Using Trust in Recommender Systems: an Experimental Analysis

Paolo Massa[1], Bobby Bhattacharjee[2]

[1] International Graduate School in Information and Communication Technologies
University of Trento, 38050 Povo (TN), Italy,
massa@itc.it
[2] Department of Computer Science, University of Maryland, College Park, Maryland,
bobby@cs.umd.edu

Abstract. Recommender systems (RS) have been used for suggesting items (movies, books, songs, etc.) that users might like. RSs compute a user similarity between users and use it as a weight for the users' ratings. However they have many weaknesses, such as sparseness, cold start and vulnerability to attacks. We assert that these weaknesses can be alleviated using a Trust-aware system that takes into account the "web of trust" provided by every user.
Specifically, we analyze data from the popular Internet web site *epinions.com*. The dataset consists of 49290 users who expressed reviews (with rating) on items and explicitly specified their web of trust, i.e. users whose reviews they have consistently found to be valuable.
We show that any two users have usually few items rated in common. For this reason, the classic RS technique is often ineffective and is not able to compute a user similarity weight for many of the users. Instead exploiting the webs of trust, it is possible to propagate trust and infer an additional weight for other users. We show how this quantity can be computed against a larger number of users.

1 Introduction

Recommender Systems (RS) [8] are widely used online (e.g. in *amazon.com*) to suggest items that users may find "interesting". These recommendations are generated using two main techniques: content-based, and collaborative filtering. Content-based systems require manual intervention, and do not scale to large item bases. Collaborative filtering (CF) [2] systems do not depend on the semantics of items under consideration; instead, they automate the recommendation process based solely on user opinions.

While CF algorithms are promising for implementing large scale recommender systems, they have their share of problems. The problems with pure CF systems can be classified in three domains: problems affecting new user start up, sparsity of useful information for existing users, and relatively easy attacks on system correctness by malicious insiders. We describe these attacks in detail in Section 1.2. In this paper, we propose an extension to pure CF systems, and assert that our extension addresses all of the problems with currently implemented

collaborative filtering algorithms. Specifically, we argue that these problems can effectively be solved by incorporating a notion of *trust* between users into the base CF system. To this end, we present an analysis of a large scale deployed recommender system (*epinions.com*): our work clearly identifies the problems with the base CF system, and we describe how trust-based extensions would solve these problems for this data set, and for similar systems.

The contributions of this paper are three-fold:

- We articulate specific problems with collaborative filtering systems currently deployed, and present a new solution that addresses all of these problems.
- We present a thorough analysis of a large, existing RS data set, and show that the problems we identified do exist in reality. (Note that we do not specifically show the existence of malicious insiders, but it is clear that such an attack is possible on the base system).
- We present preliminary results that show that our trust-based solution does alleviate or eliminate the problems that we identify in the base system.

The rest of the paper is structured as follows: first, we introduce Recommender Systems (Section 1.1), their weaknesses (Section 1.2) and how trust-awareness can alleviate them (Section 1.3). Section 2 presents experiments on *epinions.com* that support our thesis while Section 3 concludes discussing new research lines based on the provided evidence.

1.1 Recommender Systems

Recommender Systems (RS) [8] suggest to users items they might like. Two main algorithmic techniques have been used to compute recommendations: Content-Based and Collaborative Filtering. The Content-Based approach tries to suggest to the user items similar to her previous selections. To achieve this, content-based RSs need a representation in terms of features of the items. Such a representation can be created automatically for machine-parsable items (such as news or papers) but must be manually inserted by human editors for items that are not yet machine-parsable (such as movies and songs). This activity is expensive, time-consuming, error-prone and highly subjective. Moreover, for some items such as jokes, it is almost impossible to define the right set of describing features and to "objectively" classify them.

Collaborative Filtering (CF) [2], on the other hand, collects opinions from users in the form of ratings to items. When asked for a recommendation, the system identifies similar users and suggests the items these users have liked in the past. The interesting point is that the algorithm doesn't need a representation of the items in term of features (i.e. genre and actors for movies) but it is based only on the judgments of the user community. Because of this, CF can be applied to virtually any kind of item: papers, news, web sites, movies, songs, books, jokes, locations of holidays, stocks. Since CF techniques don't require any human intervention for tagging content, they promise to scale well to large item bases. In the rest of this paper we concentrate on RSs based on CF.

The traditional input to a CF algorithm is a matrix that has a row for every user and a column for each of the items. The entry at each element of the matrix is the user's rating of that item. Figure 1 shows such a matrix.

