Design and Analysis of a Gossip-based Decentralized Trust Recommender System

Stefan Magureanu
KTH Royal Insitute of Technology
Stockholm, Sweden
magur@kth.se

Nima Dokoohaki
KTH Royal Institute of Technology
Stockholm, Sweden
nimad@kth.se

Shahab Mokarizadeh
KTH Royal Institute of Technology
Stockholm, Sweden
shahabm@kth.se

Mihhail Matskin
KTH Royal Institute of Technology
Stockholm, Sweden
misha@kth.se

ABSTRACT

Information overload has become an increasingly common problem in today's large scale internet applications. Collaborative filtering(CF) recommendation systems have emerged as a popular solution to this problem by taking advantage of underlying social networks. Traditional CF recommenders suffer from lack of scalability[18] while decentralized recommendation systems (DHT-based, Gossip-based etc.) have promised to alleviate this problem. Thus, in this paper we propose a decentralized approach to CF recommender systems that takes advantage of the popular P2P T-Man algorithm to create and maintain an overlay network capable of generating predictions based on only local information. We analyze our approaches performance in terms of prediction accuracy and item-coverage function of neighborhood size as well as number of T-Man rounds. We show our system achieves better accuracy than previous approaches while implementing a highly scalable, decentralized paradigm. We also show our system is able to generate predictions for a large fraction of users, which is comparable with the centralized approaches.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Qualitative Analysis, Metrics, Accuracy

Keywords

Trust, Decentralized, Gossip, Recommender Systems

1. INTRODUCTION

In today's large scale internet applications, users are dealing with very large amounts of data that can become time-consuming to analyze. This is known as the *information overload* problem. A popular way to address this matter is to use recommendation systems. The most common use of such systems is in e-commerce applications where a user needs to browse very large databases of items. A CF recommender system can alleviate this problem by offering the user a shortened list of items which other clients with similar taste have found interesting. Recommender systems are also being used in social networks as a way of helping users discover new links.

CF Recommenders rely on the presumption that people tend to assign more weight to suggestions coming from friends or people with similar interests. CF strategies similarly put more weight on suggestions received from more similar users. This means that this class of approaches rely on having enough information on users to determine the relationship between them with a high degree of accuracy. This leads to the recommendation accuracy and coverage being dependent on whether the system can accurately determine the type of relationship between users. An alternative approach uses matrix factorization (MF) to generate predictions even for very sparse datasets. At the cost of accuracy, MF-based approaches manage to produce more predictions than CF recommenders, this being the reason why MF-based recommender systems are growing in popularity, as real-life databases are often very sparse.

Traditional recommender systems (both CF and MF based) are implemented in a centralized fashion, in order to increase item coverage. This paradigm however results in high computational costs and renders these systems impractical and costly to run. Thus, decentralized approaches are needed in practice. Distributed approaches rely on techniques borrowed from P2P(peer-to-peer) and Grids such as DHTs or Gossip-based algorithms. Since traditional CF recommenders tend to group similar users together, which is exactly what the Network Overlay is doing, there is no significant loss in prediction accuracy. To further improve performance, it is desirable to implement trust-awareness withing neighborhoods, as a complement to this method.

In this paper we will address the scalability problem of

the centralized trust-based recommenders by using popular techniques from P2P systems. We propose the use of T-Man[9] to cluster similar users together and use a novel trust inference model to improve prediction accuracy over previous trust metrics. We showcase the improvements our approach achieves and analyze its performance over two important datasets, namely Epinions and Yahoo! Webscope[1]. We analyze the influence of neighborhood size and the number of T-Man rounds on prediction accuracy and item coverage. Also, we propose methods for increasing item coverage by varying the distance metric used by T-Man and introducing recursive predictions.

In the following section we will describe previous work in the field of trust-aware recommender systems with a focus on decentralized CF. In section III we will present our approach to creating the network overlay and to the computation of trusts between neighboring nodes. Section IV will be dedicated to accommodating our experimental setup while section V will contain the results of our experiments and evaluation of the performance of the system. The final section will be dedicated to the conclusions and future work.

2. BACKGROUND

In this section we will be briefly describing previous work in the domain of recommender systems with a focus on decentralized approaches and trust inference techniques. Gossip-based recommenders rely on epidemic network overlay algorithms to allow nodes to generate recommendation by only using a limited amount of information available to them in the overlay network. The main advantages of these algorithms are their intrinsic scalability and their ability to generate predictions very fast, since only local information is used.

2.1 Decentralized CF

Using peer-to-peer techniques in the context of distributed recommender systems has been considered in other works. This paradigm shift is common when dealing with very large databases such as the case of social networks, due to its intrinsic scalability. Research into the field of recommenders has also shown interest in decentralized approaches as a replacement for the more traditional centralized techniques. In [25], Ziegler presents a detailed analysis of the challenges related to decentralized recommenders and proposes a framework for implementing such systems. The two main types of CF recommenders are *memory-based* and *model-based*. Memory-based CF recommenders use the ratings of a subset of users to generate recommendations, usually the most similar users to a user or its neighbors in the network. This method is usually referred to as user-based CF. A variation of this method uses the same principle but it predicts the rating of an item based on similar items rather than users. This method is referred to as item-based CF(or content-based CF). Model-based CF recommenders rely on creating a model from a collection of users and items. The resulting model can then be used to make recommendations without the need of storing the collection from which it was inferred. The drawback of this approach is that creating a model is usually more complicated and harder to implement than simply using ratings expressed by other users directly, as in the case of memory-based recommenders.

