On the Comparative Accuracy of Tamber and Competitors

The Tamber Team

1 Introduction

In the current generation of web and mobile applications, content is becoming increasingly abundant in a phenomenon commonly referred to as Information Overload [1]. The large sea of available content presents a challenge that traditional curated publication channels have not had to face. Social networks, such as Facebook, have long employed selection algorithms to maintain the quality of the user experience as they scale. As they have continued to grow, they have turned to increasingly elaborate ranking algorithms, which add supplementary models or modify existing ones (e.g. creating more finely tuned ways of balancing the comments on a post against the post's age).

Meanwhile, recommender systems, which aim to match users with content algorithmically, have only been implemented successfully in online advertising and a handful of popular web platforms – namely Amazon, Netflix, and Pandora. However, Amazon, which sells over 300 million items, has long suffered in recommendation quality, while Netflix and Pandora's recommender systems are highly customized for the narrow band of content that they feature. These limitations are both the result of deeper problems of traditional recommender systems: their inability to process large quantities of data in real time, and, more importantly, their low accuracy in comparison to expert curators. For Amazon, the massive scale of their data requires them to use coarser evaluations which can run reasonably quickly but tend to either underfit and feel overly general (e.g. category-level recommendations), or overfit and feel very linear (e.g. "users who bought this also bought..."). Netflix overcomes these pitfalls by leveraging their exceptionally dense dataset of over 60 million worldwide users reviewing about 10,000 content items and a dedicated team of engineers who tailor recommendation models to fit film and television content. And Pandora, which is highly accurate, offloads the evaluation of content to a staff of experts who manually process each song.

While these cost-intensive solutions have proven successful, they do not address the fundamental problems in mainstream recommendation technology, and are not reasonable approaches for most technology companies. The open source platform Apache Mahout, which uses similar underlying methods as Amazon and Netflix, provides the foundation for many of the most popular recommendation systems today, including Foursquare, LinkedIn, and Twitter [2]. However, these recommendation strategies face the same fundamental limitations.

Our recommender system, Tamber, solves these problems by combining more sophisticated and holistic prediction algorithms with systems that understand culture and

taste more deeply. In this paper, we will compare Tamber's performance to that of Apache Mahout.

2 Accuracy Metrics

There are many prominent techniques found in literature describing how to evaluate the performance of a recommendation system. While the most realistic test for predicting an engine's performance is to design an extensive A/B test, this approach is intensely resource-consuming and not always viable; for example, attempting to test too many recommendation systems or having too small of a user base will make the user feedback less reliable. Therefore, having a sizeable user behavior dataset along with a modular test that can automatically evaluate multiple different recommendation engines is preferred.

One of the main challenges in designing such tests is quantizing exactly how "good" a recommendation is. Traditionally, these tests are designed to predict how a user would rate a certain item. Large datasets of user ratings are prepared beforehand and split into two pieces: one piece is used to train the recommendation engine and the other is used to score the engine's performance based on how closely the recommendation engine can reproduce the users' actual ratings. The Root Mean Square test is one notable canonical variant of such a test.

However, we believe that such ratings-based tests do not accurately reflect how well an actual end-user will receive the engine's recommendations. This is because end-users often do not rate or review items. Instead most users interact with content on the internet through implicit behaviors; these implicit behaviors might include clicking a link, liking a Facebook post, or buying a product. When user ratings are available, these behaviors are of course valuable; however basing an entire accuracy test on just this small subset of user behaviors means that one can miss the subtleties of more complex user behaviors.

To avoid the limitations of ratings-based testing, we evaluated Tamber using metrics that measure recommendation "hits": recommendations with which the user later showed some positive interaction. In particular, instead of attempting to predict how a user would rate an item, Tamber produces a list of recommendations for each user ordered by the likelihood of detecting the user positively interacting with the recommended item. Each time the test data showed that the user did positively interact with one of the items on their list of recommended items, we recorded a "true positive"; each recommendation that the user negatively interacted with was recorded a "false positive"; lastly, each item that the user interacted positively with that was not recommended was recorded as a "false negative." By using these metrics, each recommendation engine can be evaluated based its frequency of bad recommendations, good recommendations, and missed potential recommendations. This kind of testing is more meaningful with the final use-case in mind: recommendation engines are often integrated by showing a user multiple pieces of potentially interesting content while tracking user interactions and learning from these behaviors.

