

Open Domain Recommendation: Social Networks and Collaborative Filtering

Sarah K. Tyler and Yi Zhang

University of California, Santa Cruz

Abstract. Commercial enterprises employ data mining techniques to recommend products to their customers. Most of the prior research is usually focused on a specific domain such as movies or books, and recommendation algorithms using similarities between users and/or similarities between products usually performs reasonably well. However, when the domain isn't as specific, recommendation becomes much more difficult, because the data could be too sparse to find similar users or similar products based on purchasing history alone. To solve this problem, we propose using social network data, along with rating history to enhance product recommendations. This paper exploits the state of art collaborative filtering algorithm and social net based recommendation algorithm for the task of open domain recommendation. We show that when a social network can be applied, it is a strong indicator of user preference for product recommendations. However, the high precision is achieved at the cost of recall. Although the sparseness of the data may suggest that the social network is not always applicable, we present a solution to utilize the network in these cases.

1 Introduction

Recommendation systems use information from both users and products to predict which products a given user might prefer. Many **Collaborative filtering** techniques have been proposed to recommend products to a user using information from other users with similar tastes and preferences.

However, prior research work on recommender systems are usually limited to a specific domain, such as movie recommendation with either the MovieLens or Netflix data. While the number of movies in general may be large, the number of new movies produced each year is relatively small, and the number of blockbusters seen by large audiences is even smaller. For most of the similar user pairs, there are often some movies seen by both of them, and for most similar movie pairs, there are often a lot of over lap of viewers. If two users have both rated the same subset of movies, one can aggregate the information to see where their similarities and differences lie; if they share similar genre tastes, favorite actors or other aspects in common. If the number of movies they have both rated is relatively small, it is harder to determine the underlying causes that makes the ratings either similar or dissimilar with respect to each other. In these contexts, additional information could be helpful. Similarly, the problem exists for product

pairs. In close domain recommendation, especially the widely studied movie recommendation scenario, one can usually achieve a reasonable degree of accuracy, because this problem does not dominate the average performance. However, this becomes a serious problem in open domain recommender systems.

To solve this problem, especially for open domain recommender systems, we exploit additional contextual information, the social and trust networks, for recommendation. If the user has selected a group of friends or identify a group of people he/she trusts, we assume they share some interests in common. While a user may add anyone to their social network, there is an implicit assumption of kinship: work acquaintance might share a professional interest while social friends may share hobbies. While the connection between any two individuals may be tenuous, a large group of connections can potentially bias the recommendation system in the right direction. Our goal is to combine these multiple types of information to enhance prediction when a small number of training ratings per product hamper the accuracy of traditional methods.

This paper compared several representative recommendation techniques on a open domain recommendation task. We found that the trust network method worked the best in terms of precision, but was applicable in a limited context thus having low recall. The regularized Singular Value Decomposition method, a collaborative filtering technique, on the other hand, fell victim to the perceptivity of the data. Being a more complex model, it did not have sufficient data to learn the hidden representations and was not able to outperform a simple baseline.

2 Related Work

Typically recommendation filtering uses collaborative filtering. The idea is to find similar individuals, and similar products, in order to make the best recommendation. For example, memory based heuristics, and model based approaches have been used in collaborative filtering tasks [1] [2] [8] [10] [17]. The drawback to these approaches is that they require significant amounts of data per user and per product.

Content based approaches such as latent semantic indexing[7], also help improve recommendation[15], but face scaling issues. As new items are introduced, they may have additional, never before seen features. This is especially true when the context includes natural language such as product description terms, where, according to Zipf's law we can expect to see new words, regardless of how large our training set is.

The notion of using trust networks is not new, both with implicit and explicit networks. Referral Chains use word co-occurrences to build an informal network and uses the network to enhance web search[9]. Yet the question seems to be how to apply it since social connections are not necessarily the same as trusted connections, and may be tenuous at best [14].

TidalTrust is a recommendation technique that allows users to specify a degree of trust for each person in the network [5][6]. The trust is then accumulated over

neighbors of varying distance to create a ranked list approach. The top ranked items are then presented to the user, and are the only ones used in the evaluation.