2.1.1 Model-based Approaches

Random walk models, such as those presented by M. Gori and M. Jamali in [6] and [8] , have been suggested as a solution to sparsity in CF recommender systems. Random walkers work by exploring the social network, starting from a user and taking a probabilistic path outwards into the network until it reaches a certain depth. The probability of a path being chosen is usually dependent on trust values between nodes. During this exploration, the walker uses the ratings of encountered users to create a model that can later be used for recommendations. In [14], B. N. Miller proposes a peer-to-peer recommender system in which nodes exchange ratings with a neighbor at each step in order to construct an item to item similarity matrix which can then be used to make offline predictions. The choice of neighbors as well as determining the neighbors of a user are implementation dependent in this approach. Unlike our approach, in [14], Miller does not maintain an overlay network. This is understandable since his proposed system does not need to keep similar profiles easily accessible and only needs a profile for a one-time computation, after which it can be discarded. In our approach, the most similar profiles must be consulted for every recommendation, meaning an overlay network is needed in order to keep those profiles easily accessible.

2.1.2 Memory-based Approaches

In [7], a DHT-based (Distributed Hash Table) approach is suggested, where the central dataset is organized into "buckets" of users which can be saved on individual nodes, each user using his most suitable "bucket" to choose neighbors with which to generate predictions. In [18], user clustering is suggested as a solution for solving scalability problems as well as a means of improving accuracy. Unlike our approach, Sarwar et. al. [18] presents clusters as groups of users where all the users in a cluster are each other's neighbors, whereas in our case, the "neighbor" relation is directional. A directional "neighbor" relation is desirable since, while a user's neighbors will be the most similar users to it, there might be others that are more similar to a neighbor than said user. Ormandi et. al.[17] determines that using gossip based algorithms to cluster a network in the context of recommender systems offers potential for increasing accuracy of prediction. This is particularly interesting for our work since in [17] the main algorithms being tested are variations of the T-Man algorithm. However the aforementioned work does not analyze item coverage and does not cover trust-awareness in recommender systems, instead focusing on load-balancing.

2.2 Trust Inference

Trust has been the focus of much research since it emerged as a reliable means of improving recommendation accuracy. Trust is presented by Mui et. al. in [15] as "a subjective expectation an agent has about another's future behavior based on the history of their encounters". Several other definitions are presented by Zhou et. al. in [24]. Zhou also presents a more thorough presentation of a wide range of approaches to trust-aware recommender systems.

Throughout this paper we use the term *trust* to denote the confidence a user has in the recommendations of another. As discussed in [12], trust complements CF recommenders by addressing such problems as the reduced computability of similarity between users and improving accuracy of prediction. In [22], Yuan et al. describes trust net-

works as being social networks with user defined trust networks. The authors determine that this type of networks hold the property of small-worldness, which involves having closely clustered users and small average path lengths between any two users. They then use this finding to define a model for recommender systems that takes advantage of the small-worldness of social networks in order to increase both accuracy and item coverage. Several approaches, such as Golbeck[5], Kuter et al.[10], Avesani et al.[2], DuBois et al.[4] and Zarghami et al.[23], also exploit underlying mechanism in a network that allows for explicitly stated trust statements between users. However, not all systems support such features and the ability of users to express confidence in others is limited due to the time and effort required to evaluate other members of the network in order to form an opinion. Therefor, the ability of recommender systems to infer trusts from limited knowledge is still a desired feature.

P. Victor et. al. in [20] proposes a model that uses distrust to complement trust. This approach helps deal more effectively with users that have undesired behavior. The concept of distrust is also used in [19] by N. Verbiest et al. In their work, Verbiest et al. analyses the effect of path length on trust and accuracy. This is particularly interesting to our work since we also observe the effects of using further neighbors on the accuracy and item coverage of our recommender system.

The technique used to infer trust between users is critical to the accuracy of a trust-based CF recommender. Pearson similarity is a popular weight metric, however, as shown in [13], using a more complex weighing measure than just similarity has the potential to offer more accurate results, especially in sparse datasets. Approaches such as those proposed by J. Golbeck et al. [5], [10] take advantage of trust ratings explicitly stated by the users themselves to infer trusts between nearby members of the network through trust propagation. In [23], Fazeli et al. proposes the use of a local trust metric together with a global trust metric computed using the in-degree of a node in relation with the trust on each incoming edge. The global trusts are then used to create a global repository of top trusted users for each item which can then be referenced by other nodes in order to find new neighbors in the network. It is important to note that our trust inference technique does not require any user-defined trust between nodes and it computes trust knowing only user ratings.

