Case Study Evaluation of Mahout as a Recommender Platform

Carlos E. Seminario
Software and Information Systems Dept.
University of North Carolina Charlotte
cseminar@uncc.edu

David C. Wilson
Software and Information Systems Dept.
University of North Carolina Charlotte
davils@uncc.edu

ABSTRACT

Various libraries have been released to support the development of recommender systems for some time, but it is only relatively recently that larger scale, open-source platforms have become readily available. In the context of such platforms, evaluation tools are important both to verify and validate baseline platform functionality, as well as to provide support for testing new techniques and approaches developed on top of the platform. We have adopted Apache Mahout as an enabling platform for our research and have faced both of these issues in employing it as part of our work in collaborative filtering. This paper presents a case study of evaluation focusing on accuracy and coverage evaluation metrics in Apache Mahout, a recent platform tool that provides support for recommender system application development. As part of this case study, we developed a new metric combining accuracy and coverage in order to evaluate functional changes made to Mahout's collaborative filtering algorithms.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval–*Information filtering*

General Terms

Algorithms, Experimentation, Measurement

Keywords

Recommender systems, Evaluation, Mahout

1. INTRODUCTION

Selecting a foundational platform is an important step in developing recommender systems for personal, research, or commercial purposes. This can be done in many different ways: the platform may be developed from the ground up, an existing recommender engine may be contracted (e.g., OracleAS Personalization[1]), code libraries can be adapted, or a platform may be selected and tailored to suit (e.g., LensKit[2], MymediaLite[3], Apache Mahout[4], etc.). In some cases, a combination of these approaches will be employed.

For many projects, and particularly in the research context, the ideal situation is to find an open-source platform with many active contributors that provides a rich and varied set of recommender system functions that meets all or most of the baseline development requirements. Short of finding this ideal solution, some minor customization to an already existing system may be the best approach to meet the specific development requirements. Various libraries have been released to support the development of recommender systems for some time, but it is only relatively recently that larger scale, open-source platforms have become readily available. In the context of such platforms, evaluation tools are important both to verify and validate baseline platform functionality, as well as to provide support for testing new techniques and approaches developed on top of the platform. We have adopted Apache Mahout as an enabling platform for our research and have faced both of these issues in employing it as part of our work in collaborative filtering recommenders.

This paper presents a case study of evaluation for recommender systems in Apache Mahout, focusing on metrics for accuracy and coverage. We have developed functional changes to the baseline Mahout collaborative filtering algorithms to meet our research purposes, and this paper examines evaluation both from the standpoint of tools for baseline platform functionality, as well as for enhancements and new functionality. The objective of this case study is to evaluate these functional changes made to the platform by comparing the baseline collaborative filtering algorithms to the changed algorithms using well known measures of accuracy and coverage [6]. Our goal is not to validate algorithms that have already been tested previously, but to assess whether, and to what extent, the functional enhancements have improved the accuracy and coverage performance of the baseline out-of-the-box Mahout platform. Given the interplay between accuracy and coverage in this context, we developed a unified metric to assess accuracy vs. coverage trade-offs when evaluating functional changes made to Mahout's collaborative filtering algorithms.

Copyright is held by the author/owner(s). Workshop on Recommendation Utility Evaluation: Beyond RMSE (RUE 2012), held in conjunction with ACM RecSys 2012. September 9, 2012, Dublin, Ireland.

[1]http://download.oracle.com/docs/cd/B10464_05/bi.904/b12102/1intro.htm
[2]http://lenskit.grouplens.org/
[3]http://www.ismll.uni-hildesheim.de/mymedialite/
[4]http://mahout.apache.org

2. RELATED WORK

Revisiting evaluation in the context of recommender platforms has received recent attention in the thorough evaluation of the LensKit platform using previously tested collaborative filtering algorithms and metrics, as reported in [2]. A comprehensive set of guidelines for evaluating recommender systems was provided by Herlocker et al [6]; these guidelines highlight the use of evaluation metrics such as accuracy and coverage and suggest the need for an ideal "general coverage metric" that would combine coverage with accuracy to yield an overall "practical accuracy" measure. Many of these evaluation metrics and techniques have also been covered recently in [12].