	Matrix Reloaded	Lord of the Rings 2	Titanic	La vita è bella
Alice	2	5		5
Bob	5		1	3
Carol		5		
Dean	2	5	5	4

Table 1. The users × items matrix of ratings is the classic input of CF.

When asked for a recommendation by an user (recommendee), a standard CF algorithm performs 3 steps.

- *It compares the recommendee against every other user in the community computing a user similarity coefficient.*
  Different techniques have been proposed for this task. Pearson correlation coefficient is the best performing and most used [3]. Proposed alternatives are Constrained Pearson correlation coefficient, Spearman correlation coefficient and cosine similarity [3]. The Pearson correlation coefficient $w_{a,u}$ (Equation 1) represents the similarity of user a with user u with regard to their ratings on items and is defined as:

$$w_{a,u} = \frac{\sum_{i=1}^{m}(r_{a,i} - \overline{r}_a)(r_{u,i} - \overline{r}_u)}{\sqrt{\sum_{i=1}^{m}(r_{a,i} - \overline{r}_a)^2 \sum_{i=1}^{m}(r_{u,i} - \overline{r}_u)^2}} \tag{1}$$

  where m is the number of items rated by both a and u, $r_{a,i}$ is the rating given by user a to item i and $\overline{r}_a$ is the mean of ratings of a. The coefficient is in [−1,1]. It is important to underline that the coefficient can be computed only if there are items rated by both the users.
- *It predicts the recommendee's rating for every item she has not yet rated.*
  The predicted rating (Equation 2) that user a might give to item i is the mean rating of a plus the weighted sum of deviation from mean rating for every user where the weight is the user similarity with user a.

$$p_{a,i} = \overline{r}_a + \sum_{u=1}^{n} w_{a,u}(r_{u,i} - \overline{r}_u) \tag{2}$$

  It is possible to consider only the k most similar users or all the users. The user similarity of a with every user is used as a weight for the opinions of that user. Intuitively if user A is very similar to user B, the opinions of user A are given importance when creating a recommendation for user B.
- *It suggests to the user the items with highest predicted ratings.*

1.2 Weaknesses of Recommender Systems

RSs are a very useful tool for dealing with information overload since they can suggest to an user the few items worth consuming out of a huge item set. We have seen that Collaborative Filtering can be applied in every domain and with no additional efforts required for human editors to tag content or classify items.

Despite the potential, we believe RSs have still failed in unveiling their disruptive power because of a number of weaknesses we present in the rest of the Section.

Data Sparseness A realistic CF matrix would contains millions of users and millions of items. In practice users only rate a few of the entire set of items and this results in a very sparse matrix.

The "sparseness" of a CF matrix is the percentage of empty cells. *Eachmovie* and *Movielens*[3], two of the public datasets typically used in research, are respectively 97.6% and 95.8% sparse. We will see in Section 2 that the *epinions.com* dataset has even higher sparseness. Sparseness is also a huge problem for freshly created RSs that, when start operating, have no ratings at all.

Consequence of sparseness is that, on average, two users, picked uniformly at random, have low overlap. For this reason, the Pearson coefficient is noisy and unreliable. In practice, it is often the case that there is no intersection at all between users and hence the similarity is not computable at all.

Singular Value Decomposition has been proposed as a technique for dimensionality reduction and consequent reduction of sparseness [9]. However, on extremely sparse datasets, it has shown no improvements over standard techniques.

Cold start A second weakness is the so called "cold start problem" [5]. This is related to the situation when a user enters the system and has expressed no ratings. CF cannot compute a recommendation under such cases. We define "cold start users" as the users who have expressed less than 5 ratings. With these users, RSs present the biggest problems and are usually unable to make good quality recommendations. However, these are the users who need good quality recommendations as an incentive to continue using the system.

Attacks on Recommender Systems Another concern is related to the existence of users that want to maliciously influence the system: if the process of creating recommendations is known and if the ratings of every user are publicly available (as it is in *epinions.com*, for instance), a very simple but effective attack is the following. Let suppose the malicious principal wants the system to recommend to user $target$ the item *spambook*. She can simply create a new user $fakeuser$, rate in the same way all the items rated by user $target$ and also rate with the highest possible rating *spambook*. In this way, when looking for users similar to $target$, the system will assign to $fakeuser$ a user similarity of 1 and will weight her rating on *spambook* the most and will probably end up recommending *spambook* to user $target$. The malicious users can even create a whole bunch of fake users

[3] For publicly available datasets, see www.grouplens.org

reinforcing each others. In general it is also possible to trash the dataset quality by automatically creating many users with pseudo-random rating profiles. We believe that this kind of attacks were not a problem for current commercial RSs because they are centralized and creating a user and rating items it is a time-expensive activity. This is true unless the attacker can create a bot doing this but there is no economic incentive for now in doing this, even if we see this can become a sort of very effective "hidden" commercial spam. We are not aware of research lines taking into account this as a problem for RSs. We think it will become a huge concern as soon as RSs start becoming more decentralized systems [6], in the sense that users profile will not be stored in the hard disks of one central RS server but will be spread across the sites of community members and publicly available in well defined semantic formats.