Other recommendation work involving social networks has focused on the user profiles. Liu et al.[12] harvested profile pages on Friendster, Orkut and MySpace, using the user supplied interests to create Interest Maps to form taste cliques for product recommendation. Their work, however, focused on the profiles and not the connections between users.

The trust network approach has been discussed in conjunction with the same data set we use. Massa et al.[13] showed the potential power of trust networks, and argued for a way of propagating trust over all users. Leskovec et al. [11] also did a study of information cascades, which is typically studied in the blog domain. In their study, users could select who to share a product with. As a result, Leskovec et al. were able to see how the social network influenced the cascade of products in a recommendation environment.

3 Methods

To see how existing recommendation techniques perform on an open domain recommendation task, we looked at three types of methods. The first approach is a base line weighted average that had no personalization. The second approach made predictions based on the social network. Lastly, a collaborative filtering approach commonly used for product recommendation in the movie domain was applied.

3.1 Baseline

In order to determine whether personalized recommendation will help, we first needed a baseline which predicts each product rating independent of the users and sans personalization. The baseline predicts a product's rating by taking the average rating for the given product in the training set. This is a common approach when the number of ratings per product and per user are small. In these instances there is not enough data to model the hidden variables and more complex models can not be trained well. In our case users review on average 2 products, and products have on average 2.4 ratings.

One added benefit of this approach is that it is very fast and simple, as there are no hidden features to estimate. This method also has two drawbacks. First, the predicted rating can easily be skewed by a single review when the training data is sparse. Second, some individuals have a predisposition to rate products either high or low and this model does not account for individual user's rating bias. This algorithm assumes a user's rating of a product i is the average rating of that particular product. The approach is detailed in Equation 1.

$$r_i = \frac{1}{n_i} \sum_u r_{iu} \quad (1)$$

Where r_{ui} is user u 's ratings of product i . n_i is the number of users that have a non missing rating for product i , $n_i = ||\{\text{Number of users where } r_{iu} \text{ is defined}\}||$.

3.2 TidalTrust

We used TidalTrust as our social network recommendation system since it provides a way of propagating trust as Massa et al.[13] called for. TidalTrust utilizes the extended trust network for users by creating a trust flow metric. If two users do not trust each other directly, the amount of trust between them is the product of the trust along each part of the path. In cases where there is more than one path, only the path with the highest trust score is used. When a prediction is needed for a specific user and product, the trust network is explored to find other users who rated the same product. If a member of the trust network has rated the item, their opinion is weighted according to their distance from the initial user, and by how much trust is exhibited along the path between the two users. This method only needs to walk the trust graph to make a prediction which can be very fast. Most users trust a very small fraction of the overall user set. The approach is outlined in Equation 2.

$$r_{iu_1} = \frac{\sum_{u_2 \in S} t_{u_1 u_2} r_{iu_2}}{\sum_{u_2 \in S} t_{u_1 u_2}} \quad (2)$$

Where $t_{u_1 u_2}$ is the trust of u_1 to u_2 and r_{iu_2} is the rating of product i by user u_2 . In our dataset, each trust instance had the same weight. The t_{u_1, u_2} term was constant and dropped out of the equation. The predicted rating in this instance is the average of the rating in the trust network. Used in this manner, TidalTrust is similar in effect to the baseline.

3.3 Singular Value Decomposition

For our collaborative filtering approach we used singular value decomposition¹. We first create a sparse user-product rating matrix. Each user is represented as a vector of products where the index into the vector indicates a given product's rating. Each product is represented as a vector of users where the index into the vector indicates a given user's rating of the product. The singular value decomposition technique is then used to find the hidden representation of users and products in a low dimensional space. Finally a new user-product rating matrix without missing entries is reconstructed using the learned low dimensional representation.

Singular value decomposition[3] is one of the effective collaborative filtering techniques that works well on the well know Netflix data set, which contains 100 Million explicit movie ratings from about 1 Million Netflix customers over a long period of time (1998-2005). It scales well, and is capable of learning the hidden factors that account for individual's uniqueness and rating patterns. It performs much better than Pearson Correlation and Netflix's own system.