Recommender system research has been primarily concerned with improving recommendation accuracy [7]; however, other metrics such as coverage [10, 4] and also novelty and serendipity [6, 3] have been deemed necessary because accuracy alone is not sufficient to properly evaluate the system. Mcnee et al [7] states that recommendations that are most accurate according to the standard metrics are sometimes not the most useful to users and outlines a more user-centric approach to evaluation. The interplay between accuracy and other metrics such as coverage and serendipity creates trade-offs for recommender system implementers and this has been widely discussed in the literature, e.g., see [4, 3] and our previous work discussing trade-offs between accuracy and robustness [11].

3. SELECTING APACHE MAHOUT

To support our research in collaborative filtering, several recommender system platforms were surveyed, including LensKit, easyrec[5], and MymediaLite. We selected Mahout because it provides many of the desired characteristics required for a recommender development workbench platform. Mahout is a production-level, open-source, system and consists of a wide range of applications that are useful for a recommender system developer: collaborative filtering algorithms, data clustering, and data classification. Mahout is also highly scalable and is able to support distributed processing of large data sets across clusters of computers using Hadoop[6]. Mahout recommenders support various similarity and neighborhood formation calculations, recommendation prediction algorithms include user-based, item-based, Slope-One and Singular Value Decomposition (SVD), and it also incorporates Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) evaluation methods. Mahout is readily extensible and provides a wide range of Java classes for customization. As an open-source project, the Mahout developer/contributor community is very active; the Mahout wiki also provides a list of developers and a list of websites that have implemented Mahout[7].

3.1 Uncovering Mahout Details

Although Mahout is rich in documentation, there are implementation details on *how Mahout works* that could only be understood by looking at the source code. Thus, for clarity in evaluation, we needed to verify the implementation of baseline platform functionality. The following describes some of these details for Mahout 0.4 'out-of-the-box':

[5]http://easyrec.org/

[6]http://hadoop.apache.org/

[7]https://cwiki.apache.org/MAHOUT/mahout-wiki.html

Similarity Weighting: Mahout implements the classic Pearson Correlation as described in [8, 5]. Similarity weighting is supported in Mahout and consists of the following method:

```
scaleFactor = 1.0 - count / (num + 1);
if (result < 0.0)
  result = -1.0 + scaleFactor * (1.0 + result);
else
  result = 1.0 - scaleFactor * (1.0 - result);
```

where *count* is the number of co-rated items between two users, *num* is the number of items in the dataset, and *result* is the calculated Pearson Correlation coefficient.

User-Based Prediction Algorithm: Mahout implements a Weighted Average prediction method similar to the approach described in [1], except that Mahout does *not* take the absolute value of the individual similarities in the denominator, however, it does ensure that the predicted ratings are within the allowable range, e.g., between 1.0 and 5.0.

Item-Based Prediction Algorithm: Mahout implements a Weighted Average prediction method. This approach is similar to the algorithm in [9], except that Mahout does *not* take the absolute value of the individual similarities in the denominator, however, it does ensure that the predicted ratings are within the allowable range, e.g., between 1.0 and 5.0. Also, Mahout does not provide support for neighborhood formation, e.g., similarity thresholding, for item-based prediction.

Accuracy Evaluation calculation: Mahout executes the recommender system evaluator specified at run time (MAE or RMSE) and implements traditional techniques found in [6, 12]. For MAE, this would be,

$$MAE = \frac{\sum_{i=1}^{n} \mid ActualRating_i - PredictedRating_i \mid}{n} \quad (1)$$

where n is the total number of ratings predicted in the test run.