The trust metric proposed by O'Donovan and Smyth in [16] is similar to ours in this respect. O'Donovan uses the known ratings to create an artificial history of predictions for each user. By *predicting* the known ratings of users using all the other users and counting the amount of *correct* predictions that each user makes, O'Donovan and Smyth establish a global trust for each user as the ratio of correct predictions to total predictions of a user(in [16], they also propose item level trust which is similar to user level trust only applied on items; both trust models can be used concurrently to offer better results). Our approach follows the same path of using artificial predictions of known ratings to adjust trust values for users. However, we compute the trusts differently, solely within a neighborhood, resulting in *local trusts* rather than *global trusts*. O'Donovan's method requires the analysis of the whole database when computing a user's trust rating which will greatly impede scalability and will become more computationally demanding as the number of users grows. Our technique can calculate a user's trust towards any subset of users in the network, making it easily implementable in a decentralized paradigm.

Unlike us, O'Donovan computes trusts as an absolute feature of a user, all users in the network have the same trust in a given user. We refer to our trusts *local* because, as we will show in section III, the values computed are relevant only in the context of a neighborhood. Thus, the trust between two users depends on the profiles of the other neighbors as well as the profiles of the two users.

3. APPROACH

In this section we will present our proposed approach. First, we will describe how we use the T-Man algorithm in the context of a recommender system for social networks, and then we describe the trust inference technique we use to compute the trust between two users based on their known ratings.

3.1 Network Overlay

A network overlay is a network constructed on top of another network, by reorganizing the logical links between nodes in order to make it more suitable for the application logic. In our case, the overlay network will be built on top of the social network and a user's resulting neighbors will not necessarily be his friends in the social network.

3.1.1 Distance Metric

The T-Man algorithm[9] is widely used in P2P systems for obtaining overlay networks for a very diverse range of purposes. T-Man is a gossip algorithm that works by having each node maintain a set of neighbors by exchanging on each iteration neighborhood entries with a node in that set deemed most *suitable*. After the exchange takes place, both nodes will replace entries from their neighborhood with nodes that are more *suitable* from the received set of entries. *Suitability* is usually represented by a distance function that is implementation dependent. To increase convergence speed, T-Man is usually used in conjunction with Cyclon[21], a gossip-based random peer sampling algorithm. Thus, each node will have a random view beside its neighborhood. After each exchange of entries with a peer, along with the step described earlier, each node also keeps the most *suitable* entries from its random view in its neighborhood.

Our goal is to fill each user's neighborhood with its most similar peers. An intuitive distance metric for our case would be using the Pearson similarity coefficient. However, Pearson similarity has a negative impact on the number of relevant predictions the system is able to make, since it disregards the number of items in common between two users. Thus, we will use a slightly different version :

$$Similarity(u_1, u_2) = \frac{\sum_i r_{u_1,i} \times r_{u_2,i}}{\sqrt{\sum_i C \times r_{u_1,i}^2} \times \sqrt{\sum_i r_{u_2,i}^2}} \quad (1)$$

where $r_{u,i}$ is the rating user u assigned to item i and C is a value in the interval [0,1] if u_2 has not rated item i and 1 if u_2 has rated item i. The higher C is, the more weight we put on the two users having common rated items at the expense of putting less weight how similar their ratings are. This new metric offers a balance between how similar two users are in terms of ratings for common items as well as in terms of the number of items in common. In our experiments we

use $C = 0.5$. We can now form neighborhoods based on the similarity of ratings in users profiles as well as the number of items they have in common, meaning we are more likely to be able to make predictions for items more relevant to users. It is important to note that in the trust inference step and recommendation step, only the nodes in the neighborhood will be used.

3.1.2 Dealing With Sparsity

In order to deal with potential coverage issues, we propose using *recursive rating prediction* requests. This way, if a neighbor does not have the desired item rated, it can ask its neighbors for a prediction for the item in question, and pass it to the user asking for a prediction as its own. This will greatly reduce situations in which none of the neighbors of a user have the item the user is interested in. However, this behavior can not scale very well, the number of involved neighbors potentially increases exponentially. Fortunately, as we will present in the section V, having recursive calls with a depth of maximum 2 is sufficient for even a very sparse database. To reduce complexity for higher range values, recursive calls can be made only when none of the neighbors have the desired item rated. Furthermore, the overhead can be reduced by allowing nodes to store the ratings of their neighbors locally, thus significantly reducing the number of nodes involved in recursive predictions at the expense of used memory.

3.2 Trust Inference

Our approach for computing trust is inspired by machine learning techniques[11], in the sense that we use a user's known ratings as a training set, based on which we tune the trusts so that we obtain sufficiently accurate predictions for the known ratings. In our experiment, trust is computed after a certain number of T-Man rounds determines the neighborhood of each user. In real scenarios, the T-Man algorithm can be run continuously and the trusts can be computed on *stable* neighborhoods, where a *stable* neighborhood is one that has not changed in a set number of rounds.

3.2.1 Modeling the system

We chose to use the formula described by (2) to calculate predictions of an item's rating for a user. The formula is one of the most popular ones for predicting ratings and is also used by Resnick prediction and O'Donovan et al in [16].

$$\frac{\sum_n (r_{n,i} - \overline{r_n}) \times w_n}{\sum_n w_n} = r_i - \overline{r} \tag{2}$$

where $r_{n,i}$ is the rating of neighbor n for item i, $\overline{r_n}$ is the average of user n's ratings, w_n is the weight the user assigns to neighbor n(or the trust of the user for n; we will be using the terms "trust" and "weight" interchangeably), r_i is the user's rating for item i and $\overline{r}$ is the user's average rating. It should be noted that the denominator sum is not of weights in absolute value, meaning we only consider weights to be positive. This makes perfect sense since we are interested in the proportions between them and not their actual values.