In our case, the singular value decomposition will likely be less effective because of how simultaneously diverse and sparse our data is. When dimensionality of the user vectors is high, each user begins to look equally far away from the

¹ We acquired our implementation of Singular Value Decomposition from Timely Development[3] which they used for the Netflix Prize and distribute freely[16].

rest. For our data, we varied the number of hidden features. Since the size of our data is limited, it would be difficult to learn many features, however our diverse domain indicates more features would be necessary.

4 Data

The data set used was collected from Epinions.com[4] in January of 2008. Epinions.com is a commercial site that allows users to rate products, give reviews, and share their opinions with other users. Each user can also specify which other users he or she trusts. Collecting these individual trust instances creates a directed graph of trust.

One of the advantages of Epinions compared to other review sites is that users can rate any kind of item they wish, from mattresses to restaurants. While a benefit to the epinions.com user base, this increases the size of the product space on which to make predictions. As a result, the number of hidden features required to make an accurate prediction in this domain is likely greater than the number needed in domains where there is more consistency between products.

Second, the same feature value may have very different effects on two different products. For example, a \$100 laptop may seem like a steal, but a \$100 pack of gum is likely over priced. Here the same feature value for price influences the recommendation in two different ways even for the same user.

The data sample itself consists of 85,139 user-product ratings, each on a scale of 1-5. The average user-product rating was 4.02 with a standard deviation of

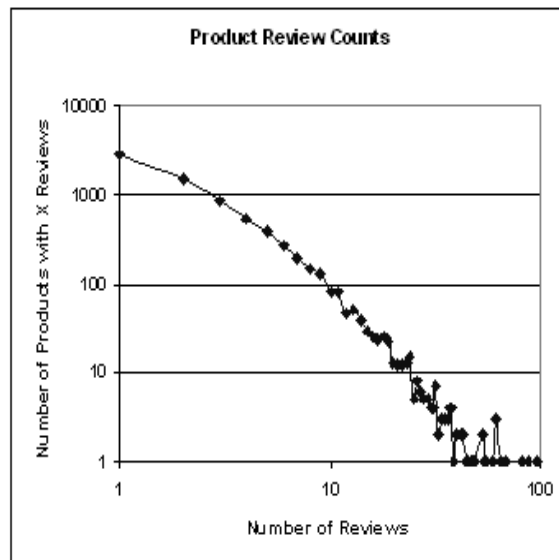


Fig. 1. The graph of the number of products in the test set with N reviews in the training set. Not shown here is the number of the 7428 products with zero reviews.

1.14. There are 35,137 unique products and 41,696 unique users. This means each user has an average of two reviews, and each product has on average 2.4 ratings. Compared with commonly used recommendation data sets, which usually have hundreds of ratings per user and tens of thousands of ratings per product, it is much harder to do well on this data set. Figure 1 shows the that data follows an approximately Zipf law curve.

In the initial TidalTrust paper, users could specify the degree to which they trusted another user. In our data set, however, all we can see is whether or not a user trusts another user. There is no notion of why a user is trusting another user, or how much trust between exists between the users. In this setting we are forced to assume uniform trust in our implementation.

5 Evaluation

In order to evaluate the different models, we need to establish some objectives. In the first case we wanted to see how closely the model conforms to ground truth and we use Mean Square Error as the objective. However, finding the ground truth value is not the primary goal of recommender systems. When creating a recommender system the primary task is to ensure that the item recommended is desirable to the user. Items that are unrelated and not recommended are less important. Therefore, in the second case, we examine whether the highly rated products are desirable to the end users.

The ratings follow a roughly Gaussian distribution, with a center of 4.02. Therefore, to achieve the first objective we used mean squared error (MSE) which follows naturally from the distribution. For the second objective we used several different methods. The first was a precision recall curve, which describes the precisions of recommended items at different “desirable” (or user preference) thresholds. For example, if the “desirable” threshold is 4, we assume a user who rated the product with a 4 or a 5 liked the given product, but a user who rated it with a 1, 2 or 3 did not. In recommender systems, precision is usually more important than recall, so our discussion often centers on precision.