3.2 Making Mahout Fit for Purpose

Through personal email communication with one of the Mahout developers, we were informed that Mahout intended to provide *basic* rating prediction and similarity weighting capabilities for its recommenders and that it would be up to developers to provide more elaborate approaches. Several changes were made to the prediction algorithms and the similarity weighting techniques for both the user-based and item-based recommenders in order to meet our specific requirements and to match the best practices found in the literature, as follows:

Similarity weighting: Defined as Significance Weighting in [5], this consists of the following method:

```
scaleFactor = count/50.0;
if (scaleFactor > 1.0)  scaleFactor = 1.0;
result = scaleFactor * result;
```

where *count* is the number of co-rated items between two users, and *result* is the calculated Pearson Correlation coefficient.

User-user mean-centered prediction: After identifying a neighborhood of similar users, a prediction, as documented in [8, 5, 1], is computed for a target item i and target user u as follows:

$$p_{u,i} = \overline{r_u} + \frac{\sum_{v \epsilon V} sim_{u,v}(r_{v,i} - \overline{r_v})}{\sum_{v \epsilon V} \mid sim_{u,v} \mid} \quad (2)$$

where V is the set of k similar users who have rated item i, $r_{v,i}$ is the rating of those users who have rated item i, $\overline{r_u}$ is the average rating for the target user u over all rated items, $\overline{r_v}$ is the average rating for user v over all co-rated items, and $sim_{u,v}$ is the Pearson correlation coefficient.

Item-item mean-centered prediction: A prediction, as documented in [1], is computed for a target item i and target user u as follows:

$$p_{u,i} = \overline{r_i} + \frac{\sum_{j \epsilon N_u(i)} sim_{i,j}(r_{u,j} - \overline{r_j})}{\sum_{j \epsilon N_u(i)} \mid sim_{i,j} \mid} \tag{3}$$

where $N_u(i)$ is the set of items rated by user u most similar to item i, $r_{u,j}$ is u's rating of item j, $\overline{r_j}$ is the average rating for item j over all users who rated item j, $\overline{r_i}$ is the average rating for target item i, and $sim_{i,j}$ is the Pearson correlation coefficient.

Item-item similarity thresholding: This method was added to Mahout and used in conjunction with the item-item mean-centered prediction described above. Similarity thresholding, as described in [5], defines a level of similarity that is required for two items to be considered similar for purposes of making a recommendation prediction; item-item similarities that are less than the threshold are not used in the prediction calculation.

Coverage and combined accuracy/coverage metric: As suggested in [6], the easiest way to measure coverage is to select a random sample of user-item pairs, ask for a prediction for each pair, and measure the percentage for which a prediction was provided. To calculate coverage, code changes were made to Mahout to provide, for each test run, the total number of rating predictions requested that were unable to be calculated as well as the total of number of rating predictions requested that were actually calculated; the sum of these two numbers is the total number of ratings requested. Coverage was calculated as follows:

$$Coverage = \frac{Total\#RatingsCalculated}{Total\#RatingsRequested} \tag{4}$$

Code changes were also made to calculate a combined accuracy and coverage metric as defined in Section 4.

4. ACCURACY AND COVERAGE METRIC

The metrics selected for this case study, accuracy and coverage, were chosen because they are fundamental to the utility of a recommender system [10, 6]. Although other metrics such as novelty and serendipity can, and should, be used in conjunction with accuracy and coverage, our objective was to evaluate the very basic requirements of a recommender system. Our implementation of coverage, referred to as prediction coverage in [6], measures the percentage of a dataset for which the recommender system is able to provide predictions. High coverage would indicate that the recommender system is able to provide predictions for a large number of items and is considered to be a desirable characteristic of the recommender system [6]. A combination of high accuracy (low error rate) and high coverage are indeed desirable by users and system operators because it improves the utility or usefulness of the system from a user standpoint [10, 6].