RSs are hard to understand and control The last weakness is reported in some papers [4, 10] that say how users often see RSs as a black box and are not aware of their working model. In fact, with current RSs it is very hard (or impossible) for the user to control the recommendation process so that if the RS starts giving bad quality recommendations, usually the user just stops using it [11].

1.3 Solution to weaknesses: Trust-awareness

We believe that taking into consideration trust relationships between principals in the system will alleviate the problems that beset RSs.

In the rest of the Section, we define our trust model and give anecdotal evidence that trust-awareness can solve the previously stated weaknesses. Section 2 will confirm the claims based on an empirical analysis of *epinions.com* dataset.

A trust statement is an explicit assertion of the fact a user trusts another users. User A can only create trust statements specifying the users she trusts. In the context of Trust-aware Recommender Systems, a trust statement made by user A towards user B means that user A consistently finds the reviews and ratings of user B valuable.

A trust (or social) network is constructed by aggregating every trust statement. A trust network is a graph in which nodes are users and directed edges are trust statements.

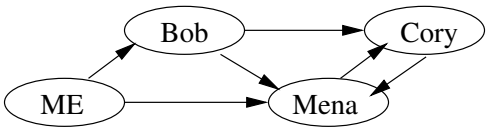


Fig. 1. A Trust network. Arrows are trust statements.

Trust metrics [1] can be used to propagate trust over the social network. Intuitively, if A trusts B and B trusts C, it is possible to infer something about

how much A might trust C. Trust metrics can be used exactly for inferring trust in unknown users depending on the social network. In this paper we analyze how trust can be used to extract useful information for making recommendations and overcome the previously cited RSs weaknesses for a real community. We leave as a future work the proposition of a trust metric able to actually predict trust values in unknown users.

We have seen in Section 1.2 that because of the high sparseness of the CF matrix, it is rare that two users have rated some items in common (we present evidence of this in Section 2). However, the Pearson correlation coefficient can be computed only on overlapping items. Thus, for any given user, the number of other users with whom it is possible to compute similarity is low. Instead, propagating trust over the social network allows to reach a larger portion of the user base. In this way it is possible to compute an alternative weight related to how much we should take other users into account when computing a recommendation.

Cold start users, who are the most critical users for standard CF, can benefit highly from trust propagation as long as they provide at least one trusted friend in the community. This can be an effective mechanism to rapidly integrate new users, especially if compared with standard CF where users are usually required to rate at least 10 items before to receive a recommendation.

Further the attack we outlined in Section 1.2 is not effective as long as the fake users are not trusted by any real user.

Unlike traditional RS, that are often seen by users as black boxes [4, 10], we think that the concept of social network is easier to grasp. Human Computer Interaction studies [10] are needed to investigate the best ways to visualize the social network and how this could help the user in understanding the recommendation model and in controlling it.

It is important to note that trust metrics predict personalized values of trust in unknown users. For this reason, the inferred trust in user A can be different for user B and user C and depends on their different webs of trust. This is different from what many current online systems do. For example, PageRank [7] (used by *google.com*) computes a global value for every web page that is the same for every searcher, *ebay.com* computes a unique global value for every buyer and seller as well, as does *slashdot.org* for its contributors.

Lastly, even if trust-awareness can be introduced inside a single centralized server, centralized approaches of data collection in general are subject to the following huge disadvantages. Expressing information (what you like, who you like) in a centralized RS server means that only that server will be able to use it. This results in users profiles being scattered in portions on many different, not cooperating servers and every single server suffers even more of sparseness. Moreover, this means the user cannot move from one RS to another without losing her profile (and, with it, the possibility to receive good recommendations). We also believe it does not make sense to introduce concerns about trust and then leave the computation of recommendations out of possible user control. We think Trust-aware Recommender Systems [6] demand a decentralized environ-

ment where every user publishes their information (trust and ratings) in some Semantic Web format and then every machine has the possibility of aggregating this information and computing recommendations on behalf of her user.