Given the fact that we know all the ratings involved, since we are applying it for items a user has already rated, we can interpret (2) as an equation with the trusts representing the variables. If we use the formula for every item a user has rated, we can form a linear system of I equations and N variables, where I is the number of rated items the user currently has, and N is the number of neighbors, of the form:

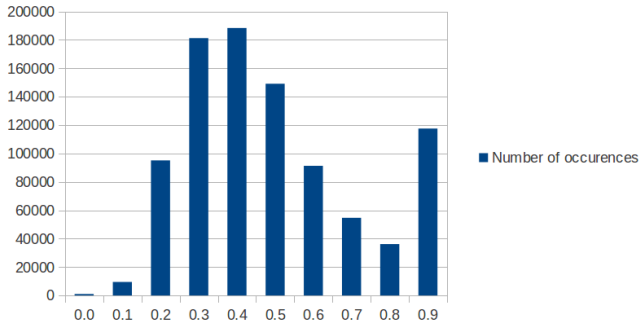

Figure 1: Trust distribution obtained for Yahoo! Webscope dataset. Each bracket represents an interval of size 0.1 in the allowed trust interval of (0,1].

$$\begin{cases} \sum_n (r_{n,0} - \overline{r_n} - r_0 + \overline{r}) \times w_n = 0 \\ \dots \\ \sum_n (r_{n,i} - \overline{r_n} - r_i + \overline{r}) \times w_n = 0 \end{cases} \tag{3}$$

This system most likely will not have positive solutions, however we can try to approximate them. For this step, we have chosen to use the algorithm proposed by D. Cartwright in [3], which uses expectation maximization[11] and iterative proportional fitting to gradually converge to an approximation of the solution of systems similar to the one above. However, in order to be able to use this method, we must first overcome the constraint that the system can only have positive coefficients. To ensure that the coefficients remain positive no matter what ratings are involved in the calculation, we will modify the system, as follows:

$$\begin{cases} \sum_n (2 \times R_{max} + r_{n,0} - \overline{r_n} - r_0 + \overline{r}) \times w_n = C \\ \dots \\ \sum_n (2 \times R_{max} + r_{n,i} - \overline{r_n} - r_i + \overline{r}) \times w_n = C \\ \sum_n w_n = N \times Trust_{mean} \end{cases} \tag{4}$$

where $C = 2 \times R_{max} \times \sum_n w_n$ which, considering the last equation in the system, will actually be a constant $C = 2 \times R_{max} \times N \times Trust_{mean}$. In the above formulae, R_{max} is the maximum rating available in the profile database. It is now obvious that all the coefficients will become positive.

3.2.2 Approximating the Trusts

As we stated earlier, we will use D. Cartwright's approach to approximate solutions for equation (4) . Our case is a particularly simplified example of the types of systems the algorithm can solve, meaning the algorithm can be reduced to a much simpler form. Thus, the algorithm is modified as follows: where C_i is the value of the sum in the left term of *equation* i of the system with the current values for weights, $coef_{i,n}$ is the coefficient of w_n in *equation* i, or $2 \times R_{max} + r_{n,i} - \overline{r_n} - r_i + \overline{r}$, and C is the constant mentioned above. Also we must mention that we initialize the weights with random values in the trust interval we chose to use. It is very important to state that the trust values are updated

Algorithm 1 Approximating the weights.

while not converged **do**
  for each n in $neighbors$ **do**
    $w_n = w_n \times \frac{\sum_{i \in Items} \frac{C}{C_i} \times coef_{i,n}}{\sum_{i \in Items} coef_{i,n}}$
  end for
end while

simultaneously. The terminating condition is described in more detail in the following section.

3.2.3 Convergence

In Cartwright's algorithm[3], the convergence of the algorithm is given by the condition that the Kullback-Leibner divergence between the solutions vector before and after a step should be below a certain threshold. However, we want to achieve a balanced trust distribution and we prefer to prevent some users from receiving an imbalanced high trust value. For example, if one of the neighbors had exactly the same ratings as the user, to solve the system, an obvious solution is to give all the other neighbors 0 trust and use the value required to satisfy the last equation of (4) , i.e. $N \times Trust_{mean}$. Such situations are often encountered in sparse datasets, when the user has a small number of rated items but a significantly higher number of neighbors, resulting in far more variables than equations.

To deal with this setback, we will limit the values for trust to a certain interval. We used a minimum value of 0.1 and a maximum value of 1. If at a step, a trust value exceeds either, it is rounded to the appropriate end of the interval after each iteration. Even so, trust distribution tends to clutter around the edges of the interval, so we interrupt the iterations when the number of values that have reached either end of the interval exceeds 10% of the size of the neighborhood. This value can be changed if needed. However, we observed that it gave a good trust distribution across the allowed interval as shown in figure 1.