What constitutes 'good' accuracy or coverage, however, has not been well defined in the literature: studies such as [10, 4, 5] and many others, endeavor to maximize accuracy (achieve lowest possible value) and/or coverage (achieve highest possible value) and view these metrics on a relative basis, i.e., how much the metric has increased or decreased beyond a baseline value based on empirical results. Furthermore, the interplay between accuracy and coverage, i.e., coverage decreases as a function of accuracy [4, 3], creates a trade-off for recommender system implementers that has been discussed previously but not been developed thoroughly. Inspired by the suggestion in [6] to combine the coverage and accuracy measures to yield an overall "practical accuracy" measure for the recommender system, we developed a straightforward "AC Measure" that combines both accuracy and coverage into a single metric as follows:

$$AC_i = \frac{Accuracy_i}{Coverage_i}, \tag{5}$$

where i indicates the ith trial in an evaluation experiment.

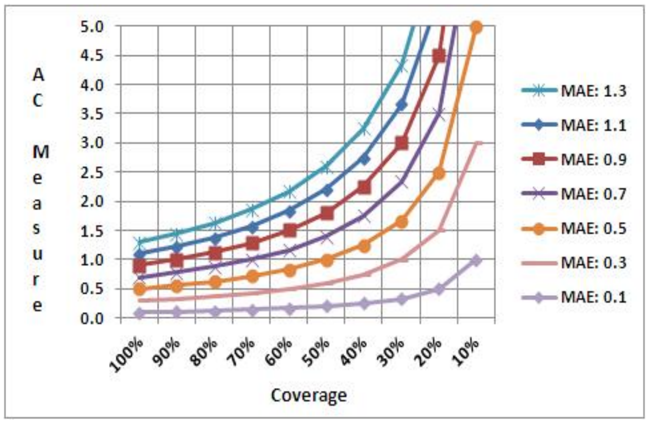

Figure 1: Illustration of the AC Measure

The AC Measure simply adjusts (upward) the Accuracy according to the level of Coverage metrics found in an experimental trial and is agnostic to the accuracy metric used, e.g., MAE or RMSE. Using a family of curves for the Mean Absolute Error (MAE) accuracy metric, Figure 1 illustrates the relationship between accuracy, coverage, and the AC Measure. As an example, following the "$MAE : 0.5$" curve we see that at 100% coverage, the AC Measure is 0.5, and at 10% coverage, the AC Measure has increased to 5. The intuition behind this metric is that when the recommender system is able to provide predictions for a high percentage of items in the dataset, the accuracy metric more closely indicates the level of system performance; conversely, when the coverage is low, the accuracy metric is "penalized" and is adjusted upwards. We believe that the major benefit of the AC Measure is that it formulates a solution for addressing the trade-off between accuracy and coverage and can be used to create a ranked list of results (low to high) from multiple experimental trials to find the best (lowest) AC Measure for each set of test conditions. The simplified visualization of the combined AC Measure shown in Figure 1 is an additional benefit. For our evaluation purposes, the use of a combined metric was ideal in addressing the inherent trade-offs between accuracy and coverage, especially in the cases where accuracy is found to be high when coverage is low; we posit that the AC Measure will also be useful for other researchers performing evaluations using accuracy and coverage.

5. EXPERIMENTAL DESIGN

The objective of this case study was to understand Mahout's baseline collaborative filtering algorithms and evaluate functional changes made to the platform using accuracy and coverage metrics. The main intent of making functional changes to Mahout recommender algorithms was to bring the Mahout algorithms in line with best practices found in the literature. Therefore, the overall hypothesis to be tested in this case study was that the modified algorithms improve Mahout's 'out-of-the-box' prediction accuracy for both user-based and item-based recommenders while maintaining reasonable coverage.

5.1 Datasets and Algorithms

The data used in this study were the MovieLens datasets downloaded from GroupLens Research[8]: the 100K dataset with 100,000 ratings for 1,682 movies and 943 users (referred to as ML100K in this study) and the 10M dataset with 10,000,000 ratings for 10,681 movies and 69,878 users (referred to as ML10M in this study). Ratings provided in these datasets consist of integer values between 1 (did not like) to 5 (liked very much).