In this Section, we have argued that Trust-awareness alleviates problems with standard RSs. In the rest of the paper, we present results from a large deployed system that support our claims. We begin with an overview of our data set.

2 Experiments on epinions.com

In this Section we present experimental results on *epinions.com* that show that our trust-based solution does alleviate the problems that we stated in Section 1.2. We begin by explaining how the dataset was collected. We then present some statistics of the community such as number of expressed reviews and friends and ratings distribution across items. We then analyze the differences in computability for the two quantities that can be used as weights for every user: Pearson coefficient and Trust. We show how the second is computable on many more users than the first and how this is especially true for "cold start users".

Epinions.com is a web site where users can review items (such as cars, books, movies, music, software, ...) and also assign numeric rating in the range from 1 (min) to 5 (max). Users can also express their *Web of Trust*, i.e. "reviewers whose reviews and ratings they have consistently found to be valuable"[4] and their *Block list*, i.e. a list of authors whose reviews they find consistently offensive, inaccurate, or in general not valuable. While the *Web of Trust* of every user is public and everyone can see it with a browser, the *Block list* is kept private by *epinions.com*. Since *epinions.com* does not provide a complete list of users or items, we obtained the dataset by crawling the *epinions.com* web site during November 2003. We conducted a depth-first search starting from some users who were classified as Advisors in the category Movies, Books and Online Stores & Services and repeatedly following all the users in their web of trust. For every user we also fetched all the reviews the user made. The information collected for every user is shown in Figure 2.

Of course, because of the way we obtained the dataset, we did not fetch all the users; in fact we downloaded all the users that were reachable walking the web of trust of starting users. If there were users not trusted by some of the reached users they were not added to our dataset. The same argument applies for items. In the collected dataset there are only the items that were rated at least once by one of the reached users.

2.1 Statistics on the Community

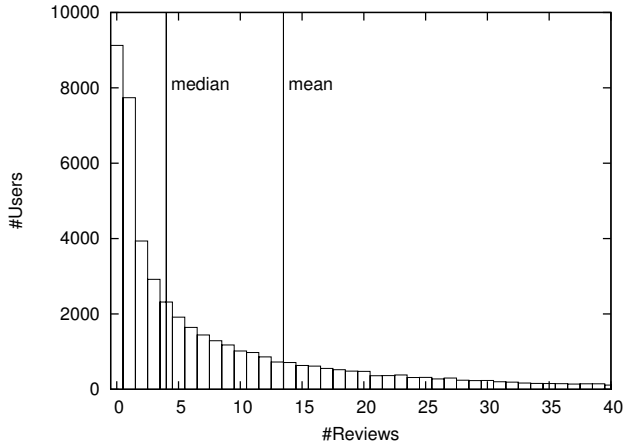
Our dataset consists of 49290 users who rated a total of 139738 different items at least once. The total number of reviews is 664824. The sparseness of the collected dataset is hence 99.99135%. This sparseness is very high if compared to

[4] From the Web of Trust FAQ (http://www.epinions.com/help/faq/?show=faq_wot)

Username	an-epinions-user				
URL	http://www.epinions.com/user-an-epinions-user				
Web of trust	trusted user URL	added on date			
	/user-my-friend1	Nov 18 '03			
	/user-my-friend2	Nov 17 '03			
	...	...			
Written reviews	item name	item URL	class	rating	date
	Dragonheart	/mvie_mu-1071583	Videos & DVDs	5	Sep30'02
	Caribbean Pot	/Caribbean_Pot	Restaurants	3	Aug19'03
	...	...	...	...	...

Table 2. The table shows an example of the information we collected for every user.

the 2 public datasets *Eachmovie* and *Movielens* (respectively, 97.6% and 95.8%) commonly used in research. We believe one of the reasons is that *Eachmovie* and *Movielens* have a smaller item base (1628 and 3900 movies, respectively). Moreover, rating an item on *epinions.com* is a time-expensive activity since the user must write a review of at least 20 words. Instead the two research systems presented the user many movies in the same web page and allowed her to rate all of them simultaneously with few clicks.