Often users will only look at the first couple of items. To measure how well our models would perform in this environment, we also include macro averages Precision@K and AverageRating@K. It turns out in our environment the number of reviews per product is often low, following a long tail distribution. For example, half of the products that exist in our testing set, do not exist in the training set. Half of those that do, only occur once in the training set. The number of ratings per user follows a similar distribution. Since a small percentage of our products have more than one rating, we often let $K=1$ for our macro averages.

We also looked at an aggregate Precision@K that was independent of users, where all user-product pairs are ordered according to rating and the precision is calculated for the top K product-user ratings. For this evaluation, we selected only two thresholds for “desirable” when computing these measures. The first, 4.0 reflects the average rating users tend towards. Using the average tends to

lead to very good results for all models, so we also choose a threshold of 4.5. Since users are bound by integer ratings, only products that received a 5/5 were deemed interesting. The models needed to make a prediction that rounded to 5.

Our original 20% split of our 80/20 partitioning consisted of 14,873 test cases. However, only 7,445 test instances contained a product or user also in the 80% partition used for testing. For our evaluations we used this subset of users.

5.1 Baseline

For our first objective, we calculated an MSE of 1.36. The F-measure for the baseline in Figure 2 exhibits a knee at 3.5. This is partially due to the fact that the average measurement calculated in the baseline can be skewed by either a very strong or very negative rating. The closer the threshold for “desirable” is to the maximum value or minimum value, the less likely it is for the average rating to recover from a strongly dissenting rating, especially when half of the data that occurs in the test set occurs less than twice. This is also reflective in the sharp downtick of precision, recall and F-measure around 4.5.

Coincidentally, the average rating is 4.02 with a standard deviation of 1.14. The high precision, recall and F-measure for the “desirable” threshold of less than 3 are inflated since the data is so strongly skewed.

The AverageRating@K for $k = 1$ was 4.0 and the AveragePrecision@K for $k = 1$ item was 0.84 and 0.59 for the 4.0 and 4.5 thresholds of “desirability” respectively. This demonstrates that the baseline is reasonably accurate in predicting whether an item is in the top half of all items, but is not as accurate at predicting the very top rated items.

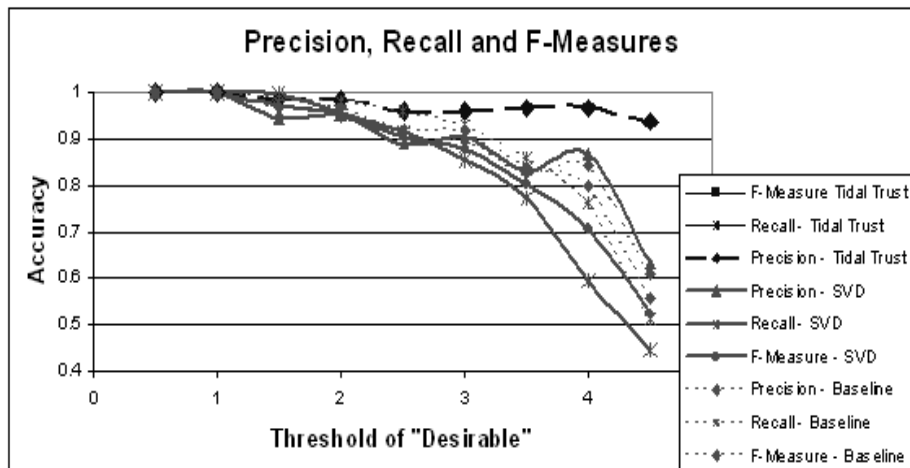


Fig. 2. A Graph of the overall precision, recall and F-measure for the each model. The recall and F-measure for Tidal Trust was less than 0.015 at all thresholds.

5.2 TidalTrust

The TidalTrust MSE was extremely low, 0.52, less than half of the MSE for the baseline although we only calculated the MSE in products where prediction was possible. Along similar lines, precision was very high as shown in Figure 2. Recall, however, was less than 0.008 for all “desirable” thresholds. This was an artifact of how few instances could be rated with TidalTrust. While TidalTrust was able to make strong predictions, it was only able to make a prediction in 75 of the 7,445 test cases, or 1% of the time. It appears that if a product was rated by someone in the given trust network, the two users often agree both in polarity and in magnitude of the rating. However, most of the time two users in a given trust network do not rate the same product. In our data set, a given trust network generally may only have one item that exists in common with more than one user. Still, this approach shows promise.

