Movie and Book Recommender

Chris Barnett
Computer Science 142
Machine Learning and Data Mining
University of California, Santa Cruz
cjbarnet@ucsc.edu

December 1, 2011

Abstract

Collaborative Filtering is a technique for generating predictions about what items a user will like based on their ratings of other items and the ratings of other users. Many collaborative filtering techniques have been developed over the years, each with different pros and cons. In this project, I have compared the effectiveness of my own user-based collaborative filtering algorithm, based on a Dot-Product similarity function, with another user-based algorithm and with the SlopeOne algorithm. I used two different evaluation metrics: Mean Absolute Error and Kendall's Tau Correlation Coefficient to evaluate the effectiveness of each algorithm. I found that, while SlopeOne was the best performing algorithm in almost all scenarios, it was equaled by the Dot-Product algorithm when evaluated by the Kendall's Tau Correlation Coefficient metric.

1. INTRODUCTION

The World Wide Web provides people all around the world access to a historically unprecedented quantity of information. There is so much information available, in fact, that just having access to all of it would be of limited utility without some way to find the information that is most valuable to you. This problem of matching appropriate information with information consumers has been tackled from a number of perspectives. One incarnation of this challenge is that of search, where information consumers desire a specific piece of information such as the answer to a question or a particular media file and need a tool to locate it in the haystack that is the web. Search Engines like Google attempt to satisfy this need. Another problem users of the web face is to discover valuable nuggets of information that were previously unknown to them. For example, someone may wish to discover interesting news articles or a good film to watch. This is the domain of Recommender Systems.

Recommendation is a fundamentally user-centric problem since the information that is valuable to one person is not necessarily of much value to another. There are two fundamental approaches to finding appropriate recommendations for a user: Collaborative Filtering and Content-Based Filtering. Collaborative Filtering finds relevant recommendations for a user based on the preferences expressed by other users of the system. This approach is very popular and is used by many media and e-commerce sites. It is the basis of Amazon's famous "users who bought x also bought y" recommendation system.

Content-Based Filtering tries to identify items that have similar characteristics to items the user has already expressed an interest in. This technique can work well for items that are easily parsed by a machine, such as text documents which can be described by a set of keywords. However, for items like movies, where most of the important features to humans are not currently understandable by machines, content-based filtering is severely limited in usefulness [Shardanand and Maes 1995]. Content-Based filtering is also unable to evaluate the value of an item to a user, only its relevance, and they cannot be used to find serendipitous items that might be of interest to a user, but which are not similar to those the user has already expressed an interest in.

In this work, I will focus on Collaborative Filtering techniques. These come in two broad families: Memory -Based and Model-Based. Memory-Based techniques were among the first to evolve and include item-based and user-based neighborhood approaches. These methods use the entire ratings matrix to a calculate a similarity value between pairs of users or items. Predictions are then made by using a weighted (by similarity) average over ratings.

Model-Based algorithms try to construct a model which approximates the underlying behavior of the users. The model is then used to predict user behavior for items they have not explicitly rated yet. Examples of model-based algorithms include Singular Value Decomposition (SVD) [Sarwar et al. 2000], Neural Networks [Billsus and Pazzani 1998], Clustering [Ungar and Poster 1998] and Bayes Networks [Breese et al. 1998].

For this project, I have analyzed three Memory-Based algorithms, including my own user-based technique. I compare their relative effectiveness over two different datasets using two different evaluation metrics.

2. RELATED WORK

Cacheda et al. [2011] performed an extensive evaluation of the most popular and most effective currently known collaborative filtering algorithms over the MovieLens and Netflix datasets. They included an algorithm of their own creation in the mix, which is based on the tendencies of users and items to rate or be rated higher or lower than average. This algorithm turned out to be competitive with the best performing algorithms on most metrics. The other best performing algorithms in their study, particularly under sparse-data conditions, were model-based algorithms such as Singular Value Decomposition (SVD) and variants, and SlopeOne. Clustering and Memory-Based approaches including User-Based and Item-Based algorithms generally performed worse in their study, particularly under sparsity conditions. This lead them to hypothesize that it is difficult to find groups of similar users in practice which is why algorithms like clustering and User-Based neighborhood approaches that base their predictions on trying to find groups of similar users are limited in their effectiveness. In my project, I found that my Dot-Product similarity algorithm performed better than Pearson Correlation in all cases and I discuss later why I believe this may be related to Cacheda et al [2011]'s suggestion that finding similar users may be a more difficult problem than has commonly been assumed.