Reviews	Users	Perc.
0	9127	18.52%
1	7739	15.70%
2	3937	7.99%
3	2917	5.92%
4	2317	4.70%
5	1916	3.89%
6	1642	3.33%
7	1443	2.93%
0 to 4	26037	52.82%
4+	23253	47.18%
Total	49290	100.00%

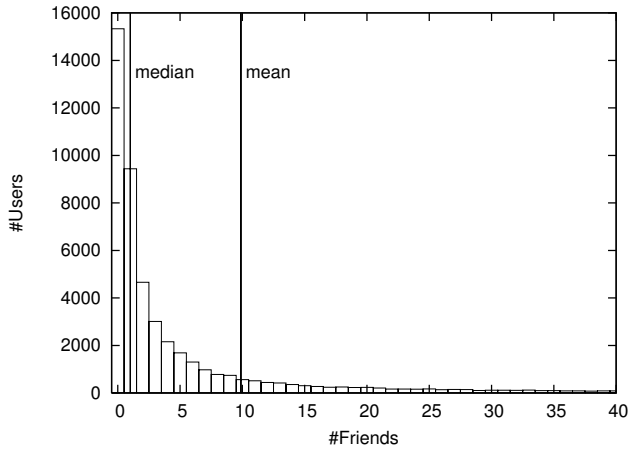
Fig. 2. Numbers of users who created x reviews. The vertical lines are the mean (13.49) and the median (4). Note the high percentage of "cold start users" (users who have less than 5 reviews).

We would like to stress the fact that *epinions.com* is a very well known site with a big and active community and nevertheless it has such a sparse matrix of ratings. The problem of sparseness is even bigger for Recommender Systems that have just started operating or those that don't have a large community base.

Moreover it is important to remember that the items that are in our dataset received at least one rating (that's because of how we collected the data); if we were to consider all the ratable items on *epinions.com* the theoretical sparseness would have been even much higher.

Figure 2 shows the number of users who created a certain number of reviews. The mean number of created reviews is 13.49 with a standard deviation of 34.16. The median is 4. In particular, 9127 users created 0 reviews while the maximum number of reviews created by one user is 1023. It is important to have a look at what we have called "cold start users", users who expressed less than 5 reviews. They are 26037 and represent 52.82% of the population.

Figure 3 shows the number of users who have expressed a certain number of friends (we call friends of *User A* the users that are in the web of trust of *User A*). The mean number of friends is 9.88 with a standard deviation of 32.85. The median is 1. In particular, 15330 users (31.30%) expressed 0 friends, i.e. have their web of trust empty, and one user added 1760 users to her web of trust.



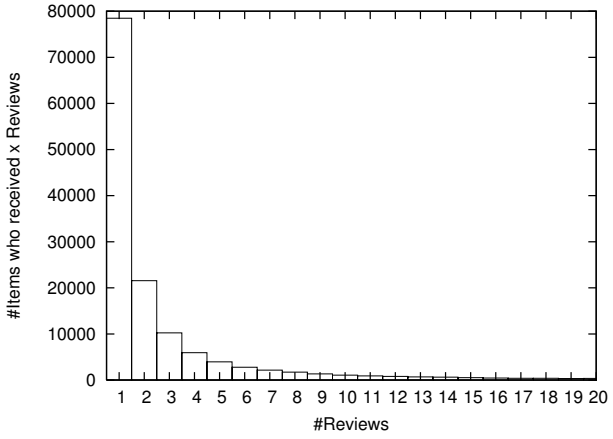
Friends	Users	Perc.
0	15330	31.10%
1	9435	19.14%
2	4661	9.46%
3	3009	6.10%
4	2157	4.38%
5	1689	3.43%
6	1303	2.64%
7	974	1.98%
8	776	1.57%
1 to 4	19262	39.08%
4+	14698	29.82%
Total	49290	100.00%

Fig. 3. Numbers of users who expressed x friends. The vertical lines are the mean (9.88) and the median (1).

Though not directly related to our results, it is interesting to note the distribution of ratings. In our dataset, 45% of the ratings are 5 (best), 29% are 4, 11% are 3, 8% are 2 and 7% are 1 (worst). The mean rating is hence 3.99.

In order to compute the user similarity between 2 users, a very important quantity is the distribution of reviews over items. It makes a big difference for the overlapping of 2 random users if the ratings generally concentrate on few items rated by almost everyone or if they are uniformly distributed over the item base. Figure 4 shows the distribution of ratings over items. Note that the vast majority of items (precisely 78465 corresponding to 56.15% of the item base) received only one review and this makes them totally useless for computing user

similarity between users, infact their columns in the matrix only contains one rating value and so, along these columns, there doesn't exist 2 users that overlap and can be compared. Only 11657 items (corresponding to 8.34%) have 10 or more reviews.



Reviews	Items	Perc.
1	78465	56.15%
2	21543	15.42%
3	10224	7.31%
4	5921	4.24%
5	3958	2.83%
6	2787	1.99%
7	2154	1.54%
8	1703	1.22%
9	1326	0.95%
2+	61273	43.85%
10+	11657	8.34%
Total	139738	100.00%

Fig. 4. Number of reviews received by Items. Note that 78465 items (56.15%) received only 1 review and are totally useless for computing user similarity. Since we fetched only the items when present in a user's review, in our dataset there are no items with 0 reviews, while instead on *epinions.com* they are the greatest part.