For User-based (see §3.1), Mahout uses Pearson Correlation similarity (with and without similarity weighting), Neighborhood formation (similarity thresholding or kNN), and Weighted Average prediction. This was tested against a modified algorithm (see §3.2) consisting of Pearson Correlation similarity (with and without similarity weighting), Neighborhood formation (similarity thresholding or kNN), and Mean-centered prediction. For Item-based (see §3.1), Mahout uses Pearson Correlation similarity (with and without similarity weighting), no Neighborhood formation, and Weighted Average prediction. This was tested against a modified algorithm (see §3.2) consisting of Pearson Correlation similarity (with and without similarity weighting), Neighborhood formation (similarity thresholding), and Mean-centered prediction.

5.1.1 Test Cases

In order to test the overall hypothesis, the following test cases were developed and executed for both user-based and item-based recommenders using the ML100K and ML10M datasets:

1. Mahout Prediction, No weighting
2. Mahout Prediction, Mahout weighted
3. Mahout Prediction, Significance weighted
4. Mean-Centered Prediction, No weighting
5. Mean-Centered Prediction, Mahout weighted
6. Mean-Centered Prediction, Significance weighted

5.1.2 Accuracy and Coverage Metrics

We used Mahout's MAE evaluator to measure the accuracy of the rating predictions. For prediction coverage, we used dataset training data to estimate the rating predictions for the test set; the random sample of user-item pairs in our testing was 30K pairs for ML100K and 25K pairs for ML10M (see §3.2). AC Measures were calculated for all test cases.

5.1.3 Dataset Partitioning

The Mahout evaluator creates holdout [9] partitions according to a set of run-time parameters. For the tests using the ML100K dataset, the training set was 70% of the data, the test set was 30% of the data, and 100% of the user data was used; a total 30K rating predictions from 943 users were requested for each test set. For the tests using the ML10M dataset, the training set was 95% of the data, the test set was 5% of the data, and 5% of the user data was used; a total 25K rating predictions from 3180 users were requested for each test set.

5.1.4 Test Variations

Various similarity thresholds and kNN neighborhood sizes were executed for each test case in order to understand and evaluate the corresponding behavior of the recommenders. For User-based recommender testing, similarity thresholds of 0.0, 0.1, 0.3, 0.5, and 0.7 and kNN neighborhood sizes of 600, 400, 200, 100, 50, 20, 10, 5, and 2 were tested. For Item-based recommender testing, in addition to using no similarity thresholding, similarity thresholds of 0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, and 0.7 were tested.

6. RESULTS AND DISCUSSION

6.1 ML10M Results

Figures 2 and 3 show the results of test cases 1 through 6 for user and item-based algorithms, respectively[10]. The key results of the experiment, for both user-based and item-based algorithms unless otherwise noted, were as follows:

1. MAE for mean-centered prediction with significance weighting is a significant improvement ($p<0.01$) over MAE for Mahout prediction, regardless of weighting, across similarity thresholds (except item-based at similarity threshold of 0.7) and kNN neighborhood sizes (except user-based at kNN of 2, not shown).

2. Mahout similarity weighting does not significantly improve ($p<0.01$) Mahout prediction MAE over prediction with no similarity weighting (except Mahout prediction for user-based and item-based at a similarity threshold of 0.4, not shown). This would indicate that Mahout similarity weighting is not very effective as a weighting technique, especially as compared to significance weighting.

6.2 ML100K Results

The results and trend lines for the ML100K experiment are similar to ML10M. The key results, for both user-based and item-based algorithms unless otherwise noted, were:

1. MAE for mean-centered prediction with significance weighting is a significant improvement ($p<0.01$) over MAE for Mahout prediction, regardless of weighting, across similarity thresholds and kNN neighborhood sizes (except user-based at kNN of 400).