McNee et al. [2006] discuss the limitations of accuracy-based metrics for evaluating the quality of recommender algorithms. They draw attention to the disconnect between what users actually want from a recommender system and what is measured by standard accuracy metrics like Mean Absolute Error (MAE) or Root Mean Squared Error (RMSE). Users are generally looking for what McNee et al. [2006] call "serendipitous" recommendations, which are recommendations for unexpected items that turn out to be valuable to the user. Traditional accuracy metrics however, tend to reward algorithms that recommend items that are very similar to those the user has already expressed a preference for. This can lead to recommendation lists that are full of items that are already well-known to the user and which are therefore not very useful to them. McNee et al. [2006] give an example of a recommender system for which "once a user rated one Star Trek movie she would only receive recommendations for more Star Trek movies". Clearly, if a user has already expressed a preference for a Star Trek movie, they already know that they can find more of the same in other Star Trek movies. Users are more likely to appreciate recommendations for other movies that they don't yet know about which they are likely to enjoy given that they like Star Trek.

The problem of finding appropriate evaluation metrics for recommender algorithms is a well-recognized problem in the field [Su and Khoshgoftaar 2009]. Celma and Herrera [2008] and others have proposed more user-centric approaches to evaluation which measure the success of an algorithm based on whether it actually produces pleasing results for users. Unfortunately, it is extremely difficult and expensive to involve real users in evaluation so less ideal forms of evaluation like MAE are most commonly used in practice.

I have used MAE for most of my evaluations, but, for several experiments, I also used Kendall's Tau Correlation Coefficient to measure the correlation between a users' ordering of their rated items and the ordering predicted by the algorithm. The Kendall Tau evaluation metric still suffers from the same limitation that cannot test how much a user will actually appreciate results, but it places more emphasis on an algorithm's ability to roughly separate the items a user will like from those he will dislike and less on its ability to predict the specific rating the user will give it.

3. EXPERIMENTS

3.1. Datasets

I found two candidate datasets for this project: the Book Crossing dataset (http://www.informatik.uni-freiburg.de/~cziegler/BX/), which consists of 92,107 users, 270,170 books and 1,031,175 ratings, and the MovieLens dataset (http://www.grouplens.org/node/73), which has 100,000 ratings divided among 1000 users and 1700 movies. In both cases, the data consists of a set of ratings where each rating consists of 3 fields: user, item, numeric rating. This represents a single rating of an item by a user in a finite range such as 1-5.

I had initially planned to use only the Book Crossing dataset, for two main reasons. Firstly, its ratings were based on a 10-point scale rather than the more common 5-point scale and I was interested in discovering which algorithms produce the best recommendations under conditions of highly granular ratings. The second reason the Book Crossing dataset interested me is that I hypothesized that user-based techniques might work better for preferences about books compared with preferences about movies because I intuited that people tend to focus on particular genres and subjects in their reading more than they stick to a single kind of movie.

For several reasons, I ended up deciding to evaluate my selected algorithms over both datasets. One of the motivations for this decision was that it would allow me to compare the relative effectiveness of the algorithms on each dataset and possibly find some evidence to support or reject my above hypotheses.

There was also a more practical reason for introducing the MovieLens dataset: it is denser, meaning there are more ratings per user and per item on average and it was also easier to get the data into the Mahout machine learning framework for evaluation. The Book Crossing dataset required a lot of reformatting before I could read it into my program. I had to remove all the zero ratings, because they represented a binary preference rather than an actual rating of zero and I only wanted to use the numeric ratings. I also had to convert the item identifiers from text ISBNs to unique integers, which I did with MySQL.

The Book-Crossing dataset actually turned out to be so sparse that some algorithms would fail to produce predictions too often to be able to reliably evaluate them. For this reason I decided to manually remove from the dataset the least prolific users (those that had rated less than 6 items) as well as all items that had only been rated once. The raw dataset contained 92,107 users, 270,170 books and 1,031,175 ratings. After the culling, 10,736 users, 40,183 books and 203,572 ratings remained.

For reasons relating to my Dot-Product algorithm which I will explain later, I transformed each dataset into a centered and non-centered version. The centered version shifts the rating values such that a neutral rating is at zero. I also roughly normalized the rating values such that the minimum rating is roughly -1 and the maximum roughly 1. Specifically, the MovieLens dataset was transformed from [1, 5] to [-1, 1] and [0.2, 1.0] and the Book-Crossing dataset was transformed from [1, 10] to [-0.9, 0.9] and [0.1, 1.0].

3.2. Methodology

I initially planned to use the Lenskit collaborative filtering framework to write and evaluate my algorithms, however, I had trouble figuring out how its evaluation framework worked and after examining the Taste collaborative filtering framework within Apache Mahout, I decided I liked and understood it better, so I decided to switch to that.

Apache Mahout's Taste framework is modular and allows developers to plug in their own implementations of many individual components of recommendation and evaluation algorithms. I wrote my own evaluation algorithm and my own user-based recommender algorithm by inheriting from fundamental abstract classes. I also wrote some basic scaffolding to make batch evaluations with varying parameters easier. I wrote my code in a mix of java and scala.

For each experiment, the dataset was randomly split by users into a training set (80%) and a test set (20%). The test set was further divided into two parts by separating a proportion of each user's ratings into the training set and leaving the rest in the test set to be evaluated. This second step is necessary because collaborative filtering algorithms rely on past ratings by users in order to predict how they would rate new items.