4. EXPERIMENTAL SETUP

To test the performance of our system and trust inference method, we used two datasets. The first dataset is obtained from Yahoo! Webscope [1], which represents relatively less sparse databases. The second is from Epinions.com, which reflects the sparse databases one would expect to encounter in a real scenario. The Yahoo dataset consists of 15,400 users and 300,000 ratings for 1,000 items, with a minimum of 10 ratings per user. The Epinions [1] dataset consists of 49,290 users, 139,738 items and 664,824 ratings, some of the users(9,127 of them) having empty profiles. This sample has a more disproportionate distribution of ratings across the scale 1 to 5, a large majority of the ratings being either 4 or 5. Also, in the case of the Epinions dataset, the average number of ratings per user is only 13.

To evaluate our methods, we will use the leave-one-out methodology in which we hide one rating for every user and then run the algorithm and try to predict the hidden ratings. We initialize the neighbors with a set of randomly picked nodes. In a real scenario, this initialization could be replaced by choosing to use the friends of a user in a social network for example. As a measure of item coverage, we evaluate the number of users for which the system can generate a prediction for the hidden rating i.e. at least one of the neighbors can provide a rating for the item in question. We evaluate the performance on the above mentioned datasets using 3 trust metrics: Pearson correlation, the metric proposed by O'Donovan in [16] and our approach, presented earlier.

[1]Epinions dataset available at http://www.trustlet.org/wiki/Epinions_datasets

5. EXPERIMENT RESULTS

In this section we will discuss the performance of our system in terms of accuracy of prediction and coverage. We will be measuring accuracy in terms of MAE (Mean Absolute Error) and Coverage as the number of users for which the system can generate a prediction for the requested item i.e. the item hidden from the profile of the user.

We have structured our findings in two sections. First, we will present the performance of our trust metric and compare the obtained MAE by different approaches against the chosen baselines. Secondly, we will present the influence of the network overlay algorithm and the results of our attempts to maximize coverage in our decentralized system.

5.1 Trust Metric and MAE

In this section we will discuss the results of our trust metric compared to previous approaches and evaluate the evolution of accuracy and coverage function of the neighborhood size. The chosen baselines for our system are regular prediction taking all the users into account with trust represented by the Pearson similarity and by the trust metric proposed by O'Donovan in [16]. These baselines are referred to as *Pearson control* and *O'Donovan control* in the figures in this section. We compare the performance of these two methods against our system using the overlay algorithm in combination with the trust metrics used in the baselines as well as with our own metric, described in the section III.

Figure 2 shows the performance of the different trust metrics over the size of the neighborhood in the case of the Yahoo! Webscope dataset. We can observe that all the metrics have increased accuracy with the size of the neighborhood and all show very similar performance. The chosen baselines are the variations of the Resnick prediction technique, presented in [16], (using Pearson correlation as similarity and O'Donovan trusts) over the whole set of users.

We notice that our decentralized approach yields better results than the best performing baseline(0.95 compared to the baseline of 1.01), while presenting the added advantage of scalability. The trust metric used in neighborhoods seems not to influence performance, Pearson, O'Donovan trust and our trust showing very close performance. This is because the Yahoo dataset only presents the ratings for 1,000 items (compared to 139,738 in the case of Epinions). This will allow for more relevant similarity computations when forming neighborhoods since odds are nodes will encounter peers with more items in common, compared to Epinions. Since the resulting neighbors will have a higher number of items rated similarly to the active user, chances are they will also have other interesting items for the active user rated similarly among themselves. Meaning that, when making predictions, since all the suggestions will be similar, the weights of each user will be less relevant than in the case of Epinions.

On the Epinions dataset however, our trust metric outperforms Pearson similarity implemented both in a distributed

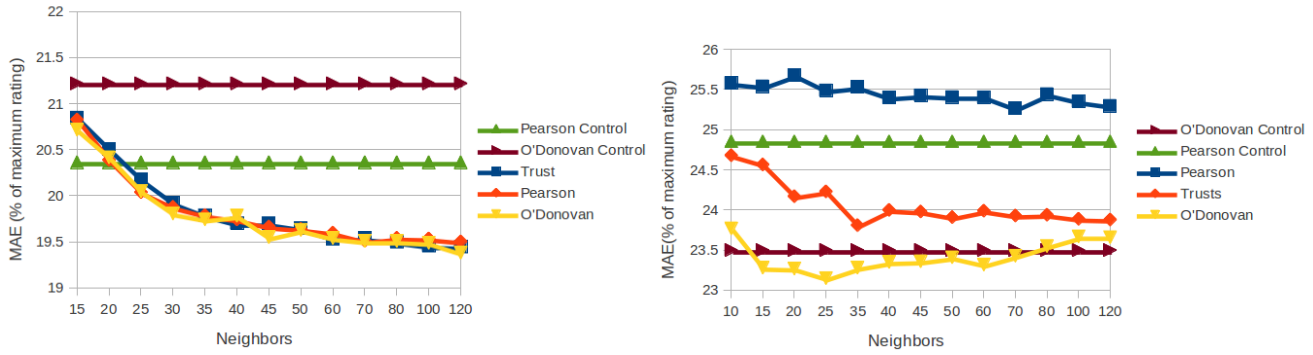

Figure 2: The evolution of MAE over neighborhood size for Yahoo! Webscope(left) and Epinions(right) datasets after 150 T-Man rounds. We use our decentralized approach in combination with Pearson similarity, O'Donovan's trust metric and our trust metric and compare with the baselines set by O'Donovan's trust metric and Pearson similarity over the whole dataset.