2. Mahout similarity weighting does not significantly improve ($p<0.01$) Mahout prediction MAE over prediction with

[8]http://www.grouplens.org

[9]Holdout is a method that splits a dataset into two parts, a training set and a test set, and the partitioning is performed by randomly selecting some ratings from all, or some, of the users. The selected ratings constitute the test set, while the remaining ones are the training set.

[10]The following curves are superimposed over each other because the values are very similar: MAE results for mean-centered prediction (no weighting and Mahout weighted), MAE results for Mahout prediction (No weighting and Mahout weighted), Coverage results for Mahout prediction and mean-centered prediction (No weighting and Mahout weighted), Coverage results for Mahout prediction and mean-centered prediction (both Significance weighted).

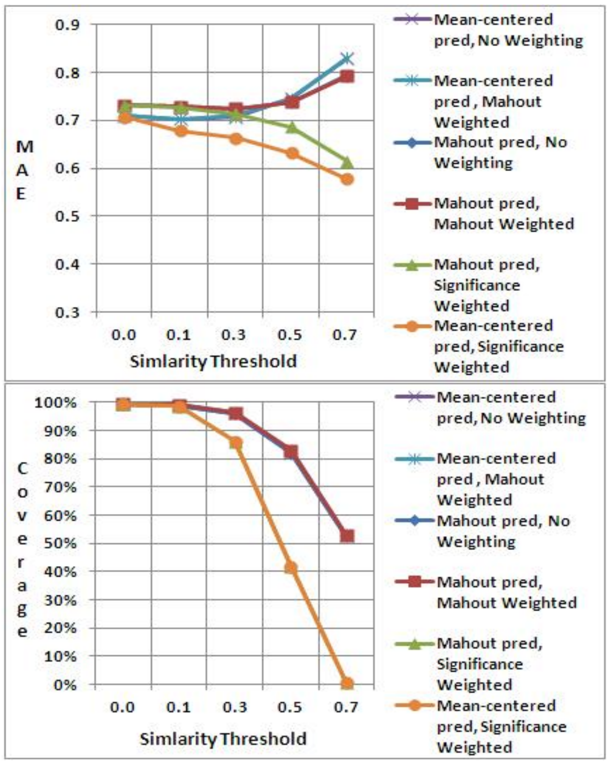

Figure 2: User-based Mahout Recommender Results for ML10M, Test cases 1 through 6

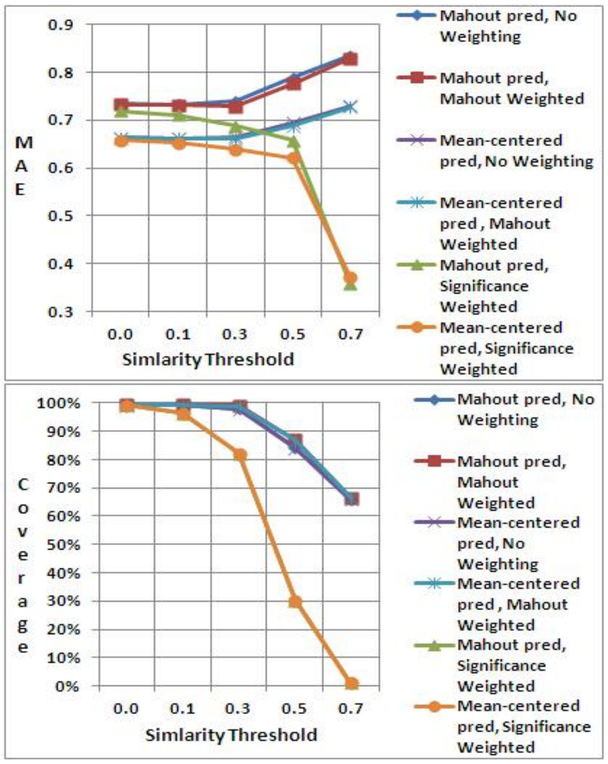

Figure 3: Item-based Mahout Recommender Results for ML10M, Test cases 1 through 6

no similarity weighting (except Mahout prediction for user-based and item-based at a similarity threshold of 0.4).