In order to evaluate an algorithm, it was provided with the training set to use as input to its prediction algorithm. It was then asked to predict user ratings for each rating held out in the test set. The predicted ratings were paired with actual ratings, grouped by user and sent to one of the evaluation algorithms to generate an evaluation score for the algorithm. Every algorithm was evaluated 5 times and the 5 evaluation scores averaged to form the scores that are shown in the results.

3.3. Algorithms

I have evaluated 3 algorithms for this project

- SlopeOne
- User-Based with Pearson Correlation Similarity function
- User-Based with Dot-Product Similarity function

I also attempted to include an Item Based algorithm, also with a Pearson Correlation Similarity function, but I encountered technical trouble getting that to work and ran out of time to debug the problem so I decided to omit it.

The SlopeOne algorithm has been explained extensively by others [Lemire and Maclachlan 2005] so I will not repeat it here.

The User-Based algorithm works as follows: calculate the similarity between the active user and every other user; find a neighborhood of the k most similar users to the active user and use a weighted average of those k users' ratings to predict the rating the active user would give to an item. The weighting on the ratings of similar users is usually the similarity value, and this is what is used in both User-Based algorithms in this experiment.

I have used a k-value of 100 for both User-Based algorithms in all experiments. I found that this amount was roughly optimal by trial and error and it was also a number suggested in the Cacheda et al. [2011] paper as yielding relatively better performance than lower values of k. See the Cacheda et al. [2011] paper (pg. 25) for a suggested explanation of why larger k values perform better.

The similarity function to determine the similarity between users can be calculated with a number of different methods. One of the most common similarity functions finds the Pearson Correlation between two users' rating vectors where only the ratings on items rated by both users are included. I have used this similarity function as a baseline to compare with my own similarity function, which I call Dot-Product similarity, because, in the simplest case, it finds the dot-product between two user's rating vectors. In order to add more complexity to the model, I experimented with adding three additional parameters to the function. I call these the Rating Scale (λ) which effectively multiplies all ratings by a constant λ; the Pre Sum Scaling (α) which scales each product of two ratings to the power α; and the Post Sum Scaling (β) which scales the similarity between two users by the power β. The resultant similarity function between two users is defined as:

$$S_{u,k} = \left[\sum_{i}^{N_{u,k}} (\lambda R_{u,i} \, \lambda R_{k,i})^{\alpha} \right]^{\beta}$$

Where:

$S_{u,k}$ = The similarity factor between User u and User k.
$N_{u,k}$ = The number of items rated by both User u and User k.
$R_{u,i}$ = User u's rating of Item i (or zero if User u has not rated Item i)
$R_{k,i}$ = User k's rating of Item i
λ = A linear scaling factor on Ratings that needs to be learned.
α = A scaling factor that needs to be learned. It acts as an indicator of the relative importance of two users' level of agreement over a single book on their overall similarity factor.
β = Another scaling factor that needs to be learned. It affects the degree to which ratings are scaled by their user's similarity with the target user.

The reason I normalized the rating scales of the two datasets to roughly [-1,1] for the centered version and [0,1] for the non-centered version was so that the scaling factor λ would be comparable across both datasets. As I will show shortly, however, scaling the ratings turns out to have no effect on the dot-product algorithm (which turns out to make sense mathematically) and so the normalization of the rating values was actually superfluous.

Dot-Product similarity deviates from the typical approach to similarity functions because the similarity values it produces are not normalized. User similarity increases unboundedly as the number of commonly rated items increases. Pairs of ratings that are both close to the same extreme contribute more to user similarity than rating pairs that are both close to zero or of opposite sign. This behavior will mean that, when rating values are centered such that a neutral rating is at zero, similarity between users will be increased when both users really like an item or when both users really dislike an item and similarity will be decreased if users have opposite preferences for an item. If rating values are not centered then only a common positive preference for an item will count strongly towards similarity. I hypothesized that the Dot-Product algorithm would perform better when the rating values are centered because its semantics seem to more closely match my intuition of what constitutes similar preferences. In order to test this hypothesis, I ran some experiments on both the centered and non-centered versions of each dataset.

3.4. Evaluation Metrics

For most of my experiments I used Mean Absolute Error (MAE) as my evaluation metric. This calculates the mean of all the absolute differences between predicted ratings and the actual ratings in the test data.

MAE places a lot of emphasis on an algorithm being able to predict very accurately the precise rating a user will give to an item. In practice, this does not directly match the functionally that is desired by users from a recommender system. Users want to be recommended items they find valuable and not items they hate, but they probably don't particularly care whether the recommender can anticipate exactly what rating they will give an arbitrary item. In fact, they themselves may rate the item differently under different circumstances. An algorithm that is very good at distinguishing generally between items a user likes and those she does not, but which doesn't accurately predict the actual values of the ratings would be unfairly penalized by the MAE algorithm.