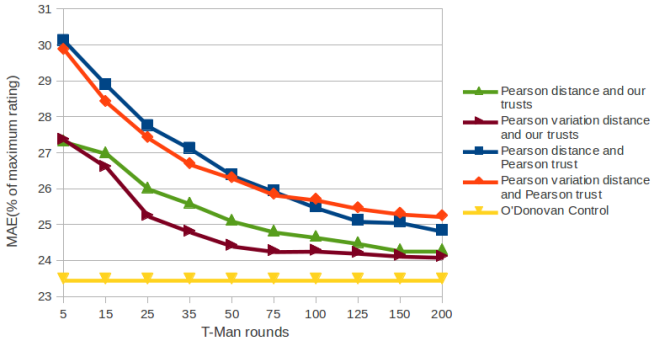

Figure 3: Evolution of MAE over T-Man rounds for different trust metrics on Epinions dataset.

and in a centralized fashion. O'Donovan trust performs better especially for smaller neighborhood sizes, however, we must note that these trusts were computed using the whole database as a training set(best performance scenario). Even so, the results are similar for high neighborhood size to our trust metric. O'Donovan's metric achieves better accuracy partly because it often fails to make predictions for items that have been rated by only a few users and for which other approaches produce predictions of lower quality. This failure is due to not finding two users that have rated an item within the designated error threshold required for computing O'Donovan trust. Also, implementing this metric in a purely decentralized fashion would be significantly more difficult as it requires a large training set. In these experiments, the distance metric chosen for forming the overlay is the variation of Pearson similarity presented in formula (1).

In figure 3 we can clearly notice how accuracy increases as the network overlay converges. In this experiment we compare the evolution of accuracy with the number of T-Man rounds and also compare the two earlier mentioned distance metrics used in the T-Man implementation: regular Pearson similarity and its variation presented in formula (1). From the point of view of prediction accuracy, both distance metrics give similar results, with the proposed variation yielding slightly better results when combined with our proposed trust. In the case of using Pearson similarity as trust, the variation only gives lower errors in the earlier rounds while using the default Pearson similarity as a distance metric offers slightly better accuracy near the convergance of the network.

5.2 Network Overlay and Coverage

We will refer to a request from a node to a neighbor for the rating of an item as a *prediction request*. A *prediction request* is successful if the neighbor can provide a rating for the desired item (either his own rating or a prediction that it obtains from its own neighbors). In section III we discussed ways in which we can overcome sparsity and we proposed two methods which can be used concurrently. The first method is using a slight variation of the Pearson similarity as the distance metric in the implementation of the T-Man overlay management algorithm. The specified function is presented in equation (1). The second is using recursive *prediction requests* which will allow neighbors that do not have a requested item rated to subsequently ask its neighbors for a prediction of the rating for the item in question and pass the prediction as its own rating to the initiating node. To avoid such requests going on forever, we limit such calls by a time-to-live, which we refer to as *search range*.

Figure 4 presents the results of varying the *search range* for the Epinions dataset. It is worth noting that in terms of coverage, our proposed decentralized approach yields very similar results for our trust metric and Pearson similarity, analyzed in the previous subsection. In the left image of figure 4, we can clearly observe that using regular prediction techniques, the coverage is very poor for sparse datasets like the one analyzed in this experiment. For 120 neighbors, less than 25% of the nodes manage to obtain a prediction from their direct neighbors, meaning that in 75% of the cases, none of the 120 neighbors had the item in question rated. Since the Epinions dataset is a good representation of a real-life scenario, such low coverage is unacceptable.

We notice that increasing the range to 1 (i.e. allowing immediate neighbors to ask their neighbors for a prediction for that item) greatly increases coverage. In this case coverage in the case of 120 neighbors approaches the baseline set by using Resnick prediction with Pearson similarity over the whole dataset (73% coverage). The baseline is only 73% because 9,127 of the 49,290 in the network have no items

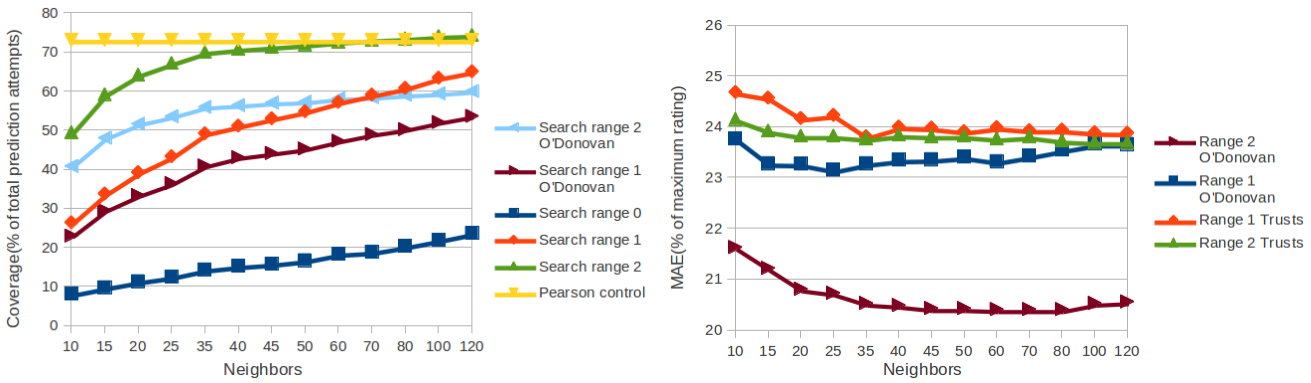

Figure 4: Influence of search range on item coverage and prediction accuracy for Epinions dataset.