3 Benchmarking Competitors

Our first test was to validate our implementation of Apache Mahout. To do this, we attempted to reproduce the Root Mean Square Error values of Said and Bellogin 2014 [3]. Following their methods, we trained Mahout on an 80%-20% split of the Movie Lens 100K dataset using an item-based Pearson similarity recommender. We were able to reproduce the values in their benchmark table confirming the validity of our Mahout setup.

In addition, we built a third recommendation engine that would recommend random items to each user. This provided a performance baseline that we compared to Mahout and Tamber. As expected, Mahout and Tamber both significantly outperformed the random recommender.

4 Methodology

We tested the performance of Tamber and Mahout on several different sets of data, which varied in length, data density (the number of behaviors per user and per item), and total behaviors tracked. The four main datasets we used were the MovieLens variants (100k, 1M, 10M and 20M), Chicago restaurant reviews from 1996-1999, the Book-Crossing dataset, and user-to-user ratings from an online dating service. For each dataset, we created five different randomized splits of the data: 80% would be used to train the engine and 20% saved for testing. We configured Mahout to be a straightforward item similarity recommender; the item similarity score would be generated by a log-likelihood model based on each item's nearest 100 neighbors. After training Tamber and Mahout on identical datasets, each engine generated 20 recommendations for each user and saved these in a database. A python script would automatically iterate through every user in the test data and check to see if there was any recorded interaction with recommended items and whether the interaction was positive or negative. We tracked three key metrics: "true positives" (the number of recommendations for each user for which a positive interaction was recorded), "false positives" (the number of recommendations per user for which a negative interaction was recorded" and "false negatives" (the number of items per user that weren't recommended but had a positive interaction recorded). These numbers were totalled over every user and used to calculate each engine's precision, recall and F1 score; these quantities are defined as:

$$\begin{aligned}
&\text{Precision} = t_p/(t_p + f_p) \\
&\text{Recall} = t_p/(t_p + f_n) \\
&\text{F1-score} = 2\ \text{Precision} \times \text{Recall}/(\text{Precision} + \text{Recall})
\end{aligned}$$

Heuristically, an engine's precision score represents the probability that a given recommendation in the top 10 will be received favorably by the user. The recall score represents what percentage of potentially good recommendations will actually be rec-

ommended by the engine. Typically recommendation systems will optimize either precision or recall. Unfortunately, these two metrics are typically inversely related to each other: having a higher recall means that the recommendation system will attempt to cast a wider net and thus lower precision. In the use case of user recommendations, it is not clear whether to value precision or recall more; however, it turns out that this is not important to us because Tamber scores an order of magnitude higher in all of our relevant metrics.

5 Results

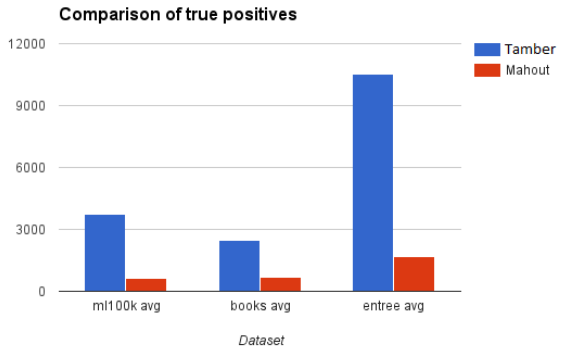

Figure 1: Tamber and Mahout's true positive scores

Over all of our datasets, Tamber scored better than Mahout by a dramatic margin (for a complete summary of results, see table 1 on page 8). For the MovieLens 100k 80%-20% split variants, a random suggestion engine averages about 200 true positives. Mahout improves upon this by a factor of 3 averaging 633 true positives while Tamber averages 3725 true positives - an improvement over Mahout by a factor of 5. While Tamber still outperforms Mahout over the smaller datasets (the ml100k, books crossing and restaurants datasets), the margin of outperformance is wider in the large datasets (the ml1m, ml10m, ml20m and dating datasets). For the MovieLens 1 million, 10 million and 20 million datasets, Tamber averaged 23637 true positives, 273208 true positives and 502162 true positives respectively. In comparison, Mahout was only able to score 3182, 13669, and 19939 true positives (compare fig 2 with fig 1). It is no surprise that the raw count of true positives increase as the size of the dataset increases; the more valuable metric is the precision scores. Across all MovieLens datasets, Tamber maintains a precision of about .19 indicating that users positively interacted with nearly 20% of their recommendations. In contrast Mahout's precision fluctuated between 0.01-0.02, worse than Tamber's precision by an entire order of magnitude.