A better measure of recommender performance would be its ability to put items in roughly the same order the user herself would, regardless of whether it gets the actual values of ratings exactly right. This is what the Kendall Tau Correlation Coefficient evaluation metric does. It finds the correlation between the order in which the user ranks a set of items and the order predicted by the recommender. I have used this metric where I could, however it requires multiple items per user in the test data to be able to work. The Book Crossing dataset turned out to be too sparse to support the Kendall Tau metric so I was only able to use the MAE metric with that dataset.

4. RESULTS

In order to find the best parameters for the Dot-Product similarity function, I evaluated it with the MovieLens-centered dataset using a range of different values for each parameter. For these experiments, 20% of the ratings of 20% of the users were included in the test set with all other ratings being used in the training set.

The results are shown below.

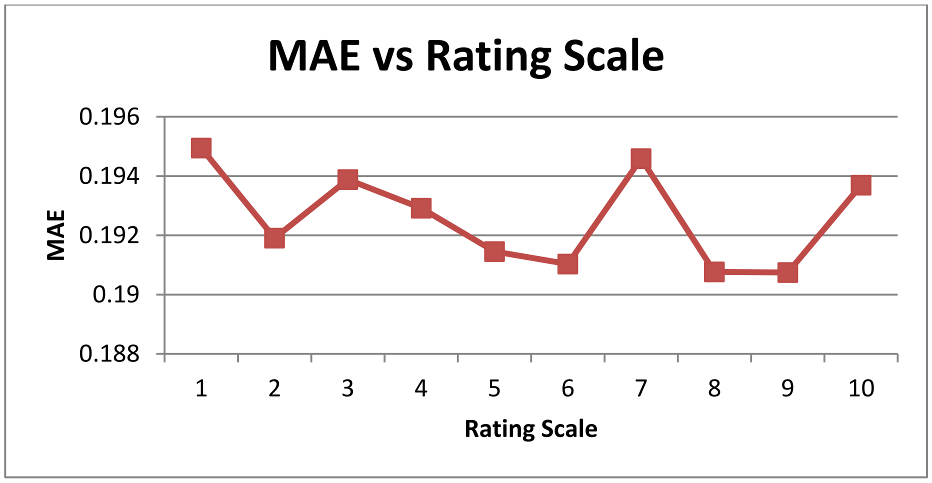

Figure 1 – Accuracy of Dot-Product similarity for various values of the Rating Scale parameter. Pre- and Post-Sum Scaling are both 1.0.

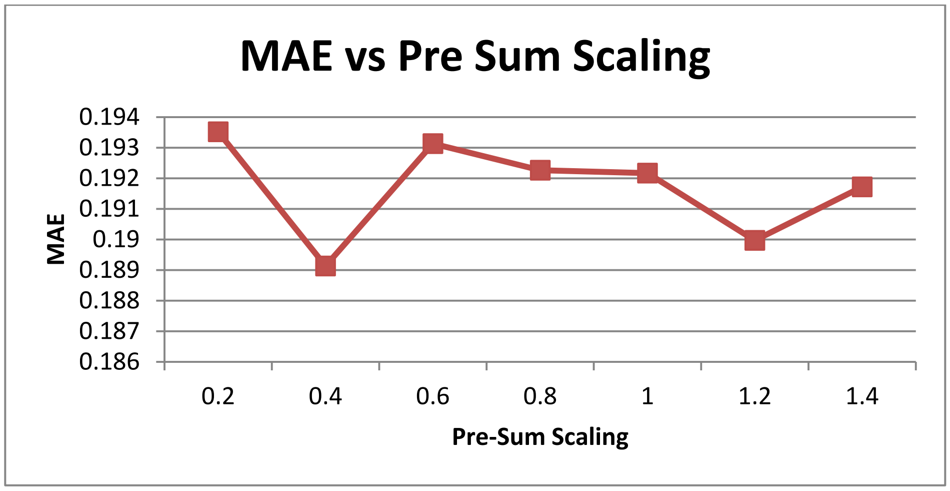

Figure 2 - Accuracy of Dot-Product similarity for various values of the Pre-Sum Scaling parameter. Rating Scale and Post-Sum Scaling are both 1.0