rated meaning we can not verify prediction results against their real rating and we consider these requests as unsuccessful. On top of that, we have hidden 1 item from each user, meaning that sparsity is even worse than in the initial dataset. Despite being computed in a centralized fashion, O'Donovan trust offers a significant loss of over 10% total item coverage across the whole range of neighbors(15% for range 2), as we stated in section 5.1. If we increase the coverage to match that of our metric, the MAE in the case of Epinions would increase to levels comparable to that of Pearson similarity.

Once we increase the search range to 2, we notice coverage is very close to the baseline starting from a neighborhood size of only 35. Even so, using a range of 2 could potentially involve 35^3 nodes in a single request, for the worse case scenario, which might produce significant overhead. However, it is still less demanding and more scalable than having to consult the ratings of every user in the network, since recursive queries can be done asynchronously between nodes. We also proposed a few ways of reducing the overhead of recursive hops in section 3.1.2. Given the sparsity of the dataset used for this experiment, we believe that using a search range higher than this is unneccessary for most real-life cases.

In the plot on the right of figure 4, we notice the effect of increasing the search range on prediction accuracy. We notice that both in the case of our trusts and O'Donovan trusts, using a range of 2 yields better results than using a range of 1. This is to be expected since there will be significantly more ratings available for the prediction phase. The advantages of using a search range of 2 are particularly obvious in the case of the O'Donovan metric where the accuracy is increased sensibly, significantly surpassing the centralized approach using the same trust metric(1.02 MAE compared to the centralized approach of 1.17). From this experiment we can infer that choosing a range for requests will represent a compromise between accuracy and coverage, on one side, and overhead on the other.

Figure 5 depicts how coverage is affected by the convergence of the network and by the distance metric used by the T-Man algorithm, discussed at the beginning of the section 3. We can observe that as neighbors become more similar to a user, coverage increases significantly. The experiment was run on the Epinions dataset, for a neighborhood size of 60 and using a search range of 1. We notice how coverage

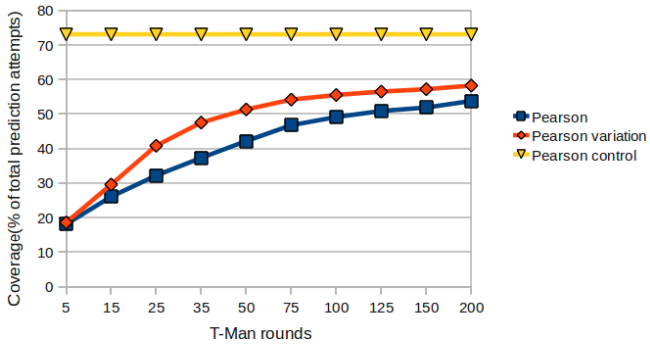

Figure 5: Influence of distance metric on coverage as network converges in the case of Epinions.

is increased when using the variation of the Pearson similarity mentioned in equation (1). This makes perfect sense since taking the number of items in common between two users into account when calculating similarity increases the probability that highly similar users will also be interested in similar items. Taking into account the results presented in figure 3, we notice that our proposed variation offers better coverage and better accuracy, as in the case of using our proposed trust metric for rating prediction. Thus, we can assume that our metric is overall a more desirable distance metric to be used in the implementation of the T-Man algorithm.

6. CONCLUSION AND FUTURE WORK

In this work, we have used techniques inspired from P2P applications to create a scalable recommender system model. We have used the T-Man algorithm to cluster similar users together using a variation of the Pearson similarity as a distance metric. We have shown the improvements in item coverage and accuracy the proposed variation achieves and implemented a new trust inference method to obtain directional trust values between a user and its neighbors while only knowing their ratings. Our inference method can compute trusts between a user and any given subset of neighbors by having them predict the user's known ratings and modifying trusts values until predictions are close to the known ratings. We showed that our decentralized approach achieves better accuracy than two popular centralized models while

maintaining comparable item coverage. Also, our trust inference method in the context of our decentralized approach performs better than using Pearson similarity and is comparable to the popular trust metric proposed by O'Donovan and Smyth in [16], while being, easily applicable in a decentralized model, in contrast to the latter metric. We also showed our trust metric allows for higher item coverage than O'Donovan's, even though the latter metric is computed centrally and ours only requires a limited neighborhood.

Future work will include observing the influence of different privacy settings of users on trust values and implementing a policy for punishing users that share too few items through lowering their trust. Also, it is worth exploring the potential of other distance metrics for T-Man and/or prediction formulae and the effectiveness of implementing our trust metric in a centralized fashion. We are also interested in pursuing gradient-based boosting algorithms as a possible replacement for our trust inferrence method.