The strong MSE metric in general, along with the precision curve show that trust networks do have a positive effect on rating prediction, but more work needs to be done to harness that power.

The baseline, when run over the 75 instances that could be predicted using TidalTrust, had a MSE of 0.49, very close and on the surface slightly better than the MSE of TidalTrust. However, in those 75 instances, TidalTrust exactly agreed with the true rating value in 59 instances. In contrast, the baseline was within 0.25 of the true rating in only 32 instances. If we increase the window to 0.5 the baseline was within the true value in 38 instances. When the trust network could make a prediction, it seems to be more accurate than the baseline at finding the exact rating.

In fact, TidalTrust so closely matched the predicted value, that it was only off from the actual value by more than 1 in three of the seventy five instances. When it was off, however, the difference was significant. In one instance TidalTrust’s predicted rating was off by 4 from the true rating, and in another it was off by 3. In contrast, the greatest the baseline was off on these 75 instances was 2.2, however there were 10 instances where the baseline was off by more than 1 and less than 2 to the true rating. If you treat the three instances of magnitude greater than 2 as outliers from both the baseline and TidalTrust, the MSE of TidalTrust is 0.23 where the MSE of the baseline is still 0.48. While in general MSE may not be an appropriate measure for recommendation accuracy, the measure does support that Tidal Trust, on average, beats the baseline.

Figure 3 and 4 offer a different perspective on the same phenomenon with the Precision@K metric. Here, items are ranked according to their predicted rating, independent of users. We see that the precision values are near 1 for all values of K. In environments where precision far outweighs recall, such as product recommendation, there is a clear advantage for Tidal Trust.

AverageRating@K for $k = 1$ was 4.1. The average precision rating for the top 1 item per users are also reflective of the same nature, 96.7 and 93.4 for the 4.0 and 4.5 thresholds respectively. This shows that Social Network information consistently influences product recommendations when they can be applied.

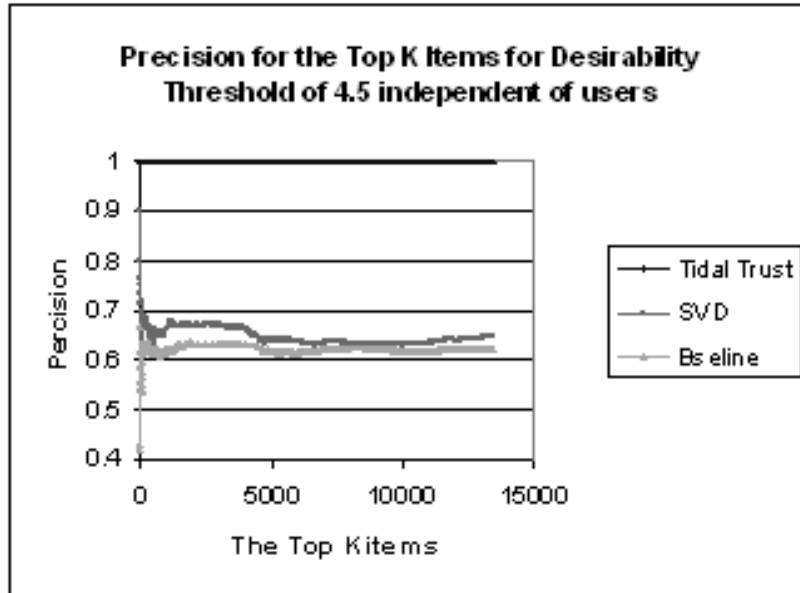


Fig. 3. A graph of the baseline precision, recall and F-measure for different thresholds of “desirable”. The Knee around 3-3.5 occurs before the average product rating.