6.3 Discussion

As hypothesized, results for both of the ML100K and ML10M experiments show significant improvements in MAE using the mean-centered prediction algorithm with significance weighting compared to the Mahout baseline prediction algorithm. However, when coverage is considered, the "best" MAE results may need a second look. Can an MAE of 0.5 or less be considered "good" when the associated coverage is in the single digits? In this case, the recommender system may only be able to provide recommendations to a very small subset of its users and is a situation that must be avoided by system operators. To help address the accuracy vs. coverage trade-off, combined measures such as the AC Measure (Section 4), can help by considering both accuracy and coverage simultaneously. For the ML10M experiment, we determined that the lowest MAE for the User-based algorithm using mean-centered prediction with significance weighting was 0.578 at a similarity threshold of 0.7 and coverage of 0.833%; the AC Measure for this result is calculated as 69.42. Similarly, the lowest MAE for the Item-based algorithm using mean-centered prediction with significance weighting was 0.371 at a similarity threshold of 0.7 and coverage of 1.02%; the AC Measure for this result is calculated as 36.32. In each of these cases, the exceedingly high values for the AC Measure indicate that these results are not very desirable in a recommender system.

Figures 4 and 5 show the AC Measure results for user and item-based algorithms using ML10M, respectively. Rather than show all 30 results for each algorithm (5 similarity thresholds x 2 prediction methods x 3 weighting types), we show only the results with calculated AC Measure values less than 1.0; therefore, the lowest MAE results reported above for user-based and item-based algorithms are clearly beyond the range of this chart. We found that the best combined accuracy/coverage results were found at higher levels of coverage and lower levels of similarity threshold, i.e., the best (lowest) AC Measure for user-based was 0.688 at a similarity threshold of 0.1 and for item-based was 0.665 at a similarity threshold of 0.0, both using mean-centered prediction and significance weighting. We can also see that, with few exceptions, mean-centered prediction is improved over the Mahout prediction for the same similarity weighting and similarity threshold. We observed similar results using the ML100K dataset where the best (lowest) AC Measure for user-based was 0.765 and for item-based was 0.746, both at a similarity threshold of 0.0 and both using mean-centered prediction and significance weighting. These results demonstrate that the "best" MAE may not always be the lowest MAE, especially when coverage is also considered; furthermore, recommender system settings such as similarity weighting and neighborhood size also need to be considered during system evaluation.

Other observations of our experiments that match results reported in [5] and serve to validate our evaluation and increase our confidence in the results are: (a) In general, significance weighting improves prediction MAE, as compared to predictions using Mahout similarity weighting or no similar-

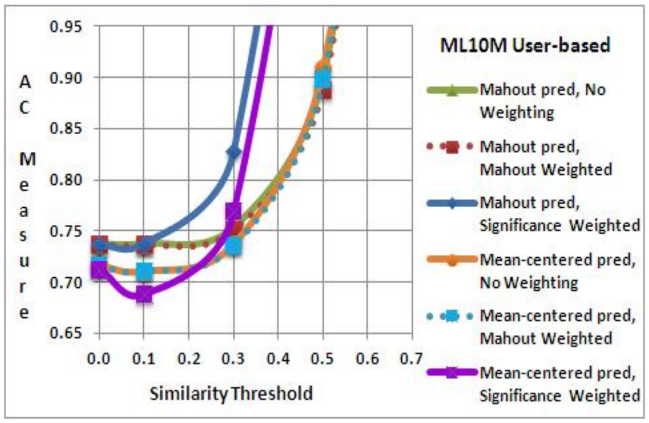

Figure 4: AC Measure for selected User-based results (lower is better)