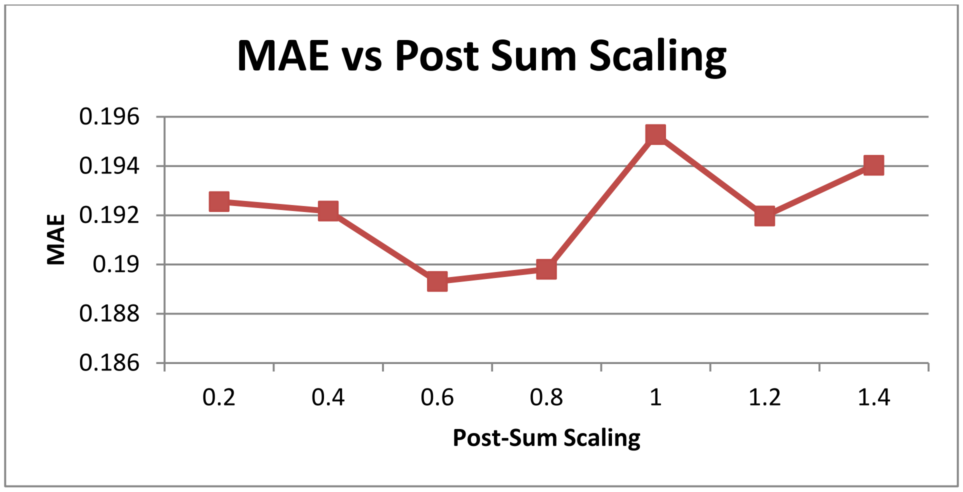

Figure 3 - Accuracy of Dot-Product similarity for various values of the Post-Sum Scaling parameter. Rating Scale and Pre-Sum Scaling are both 1.0

These results show that, for all three parameters, any effect they might have on accuracy is far less than the noise inherent in the experiment. Their effect can therefore be assumed to be insignificant or non-existent. Upon discovering this, I re-examined the mathematical significance of each parameter and discovered that, for the Rating Scale parameter at least, its effect should be expected to be zero. The λ parameter in the Dot-Product similarity function acts as a constant multiplier of the whole function. Since the similarity value is used only as a weight, and only relative weight matters, multiplying all similarity values by a constant will not change the predictions of the algorithm and will thus have no effect on its effectiveness.

The insignificance of the Pre- and Post-Sum Scaling parameters is somewhat harder to explain. I can only surmise that the general concept of attributing similarity to users for every item whose rating they roughly agree on is much more important than specifically how much similarity should be attributed to a particular combination of ratings.

Since none of the parameters of the Dot-Product algorithm appeared to have any noticeable effect I used the pure Dot-Product algorithm (where all three parameters are 1) for all remaining experiments.

For all the following results, 20% of the users in the dataset were tested and a varying proportion of their ratings (horizontal axis) were used to train the algorithm about their preferences. The first two charts are for experiments using the Book-Crossing dataset.