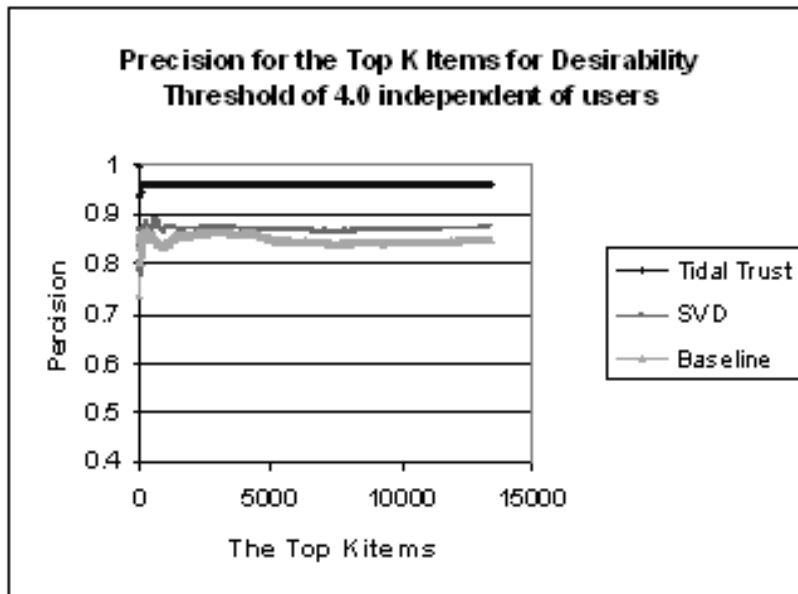


Fig. 4. A graph of the baseline precision, recall and F-measure for different thresholds of “desirable”. The Knee around 3-3.5 occurs before the average product rating.

5.3 Singular Value Decomposition

The final approach used was collaborative filtering in the form of singular value decomposition.

In this case the MSE dropped considerably to 1.5, just above the baseline. This is indicative of two things. The data itself is so sparse that the hidden features cannot be trained well, and that collaborative filtering may require much more data than is available.

The precision and recall curves are similar to the curves in Figure 2, and are nearly identical to the baseline. We see, however that Precision is slightly higher than the baseline by Figures 3 and 4 according to the Precision@K metric.

In terms of the macro average, Precision@K for k of 1, SVD scores 86.2 62.7 for thresholds of 4.0 and 4.5 respectively. Like the baseline, the SVD model is more accurate when it predicts a product is in the top 50

Netflix Data. It is worth noting that Singular Value Decomposition is the collaborative filtering technique that is often used in the movie domain because it is able to distill the movies and users into smaller feature vectors where better inferences can be drawn over them. The MSE on a sample of Netflix data was 0.77, half the MSE of the Epinions.com data. This is also better than most reported MSE for Netflix, but only slightly better and could be due to the sampling bias.

The key difference between Netflix and Epinions is the amount of reviews per user and product. In a random sample of approximately 4589 Netflix users, each user has made an average of 27.9 reviews and each product had an average of 15.4 reviews. This additional information provides more certainty when learning the hidden features.

Our Epinions.com data is simultaneously very sparse, making it difficult to learn hidden features, and very vast, making a greater number of hidden features necessary. This makes sense since we have, on average, 2.4 ratings per product and many products have 1 rating. Learning multiple hidden features is difficult with the limited amount of data.

6 Conclusions

It is clear that the dimensionality requirements of Singular Value Decomposition makes the model complexity too high in the epinions.com context. Although Singular value decomposition outperforms the baseline, it does not outperform it by much. This indicates it was unable to adequately learn the hidden features and improve recommendation accuracy.

The trust network proved to be very useful in the limited number of times it could be applied. It's accuracy was higher than the baseline for both evaluation objectives. Even without considering the possible outliers, the trust network preformed very well. It seems when a product is reviewed by your trust network, your opinion of it will be closer to that of your network, rather than the entire user base. Unfortunately, it could rarely be applied, limiting it's utility. In our

sample it was only able to make a prediction less than 1% of the time. If the social network information could be applied in more cases, the accuracy of the recommendation would surely increase.

7 Future Work

The social network aspect of the data shows great promise for improving recommendation, but the current system is not capable of capitalizing on it. If one can bootstrap the trust network, they may see a significant improvement.