We have seen that the sparseness in the users $\times$ items matrix of ratings is very high (99.99135%) and that the greatest part of items received few reviews. In Section 2.2, we will show how this results in low overlapping among user ratings and hence in reduced computability of Pearson coefficient. In Section 2.3, we will show how trust propagation suffers less from the sparseness and allows to infer "trust" in a larger number of users.

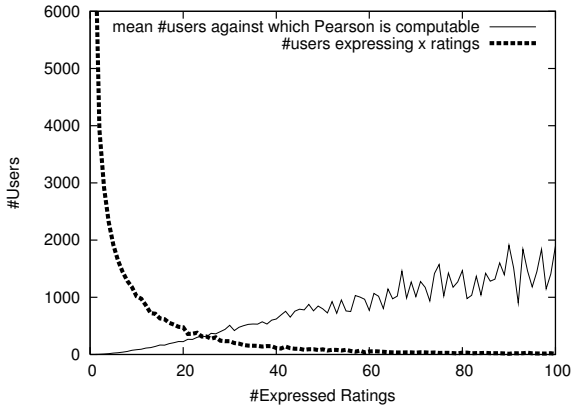
2.2 Computability of User Similarity

When predicting the rating of user A on an item, RSs (based on Collaborative Filtering) first compute a similarity coefficient of user A against all the other users. This coefficient is then used to weight the ratings given by other users.

As argued in Section 1.3, we could also use direct or propagated trust as a weight for the opinions of other users.

We show here how user similarity between 2 users is a quantity that can be usually computed against a very small portion of the user base and usually can be computed only based on a small number of overlapping items producing a noisy and unreliable value. Instead the number of users reachable through some trust chain is generally very high. Along a chain of trust is possible to propagate and infer trust using a trust metric.

This difference in the number of users in which it is possible to compute similarity and trust is even exacerbated for cold start users, i.e., users who expressed few ratings and friends. These users are usually the largest portion of users and also the ones that will benefit the most from good quality recommendations.



User base	Nr. Users	Frac. of Total	Mean Nr. Comparable Users
All	49290	100.00%	160.73
Cold	26037	52.82%	2.74
10-	33504	67.97%	11.27
20-	40597	82.36%	33.53
50+	2661	5.40%	1447.88
100+	882	1.79%	2162.52

Fig. 5. The thick line plots the number of users who have expressed a specific number of ratings. For each of these users, the thin line plots how many *comparable* users exist in the system on average. (By comparable we mean that the 2 users have rated at least 2 items in common). The table groups results for class of users depending on number of expressed ratings.

In Figure 5 we plot the number of comparable users averaged over all the users who created a certain number of reviews. We define 2 users comparable if they have rated at least 2 items in common. On every comparable user it is possible to compute the Pearson correlation coefficient and to use it as a weight for that user. Unsurprisingly, the users who created many reviews have a higher number of users against which Pearson is computable. However the plot shows that even for that users the coverage over user base is very limited: for example, the 54 users who created 60 reviews have a mean number of users against which Pearson is computable of 772.44 that is only the 1.57% of the entire user base.

Figure 5 shows only a portion of the total graph in fact the y axis can go up to 49290 users and the x axis up to 1023 items. In an ideal system, it would be possible to compare one user against any other user; in this case the mean number of users would have been 49289 independently of the number of written reviews.. Instead, Figure 5 makes evident how on *epinions.com* dataset, the technique is far from ideal.

Let us now concentrate on "cold start users" who, by the way, are the majority of users in the system. For them (as shown in the 2nd row of the table in Figure 5) Pearson is computable on average only against 2.74 users over 49290! And also only 1413 of the 26037 cold start users have at least 10 users against which Pearson is computable. It is worth noting that, even for the most overlap-

ping user, Pearson correlation coefficient is computable only against 9773 user that is 19.83% of the entire population.

This plot is a stray evidence of how Pearson correlation coefficient is often incomputable and hence ineffective.