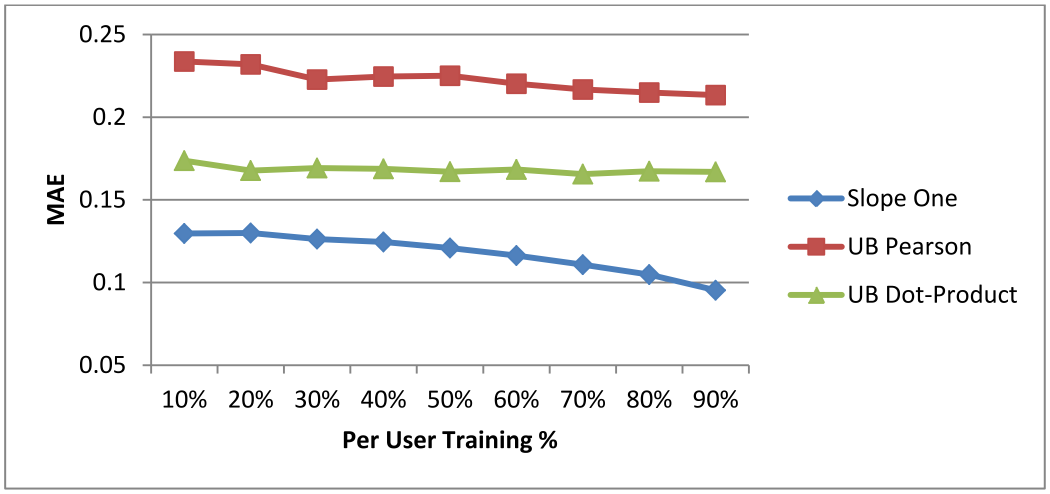

Figure 4 - MAE by algorithm for the Book Crossing dataset

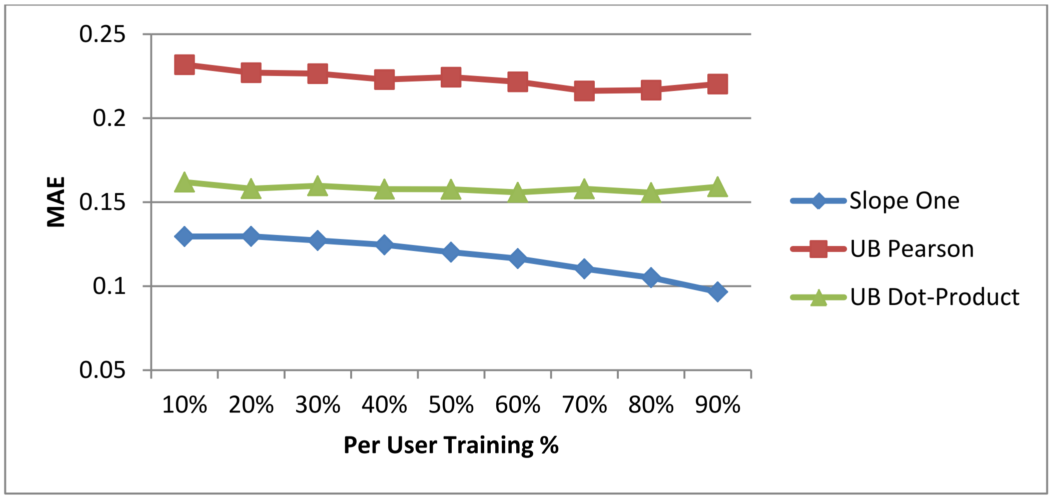

Figure 5 - MAE by algorithm for the Book Crossing (centered) dataset

These results show a very clear separation in accuracy between the three algorithms. SlopeOne performs the best, followed by the User-Based algorithms with Dot-Product similarity taking a clear lead over Pearson similarity. Both SlopeOne and Pearson are unaffected by centering the data. This is expected because they are only concerned with the relative values of ratings and not by their actual values. Dot-Product, however, is affected by actual values of the ratings and it shows a marked, though relatively small improvement in accuracy for the centered dataset. This supports my hypothesis that the behavior of the algorithm on the centered data would more closely match the actual semantics of what constitutes similarity of preferences. For the centered dataset, a pair of ratings that are both of the same sign will add to the similarity between those two users while a pair of ratings of the opposite sign will subtract from their similarity. This matches our intuition that if both people like or both dislike something then they are similar, while if one likes it and the other doesn't then they are dissimilar. The non-centered dataset has slightly different semantics. In a sense, it treats all ratings as positive to varying degrees, even a 1 out of 5. So if one user gives an item a 5 and another gives it a 1, this will add something to the similarity between users because they supposedly both liked it. This does not match our intuition, though, because these users gave the item opposite ratings so we would expect their similarity to be diminished as a result. One way to interpret the greater success of the Dot-Product algorithm on the centered dataset is as confirmation that most users do in fact tend to think of a rating in the bottom half of the rating spectrum as a negative rating rather than as a slightly positive rating.

Another striking thing about the above results is that the SlopeOne algorithm appears to be the only one that benefits substantially from the additional rating information about the evaluated users. It is possible that this is an indication that the two User-Based algorithms do not utilize as much of the available information as SlopeOne. I am not satisfied by this explanation, however, and, since I have not encountered a similar effect in other studies, I am tempted to consider the possibility that it is an anomalous result. At any rate, further analysis is required to fully understand the cause(s) of this phenomenon.