While a specific product may not be common in a trust network, the trust network may share general interests, which then influences what products they would like to purchase. Thus the trust network's relative opinions of similar products may be useful for rating prediction. For example, a Single Lens Reflex (SLR) is a type of advanced lens for digital cameras. For the average user, the additional cost may be prohibitive for them to purchase a camera with such a lens, but to an amateur photographer, the lens may be worthwhile. It is likely that our amateur photographer has friends, colleagues or acquaintances with whom he or she shares his or her interests. If these acquaintances are in the social network of our amateur photographer, they may rate similar products with SLR lenses. Therefore, other products with similar features may have an average rating in the social network that is close to our amateur photographers' true rating for the given product. This can help us estimate the rating for a new, unseen product from the explicit network in the same way that collaborative filtering is often applied for the implicit network.

We propose using the products' contextual features to find similar products. The social network recommendation for each product can be weighted according to the similarity of the desired product in addition to the trust distance between the user and the network. This approximate rating can be used in cases where the predicted rating cannot be established directly. It is our belief that this method will show continued improvement over collaborative filtering in this general domain.

References

1. Breese, J.S., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. Technical report, Microsoft Research, One Microsoft Way, Redmond, WA 98052 (1998)
2. Delgado, J., Ishii, N.: Memory-based weightedmajority prediction for recommender systems. In: ACM SIGIR 1999 Workshop on Recommender Systems (1999)
3. T. Development. Timely development singular value decomposition implementation (visited on January 20, 2008) (2008), <http://www.timelydevelopment.com/Demos/NetflixPrize.htm>
4. Epinions.com (Crawled between January 11th and January 19th, 2008) (2008), <http://www.epinions.com>
5. Golbeck, J.: Personalizing applications through integration of inferred trust values in semantic web-based social networks. In: Proceedings of Semantic Network Analysis Workshop (2005)

6. Golbeck, J.: Generating predictive movie recommendations from trust in social networks. In: Proceedings of the Fourth International Conference on Trust Management (2006)
7. Hofmann, T.: Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.* 22(1), 89–115 (2004)
8. Jin, R., Chai, J.Y., Si, L.: An automatic weighting scheme for collaborative filtering. In: SIGIR 2004: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 337–344. ACM Press, New York (2004)
9. Kautz, H., Selman, B., Shah, M.: Referral web: combining social networks and collaborative filtering. *Commun. ACM* 40(3), 63–65 (1997)
10. Konstan, J.A., Miller, B.N., Maltz, D., Herlocker, J.L., Gordon, L.R., Riedl, J.: GroupLens: Applying collaborative filtering to Usenet news. *Communications of the ACM* 40(3), 77–87 (1997)
11. Leskovec, J., Singh, A., Klienber, J.: Patterns of influence in a recommendation network. In: Ng, W.-K., Kitsuregawa, M., Li, J., Chang, K. (eds.) PAKDD 2006. LNCS (LNAI), vol. 3918, pp. 380–389. Springer, Heidelberg (2006)
12. Liu, H., Maes, P.: Interestmap: Harvesting social network profiles for recommendation. In: Beyond Personalization (2005)
13. Massa, P., Bhattacharjee, B.: Using trust in recommender systems: An experimental analysis. In: Jensen, C., Poslad, S., Dimitrakos, T. (eds.) iTrust 2004. LNCS, vol. 2995, pp. 221–235. Springer, Heidelberg (2004)
14. McDonald, D.W.: Recommending collaboration with social networks: a comparative evaluation. In: Proceedings of the SIGCHI conference on Human factors in computing systems (2003)
15. Melville, P., Mooney, R.J., Nagarajan, R.: Content-boosted collaborative filtering for improved recommendations. In: Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI 2002), Edmonton, Canada (2002)
16. Netflix. Netflix prize (visited on November 30, 2006) (2006), <http://www.netflixprize.com>
17. Zhang, S., Wang, W., Ford, J., Makedon, F., Pearlman, J.: Using singular value decomposition approximation for collaborative filtering. In: CEC 2005: Proceedings of the Seventh IEEE International Conference on E-Commerce Technology, Washington, DC, USA, pp. 257–264. IEEE Computer Society, Los Alamitos (2005)