While Tamber outperforms Mahout across all datasets, Tamber outperforms Mahout by a wider margin on larger datasets. For example, on our dataset of books Tamber was only able to outperform Mahout's precision by 10% instead of an entire order of magnitude. However, this dataset appears to be anomalous: for all other datasets, Tamber maintains its outperformance by a factor of 10. Trained on the online dating dataset, Tamber was able to achieve a precision of around 0.1 whereas Mahout's precision is around 0.01 (see fig 3). Tamber's recall is similarly also ten times higher than Mahout's

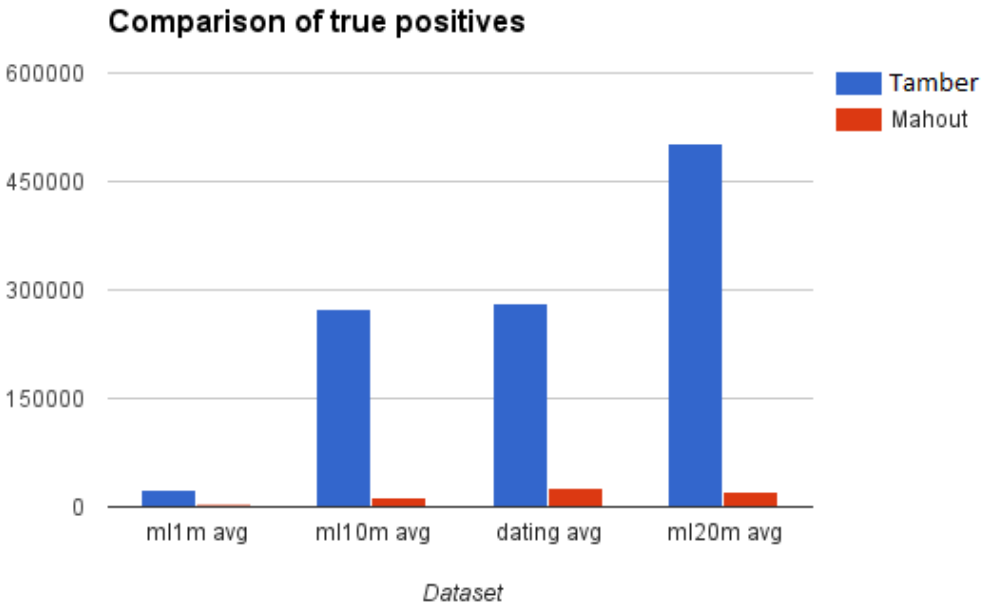

Figure 2: Comparison of Tamber and Mahout's true positive scores; Tamber's recommendations scale better with dataset size

recall.

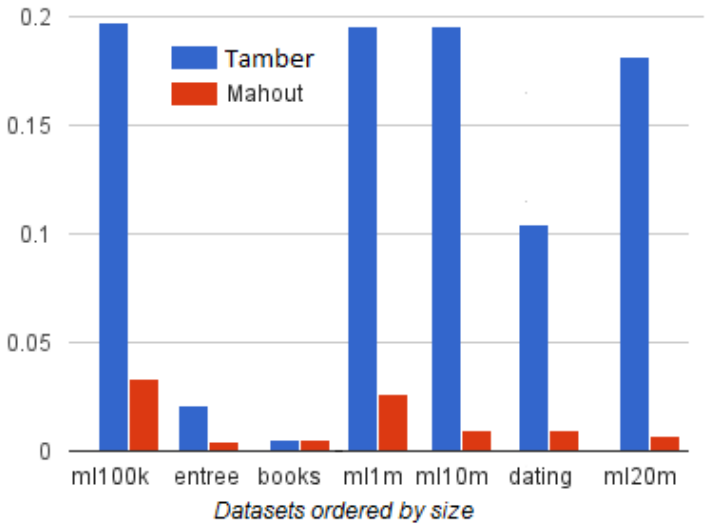

Figure 3: Comparison of Tamber and Mahout's precisions (true positives normalized over total recommendations)