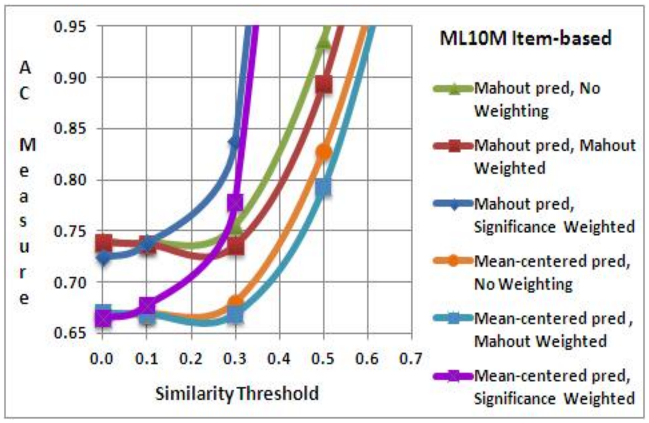

Figure 5: AC Measure for selected Item-based results (lower is better)

ity weighting; (b) As the similarity threshold increases, MAE for mean-centered prediction with significance weighting improves and coverage degrades, whereas MAE and coverage both degrade for Mahout prediction with Mahout weighting; (c) Coverage decreases as neighborhood size decreases.

7. CONCLUSION

Our case study of Mahout as a recommender system platform highlights evaluation considerations for developers and also shows how straightforward functional enhancements improves the performance of the baseline platform. We evaluated our changes against current Mahout functionality using accuracy and coverage metrics not only to assess baseline results, but also to provide a view of the trade-offs between accuracy and coverage resulting from using different recommender algorithms. We reported cases where the lowest MAE accuracy results were not necessarily always the 'best' when coverage results were also considered, and we instrumented Mahout for a combined accuracy and coverage metric (AC Measure) to evaluate these trade-offs more directly. We believe that this case study will provide useful guidance in using Mahout as a recommender platform, and that our combined measure will prove useful in evaluating algorithm changes for the inherent trade-offs between accuracy and coverage.

8. REFERENCES

[1] C. Desrosiers and G. Karypis. A comprehensive survey of neighborhood-based recommendations methods. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*. Springer, 2011.

[2] M. D. Ekstrand, M. Ludwig, J. A. Konnstan, and J. T. Riedl. Rethinking the recommender research ecosystem: Reproducibility, openness, and lenskit. In *Proceedings of the 5th ACM Recommender Systems Conference (RecSys '11)*, October 2011.

[3] M. Ge, C. Delgado-Battenfeld, and D. Jannach. Beyond accuracy: Evaluating recommender systems by coverage and serendipity. In *Proceedings of the 4th ACM Recommender Systems Conference (RecSys '10)*, September 2010.

[4] N. Good, J. B. Schafer, J. A. Konstan, A. Borchers, B. Sarwar, J. Herlocker, and J. Riedl. Combining collaborative filtering with personal agents for better recommendations. In *Proceedings of the 16th National Conference on Artificial Intelligence (AAAI-99)*, July 1999.

[5] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the ACM SIGIR Conference*, 1999.

[6] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53, 2004.

[7] S. Mcnee, J. Riedl, and J. Konstan. Accurate is not always good: How accuracy metrics have hurt recommender systems. In *Proceedings of the Conference on Human Factors in Computing Systems(CHI 2006)*, April 2006.

[8] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. GroupLens: an open architecture for collaborative filtering of netnews. In *Proceedings of the ACM CSCW Conference*, 1994.

[9] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the World Wide Web Conference*, 2001.

[10] B. M. Sarwar, J. A. Konstan, A. Borchers, J. Herlocker, B. Miller, and J. Riedl. Using filtering agents to improve prediction quality in the grouplens research collaborative filtering system. In *Proceedings of the ACM 1998 Conference on Computer Supported Cooperative Work (CSCW '98)*, November 1998.

[11] C. E. Seminario and D. C. Wilson. Robustness and accuracy tradeoffs for recommender systems under attack. In *Proceedings of the 25th Florida Artificial Intelligence Research Society Conference (FLAIRS-25)*, May 2012.

[12] G. Shani and A. Gunawardana. Evaluating recommendation systems. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*. Springer, 2011.