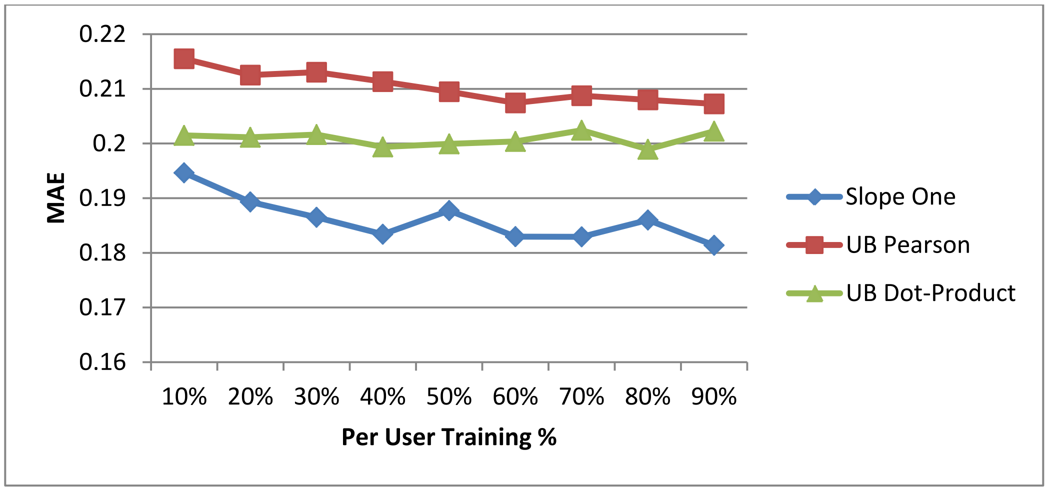

Figure 6 - MAE by algorithm for the MovieLens dataset

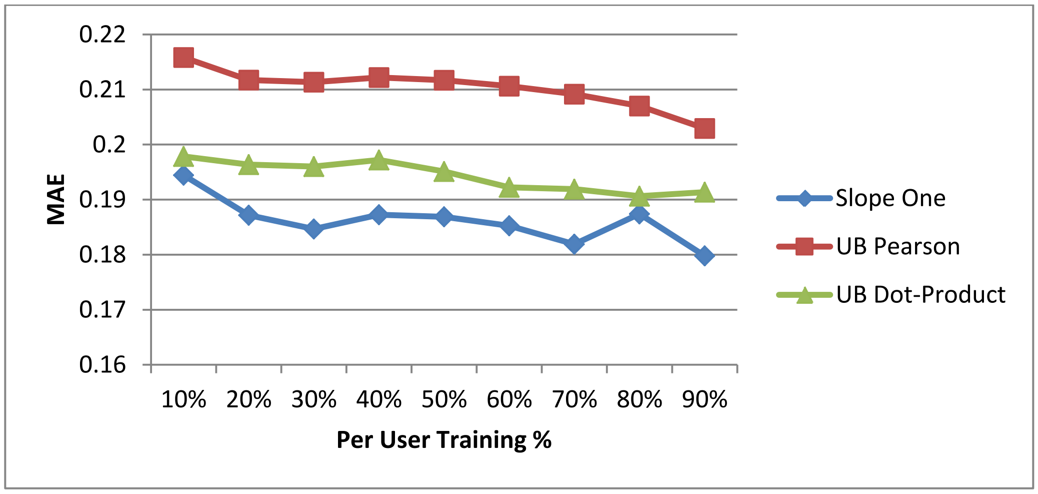

Figure 7 - MAE by algorithm for the MovieLens (centered) dataset

The results for the MovieLens dataset show the same patterns as with the Book Crossing dataset, which is as we would expect. SlopeOne is still the best algorithm, followed by Dot-Product and then Pearson, and the accuracy of the Dot-Product algorithm improves when we center the data around zero. Strikingly, though, all algorithms perform worse on the MovieLens dataset than on the Book Crossing dataset by a substantial margin. This is actually quite a surprising result because while the Book Crossing dataset has more data in total, it is much sparser, which should give the algorithms less data to work with per user. One possible explanation for the better performance on the Book Crossing dataset is that it provides more granular ratings (a 10-point scale instead of a 5-point scale).

The last two charts show the performance of the three algorithms (again using the MovieLens dataset) against the Kendall Tau Correlation Coefficient evaluation metric. Here a higher value on the vertical scale indicates better performance because it means the orderings of user ratings, as predicted by the algorithm, were highly correlated with the actual orderings specified by the real user ratings.

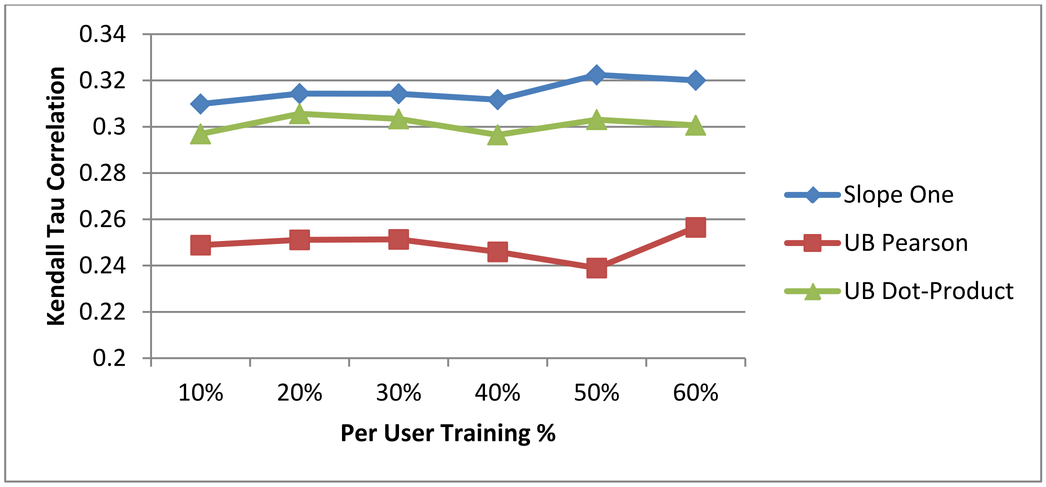

Figure 8 - Algorithms are evaluated against the Kendall Tau Correlation Coefficient evaluation metric using the MovieLens dataset