2.3 Computability of Trust

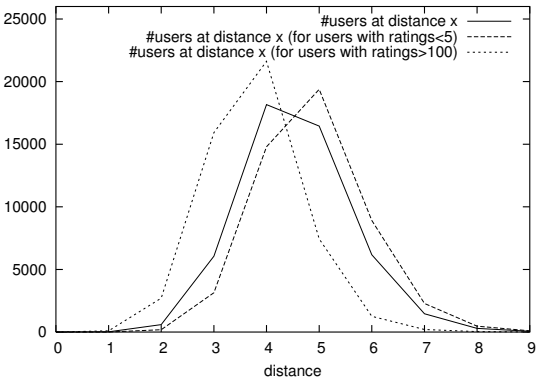
In this Section, we provide evidence about the potential use of trust in alleviating the RSs problems.

We have suggested that, for a given user (the recommendee), it is possible to compute trust in every other user and then use this computed quantity as a weight for those users. Trust (either direct or propagated) can potentially also be even combined with user similarity. In this paper, we simply provide evidence of the fact that it is possible to infer a trust value on a big portion of the user base. For this reason, we compute for every user the minimum number of steps needed to reach every other user. In this way, for every recommendee, we end up with some class of equivalence on trust: all the users reachable in one step (direct friends), the users reachable in 2 steps (friends of friends), etc. We are of course aware that it makes a big difference if a user at distance 2 from the recommendee is trusted by only one of her friends of by all of her friends, but the goal of this paper is not to propose a suitable trust metric for the domain but just to show that propagating trust is feasible and has the potential to solve RSs weaknesses. Note that on our dataset, sophisticated inference is difficult because the collected *epinions.com* trust data is binary, i.e. users either completely trust others (or not).

Figure 6 shows the mean number of users at a certain distance. The different lines represent different subsets of users (all users, users who created less than 5 and more than 100 reviews). Some users cannot reach all the other users through any trust chain. Such users are not considered when computing the mean number of users at a certain distance. Table of Figure 6 shows for different class of users the number of users who are connected to everyone in the system. Considering that 15330 users provided 0 friends and of course cannot reach all the other users through some trust chain, over the remaining users (33960) almost everyone (32417) is connected to every user in the network. Of the 1543 users who cannot reach every other user, 1224 are users who provided just one friend in their web of trust.

The mean distance over all users is 4.56 (see table of Figure 6). One user has a mean distance of 9.67 (maximum) and another one (who has 1760 users in her web of trust) has a mean distance of 2.89 (minimum). These data show that the trust network is very connected and in part this depends also on the way we collected the dataset.

Figure 6 shows that, for users who wrote more than 100 reviews, other users are generally closer in the trust network (and hence a trust metric could infer trust more easily). For cold start users, other users are in general less close but anyway many of them are reachable so that it is possible to predict a trust value for them. It is important to note how even for users with just one friend (last



Userbase	#Users reach-ing all	Mean mean dist	Mean max dist
All users	32417	4.56	10.40
ratings<5	13106	4.87	10.71
ratings<10	18540	4.79	10.63
ratings<20	24397	4.71	10.54
ratings>50	2570	3.90	9.75
ratings>100	860	3.78	9.62
friends=1	8211	5.11	10.96

Fig. 6. The Figure shows the mean number of users at distance x for different class of users. The table shows, for different class of users, the users who are connected to every other user (in absolute value and in percentage), the mean of the mean distances and the mean of the max distances.

row of table of Figure 6), trust can be very effective. In fact, the mean distance averaged only over them is 5.11 (compared with 4.56 over all users). We believe that just adding one friend is a very easy and quick way to bootstrap the system. In this way, the system can be able to make good recommendations soon also to new users (who are generally also "cold start users" since they have provided few ratings as well).

We now summarize the results of the previous figures in Table 3 which provides the final argument. In fact it shows how, for a given user, the standard CF technique (Pearson correlation coefficient) on average allows to compute user similarity only on a small portion of the user base, precisely 160.73 over 49290 (less than 1%!). On the other hand, by propagating trust it could be possible to infer trust in the other users and use this value as an alternative weight when creating a recommendation. For the average user, in one trust step it is possible to cover 9.88 users (direct friends), in 2 steps 399.89 users (friends of friends), in 3 steps 4386.32 users and in 4 steps 16333.94 users. In computing these values we considered also the users who were not able to reach all the other users, for example the users who provided 0 friends.

The previous difference in coverage of the user base with the two techniques is even exacerbated in the case of "cold start users", users who expressed less than 5 ratings. The mean number of users against which Pearson is computable for this class of users is only 2.74 (0.0056% of the users). Instead propagating trust it is possible to reach 94.54 users in just 2 steps and 9120.78 in 4 steps.

In this Section we have analyzed how on a dataset of real users (*epinions.com*), using a trust metric in order to assign a trust weight to unknown users can potentially be much more effective than computing user similarity as traditional RSs do. In the next Section, we summarize the contributions and discuss which research lines the provided evidence opens up.