Tamber's most significant outperformance of Mahout was on the Chicago restaurant reviews dataset. This dataset is actually considerably more sparse than the online dating and MovieLens datasets. However, the restaurant reviews data had the benefit of tracking multiple implicit user behaviors; instead of rating restaurants on a numerical scale, users offered a range of different responses such as "the restaurant is too expensive" or "the restaurant is too noisy." Because Tamber was designed with tracking multiple user behaviors in mind, it is no surprise that Tamber outperformed Mahout by the widest margin on this dataset. Instead of scoring the usual 10 times higher than Mahout, Tamber scored a precision of 0.02 and a recall of .78, 50 times higher than Mahout's precision score of 0.004 and recall score of .12.

In addition to offering higher quality suggestions across the board, Tamber also generates recommendations much faster than Mahout. In fact, after the initial training phase, Tamber is able to generate recommendations in mere milliseconds. Mahout, in comparison, takes hundreds of milliseconds to produce the same number of lower quality results.

6 Conclusion

The recommendation system Tamber offers numerous advantages over Mahout, Tamber's most widespread competitor. Recommendation engines built on Mahout suffer from slow speeds and worse accuracy. Crucially, platforms built on Mahout or its underlying methods cannot deal with the fundamental issues facing recommendation technology: Mahout has no answer for users who demand both high precision recommendations and quick, scalable calculations. Tamber, on the other hand, is specifically engineered to avoid these problems. Moreover, unlike platforms such as Netflix and Pandora, Tamber is flexible enough to recommend a broad spectrum of content types. In the rapidly growing world of internet content, users should be increasingly less tolerant of mediocre recommendations. Tamber offers an easy and commercially available solution; instead of committing large in house teams of developers to build custom recommendation Engines, businesses can readily use Tamber "out-of-the-box" and begin generating recommendations while expending only minimal resources.

References

[1] Koltay, T. "Information Overload, Information Architecture and Digital Literacy." *Bulletin of the American Society for Information Science and Technology,* 38(1) (2011): 33-35. Retrieved June 17, 2015, from https://www.asis.org/Bulletin/Oct-11/OctNov11_Koltay.pdf

[2] "Powered by Mahout." *Mahout.* The Apache Software Foundation, 2014. Web. Retrieved June 17, 2015, from https://mahout.apache.org/general/powered-by-mahout.html

[3] Said, A. and Bellogin, A. "Comparative Recommender System Evaluation: Benchmarking Recommendation Frameworks." *RecSys '14 Proceedings of the 8th ACM Conference on Recommender Systems.* New York: ACM, 2014. 129-136. Retrieved June 17, 2015, from http://ir.ii.uam.es/∼alejandro/2014/recsys-benchmarking.pdf

7 Appendix

Table 1: Summary of results

Dataset/Engine	Precision	Recall	F1 Score
books crossing			
Tamber	0.005577202	0.03012048	0.009411694
Mahout	0.005072086	0.00858649	0.006377116
Random	7.61E-06	7.74E-05	1.39E-05
restaurants			
Tamber	0.0212743	0.7822684	0.04142206
Mahout	0.004696476	0.1236644	0.00904928
Random	0.000790423	0.03090528	0.001541422
ml-100k			
Tamber	0.1977708	0.2260234	0.2109548
Mahout	0.03360502	0.03841006	0.03584716
Random	0.01092522	0.01248696	0.01165398
ml-1m			
Tamber	0.1958172	0.1412678	0.1641288
Mahout	0.02636104	0.01901894	0.02209604
Random	0.007458814	0.005381476	0.006252102
ml-10m			
Tamber	0.195708	0.165707	0.1794624
Mahout	0.009791282	0.008290834	0.008978808
Random	0.002240844	0.001897444	0.002054898
ml-20m			
Tamber	0.1815184	0.1522744	0.1656156
Mahout	0.007206992	0.006046434	0.006575898
Random	0.0009236988	0.0007749592	0.000842817
dating			
Tamber	0.1045906	0.1453942	0.1216624
Mahout	0.009658668	0.01340802	0.0112286
Random	8.41E-05	0.0001169636	9.79E-05