7. REFERENCES

[1] Yahoo! Webscope dataset ydata-ymusic-user-artist-ratings-v1_0. http://research.yahoo.com/Academic_Relations.

[2] P. Avesani, P. Massa, and R. Tiella. A trust-enhanced recommender system application: Moleskiing. In *In SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, pages 1589–1593. ACM Press, 2004.

[3] D. Cartwright. An iterative method converging to a positive solution of certain systems of polynomial equations. *Journal of algebric statistics Vol. 2, No. 1*, pages 1–13, 2011.

[4] T. DuBois, J. Golbeck, and A. Srinivasan. Predicting trust and distrust in social networks. In *SocialCom/PASSAT*, pages 418–424. IEEE, 2011.

[5] J. Golbeck. Combining provenance with trust in social networks for semantic web content filtering. In L. Moreau and I. T. Foster, editors, *International Provenance and Annotation Workshop*, volume 4145 of *Lecture Notes in Computer Science*, pages 101–108. Springer, 2006.

[6] M. Gori and A. Pucci. Itemrank: A random-walk based scoring algorithm for recommender engines. In M. M. Veloso, editor, *IJCAI*, pages 2766–2771, 2007.

[7] P. Han, B. Xie, F. Yang, and R. Shen. A scalable p2p recommender system based on distributed collaborative filtering. *Expert Syst. Appl.*, 27(2):203–210, 2004.

[8] M. Jamali and M. Ester. Trustwalker: a random walk model for combining trust-based and item-based recommendation. In J. F. E. IV, F. Fogelman-SouliÃl', P. A. Flach, and M. Zaki, editors, *KDD*, pages 397–406. ACM, 2009.

[9] M. Jelasity and O. Babaoglu. T-man: Gossip-based overlay topology management. In *The Fourth International Workshop on Engineering Self-Organizing Applications (ESOA'06)*, Hakodate, Japan, May 2006.

[10] U. Kuter and J. Golbeck. Sunny: A new algorithm for trust inference in social networks using probabilistic confidence models. In *AAAI*, pages 1377–1382. AAAI Press, 2007.

[11] B. Marlin. Collaborative Filtering: A Machine Learning Perspective. Master's thesis, University of Toronto, 2004.

[12] P. Massa and P. Avesani. Trust-aware collaborative filtering for recommender systems. In *In Proc. of Federated Int. Conference On The Move to Meaningful Internet: CoopIS, DOA, ODBASE*, pages 492–508, 2004.

[13] P. Massa and P. Avesani. Trust-aware recommender systems. In *RecSys '07: Proceedings of the 2007 ACM conference on Recommender systems*, pages 17–24, New York, NY, USA, 2007. ACM.

[14] B. N. Miller, J. A. Konstan, and J. Riedl. Pocketlens: Toward a personal recommender system. *ACM Trans. Inf. Syst.*, 22(3):437–476, July 2004.

[15] L. Mui, M. Mohtashemi, and A. Halberstadt. A computational model of trust and reputation. In *35th Hawaii International Conference on System Science (HICSS)*, 2002.

[16] J. O Donovan and B. Smyth. Trust in recommender systems. In *IUI '05: Proceedings of the 10th international conference on Intelligent user interfaces*, pages 167–174. ACM Press, 2005.

[17] R. Ormandi, I. Hegedüs, and M. Jelasity. Overlay management for fully distributed user-based collaborative filtering. In P. D'Ambra, M. R. Guarracino, and D. Talia, editors, *Euro-Par (1)*, volume 6271 of *Lecture Notes in Computer Science*, pages 446–457. Springer, 2010.

[18] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Recommender systems for large–scale e–commerce: Scalable neighborhood formation using clustering. *In Proceedings of the Fifth International Conference on Computer and Information Technology (ICCIT'02)*, December 27–28 2002.

[19] N. Verbiest, C. Cornelis, P. Victor, and E. Herrera-Viedma. Trust and distrust aggregation enhanced with path length incorporation. *Fuzzy Sets and Systems*, (0):–, 2012.

[20] P. Victor, C. Cornelis, M. De Cock, and P. Pinheiro da Silva. Gradual trust and distrust in recommender systems. *Fuzzy Sets Syst.*, 160(10):1367–1382, May 2009.

[21] S. Voulgaris, D. Gavidia, and M. van Steen. Cyclon: Inexpensive membership management for unstructured p2p overlays. *J. Network Syst. Manage.*, 13(2):197–217, 2005.

[22] W. Yuan, D. Guan, Y.-K. Lee, S. Lee, and S. J. Hur. Improved trust-aware recommender system using small-worldness of trust networks. *Knowledge-Based Systems*, 23(3):232 – 238, 2010.

[23] A. Zarghami, S. Fazeli, N. Dokoohaki, and M. Matskin. Social trust-aware recommendation system: A t-index approach. In *Web Intelligence/IAT Workshops*, pages 85–90. IEEE, 2009.

[24] X. Zhou, Y. Xu, Y. Li, A. Josang, and C. Cox. The state-of-the-art in personalized recommender systems for social networking. *Artificial Intelligence Review*, 37(2):119–132, May 2011.

[25] C. N. Ziegler. *Towards Decentralized Recommender Systems*. PhD thesis, Albert-Ludwigs-Universität Freiburg, Germany, 2005.