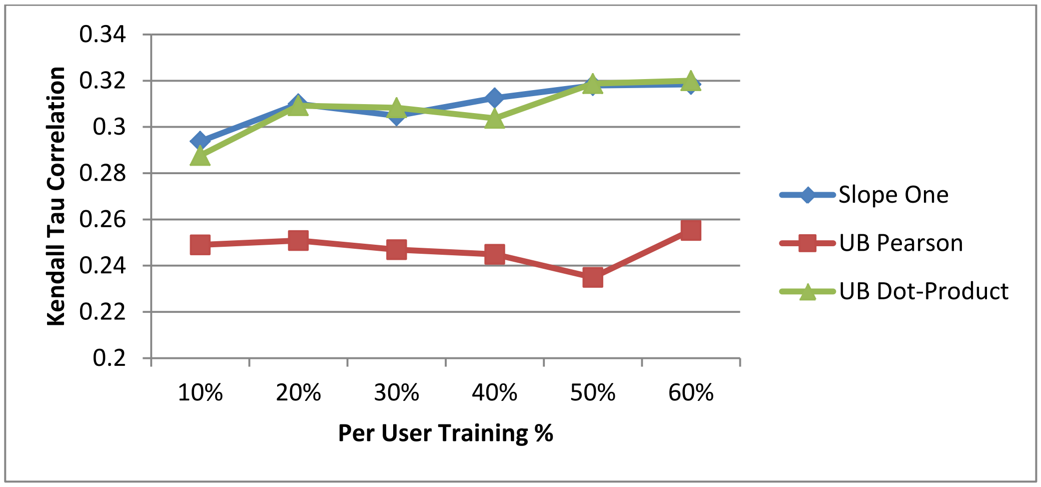

Figure 9 - Algorithms are evaluated against the Kendall Tau Correlation Coefficient evaluation metric using the MovieLens (centered) dataset

For the non-centered MovieLens dataset, the algorithms are still ranked the same in terms of performance. Strikingly, in this experiment, even the SlopeOne algorithm shows very little improvement as the proportion of test users' ratings used in training increases.

In the final chart we see the Dot-Product algorithm finally catch up to SlopeOne. We expected the performance of the Dot-Product algorithm to increase after the data was centered because it did for both datasets under the MAE evaluation metric. What this chart shows, however, is that the Dot-Product algorithm performs relatively better with respect to the other algorithms when evaluated by its ability to rank the value of items for a user compared to when it is evaluated by its ability to precisely predict the rating a user will give to items.

The fact that the Dot-Product algorithm performs reliably better than the Pearson User-Based algorithm and that it is even comparable to SlopeOne according to the Kendall Tau metric suggests that there is something about the Dot-Product similarity function that captures something essential about what constitutes similarity of tastes between users. I think Cacheda et al. [2011] were on to something when they claimed that finding users with similar tastes is harder than often presumed in the collaborative filtering community. Similarity functions like Pearson Correlation assume that the more closely a user's entire set of ratings resembles that of another user, the more similar those users' tastes are. In practice, however, it is probably very rare for any two people to come close to agreeing on all fronts, even in a restricted domain like books. Nonetheless, the more items two users agree on, the more similar their tastes are likely to be, with agreement about items they feel strongly about being stronger indicators of similar tastes. This is the idea modeled by the dot-product algorithm and the fact that it performs well in practice suggests that this conception of what constitutes user similarity is more accurate than Pearson Correlation.

5. CONCLUSION

In this study, I compared the effectiveness of three different collaborative filtering algorithms at predicting the value of books and movies to a specific user. I found that, while the SlopeOne algorithm performed better than both user-based algorithms in almost all cases, when my Dot-Product version of the user-based algorithm was used on a centered dataset and evaluated by the Kendall Tau Correlation Coefficient instead of Mean Absolute Error, it performed as well as SlopeOne. I briefly discussed the limitations of typical evaluation metrics for recommender systems and suggested that, while the Kendall Tau Correlation Coefficient metric is still far from ideal, it measures something closer to what users actually want out of a recommender system. Thus I conclude that, based on the results of this study, the Dot-Product is a better similarity function than Pearson Correlation for User-Based algorithms and that it raises the effectiveness of user-based collaborative filtering to be on par with SlopeOne.

6. REFERENCES

Billsus, D. and Pazzani, M. J. 1998. Learning collaborative information filters. In *Proceedings of the 15th International Conference on Machine Learning*. Morgan Kaufmann, San Francisco, CA, 46–54

Breese, J. S.,Heckerman, D., and Kadie, C. 1998. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence*. 43–52.

Cacheda, F., Carneiro, V., Fernandez, D., and Formoso V. 2011. Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems. *ACM Trans*. Web 5, 1, Article 2 (February 2011), 33 pages

Celma, O. and Herrera, P. 2008. A new approach to evaluating novel recommendations. In *Proceedings of the 2008 ACM conference on Recommender systems (RecSys '08)*. ACM, New York, NY, USA, 179-186

Lemire, D. and Maclachlan, A. 2005. Slope one predictors for online rating-based collaborative filtering. In Proceedings of the SIAM Data Mining Conference (SDM ’05).

McNee, S.M., 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* (CHI EA '06). ACM, New York, NY, USA, 1097-1101

Shardanand, U. and Maes, P. 1995. Social information filtering: algorithms for automating “word of mouth”. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI’95).* ACM Press/Addison-Wesley Publishing Co., New York, NY, 210–217.

Su, X. and Khoshgoftaar, T.M. 2009. A survey of collaborative filtering techniques. Adv. in *Artif. Intell*. 2009, Article 4 (January 2009)

Ungar, L. and Foster, D. 1998. Clustering methods for collaborative filtering. In Proceedings of the Workshop on Recommendation Systems. *AAAI Press*, Menlo Park, CA.