the Fourth ACM Conference on Recommender Systems*, 39–46. New York, NY, USA: ACM.

Ekstrand, M. D.; Harper, F. M.; Willemsen, M.C.; and Konstan, J. A. 2014. "User Perception of Differences in Recommender Algorithms." In *Proceedings of the 8th ACM Conference on Recommender Systems (RecSys '14)*, 161–68. New York, NY: ACM.

Ekstrand, M. D.; Ludwig, M.; Konstan, J. A.; and Riedl, J. T. 2011. "Rethinking the Recommender Research Ecosystem: Reproducibility, Openness, and LensKit." In *Proceedings of the Fifth ACM Conference on Recommender Systems (RecSys '11)*, 133–40. New York, NY: ACM.

Funk, S. 2006. "Netflix Update: Try This at Home." Blog. *The Evolution of Cybernetics*. December 11. http://sifter.org/~simon/journal/20061211.html.

Goffman, W. 1964. "A Searching Procedure for Information Retrieval." *Information Storage and Retrieval* 2 (2): 73–78.

Gunawardana, A.; and Shani, G. 2009. "A Survey of Accuracy Evaluation Metrics of Recommendation Tasks." *J. Mach. Learn. Res.* 10: 2935–62.

Harper, F. M.; and Konstan, J. A. 2015. "The MovieLens Datasets: History and Context." *ACM Trans. Interact. Intell. Syst.* 5 (4): 19:1–19:19. doi:10.1145/2827872.

Herlocker, J.; Konstan, J. A.; Terveen, L.; and Riedl, J. 2004. "Evaluating Collaborative Filtering Recommender Systems." *ACM Trans. Inf. Syst.* 22 (1): 5–53.

Konstan, J. A.; and Adomavicius, G. 2013. "Toward Identification and Adoption of Best Practices in Algorithmic Recommender Systems Research." In *Proceedings of the International Workshop on Reproducibility and Replication in Recommender Systems Evaluation*, 23–28. RepSys '13. New York, NY, USA: ACM.

Koren, Y. 2008. "Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model." In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '08),* 426–34. New York, NY: ACM.

McNee, S.; Riedl, J.; and Konstan, J. A. 2006. "Being Accurate Is Not Enough: How Accuracy Metrics Have Hurt Recommender Systems." In *CHI '06 Extended Abstracts on Human Factors in Computing Systems*, 1097–1101. Montréal, Québec, Canada: ACM.

Miller, P. S. 1960. *Astounding Science Fact & Fiction*, 162: 2.

Paterek, A. 2007. "Improving Regularized Singular Value Decomposition for Collaborative Filtering." In *KDD Cup and Workshop 2007*.

Rossetti, M.; Stella, F.; and Zanker, M. 2016. "Contrasting Offline and Online Results When Evaluating Recommendation Algorithms." In *Proceedings of the 10th ACM Conference on Recommender Systems*, 31–34. New York, NY, USA: ACM.

Said, A.; and Bellogin, A. 2014. "Comparative Recommender System Evaluation: Benchmarking Recommendation Frameworks." In *Proceedings of the Eighth ACM Conference on Recommender Systems (RecSys '14)*. New York, NY: ACM.

Sarwar, B.; Karypis, G.; Konstan, J.; and Reidl, J.. 2001. "Item-Based Collaborative Filtering Recommendation Algorithms." In *Proceedings of the 0th International World Wide Web Confernece (WWW '01)*, 285–95. ACM.

Sturgeon, T. 1958. "Sturgeon's Law." *Venture Science Fiction*, 66: 2.

Voorhees, E. M. 2001. "The Philosophy of Information Retrieval Evaluation." In *Evaluation of Cross-Language Information Retrieval Systems*, edited by Carol Peters, Martin Braschler, Julio Gonzalo, and Michael Kluck, 355–70. Lecture Notes in Computer Science 2406. Springer Berlin Heidelberg.

Yilmaz, E.; and Aslam, J. A. 2006. "Estimating Average Precision with Incomplete and Imperfect Judgments." In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, 102–111. CIKM '06. New York, NY, USA: ACM.

Ziegler, C. N.; McNee, S.; Konstan, J. A.; and Lausen, G. 2005. "Improving Recommendation Lists through Topic Diversification." In *Proceedings of the 14th International Conference on World Wide Web*, 22–32. New York, NY: ACM.