Mean number of Comparable users for all user					Mean number of Comparable users for cold start users				
propagating Trust (up to different distances)				using Pearson	propagating Trust (up to different distances)				using Pearson
1	2	3	4		1	2	3	4	
9.88	400	4386	16334	161	2.14	**94.54**	1675	9121	**2.74**

Table 3. Mean number of comparable users with different methods: Trust and Pearson correlation coefficient. For trust, we indicate the mean number of users reachable through some trust chain in at most x steps. For Pearson, we indicate the mean number of users against which Pearson coefficient is computable (i.e. overlap of at least 2 items). Both are computed over every user (even the ones with 0 ratings or 0 friends).

3 Contribution and Future work

This paper presents evidence that Recommender Systems [8] that incorporate trust can be more effective than systems based on classic techniques, such as Collaborative Filtering (CF). In particular, CF involves the computation of a user similarity measure (for example, Pearson Correlation Coefficient [2]). We have shown how this quantity, on average, is computable only against a very small portion of the user base and is, in most cases, a noisy and unreliable value because computed on the few items rated in commons by two users. Instead, trust-aware techniques can produce a trust score for a very high number of other users; the trust score of a user estimates the relevance of that users' preferences.

We have argued how even for "cold start users", i.e. users who provided few ratings and usually are the majority, trust propagation could be very effective (especially when compared with Pearson correlation coefficient that is almost always incomputable). The reported evidence opens the way to a number of research paths we briefly explore in the rest of the Section.

Trust metrics Of course not all the users at the same distance should be trusted the same. A trust metric [1], given a trust network, infers trust in unknown users. Studying different trust metrics is certainly important in a time when more and more data about real social networks are starting to become available in electronic format. Trust propagation is a compelling research line especially when applied to social networks who have weighted trust relationships (ex: A trusts B 0.7 and C 0.1). In this sense it would be very interesting to analyze also the web of distrust of *epinions.com* users. A special attention should deserve research about possible attacks from malicious users. We have a project and a Wiki for studying these issues[5].

Comparison of Trust and User Similarity The next steps will be to analyze what the relationships between Trust and User Similarity are: e.g., how often and how

[5] The Wiki (a writable web site) is at http://moloko.itc.it/trustmetricswiki/moin.cgi

much are they consistent? In which cases a trust metric suggests a user that happens to be very dissimilar or vice versa does not find a very similar user? A user reviews more items in common with her friends than with a random user? Based on this comparison, successful ways to combine User Similarity and Trust in order to decide a weight for the users' ratings can be proposed.

Recommendations computation The final goal is to produce recommendations using Trust-aware Recommender Systems and to compare these recommendations with traditional systems. We will analyze the performances of systems that use only trust (inferred with different trust metrics), only user similarity and combinations of the two.

User acceptance Human Computer Interaction studies [10] are needed to investigate the best ways to visualize the social network and how this could help the user in understanding the recommendation model and to control it.

References

1. Jennifer Golbeck, James Hendler, and Bijan Parsia. Trust networks on the Semantic Web. In *Proceedings of Cooperative Intelligent Agents*, 2003.
2. D. Goldberg, D. Nichols, B.M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.
3. J. Herlocker, J. Konstan J., A. Borchers, and J. Riedl. An Algorithmic Framework for Performing Collaborative Filtering. In *Proceedings of the 1999 Conference on Research and Development in Information Retrieval*, 1999.
4. J.L. Herlocker, J.A. Konstan, and J. Riedl. Explaining Collaborative Filtering Recommendations. In *Proc. of CSCW 2000.*, 2000.
5. D. Maltz and K. Ehrlich. Pointing the Way: Active Collaborative Filtering. In *Proc. of CHI-95*, pages 202–209, Denver, CO, 1995.
6. Paolo Massa. Trust-aware Decentralized Recommender Systems. Phd Proposal, 2003, University of Trento, http://sra.itc.it/people/massa/massa03trustaware.pdf.
7. Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.
8. P. Resnick and H.R. Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.
9. B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Application of dimensionality reduction in recommender systems–a case study. in ACM WebKDD Workshop, 2000.
10. K. Swearingen and R. Sinha. Beyond algorithms: An HCI perspective on recommender systems. in ACM SIGIR 2001 Workshop on Recommender Systems, New Orleans, Lousiana, 2001.
11. Jeffrey Zaslow. If TiVo Thinks You Are Gay, Here's How to Set It Straight. The Wall Street Journal, 26 November 2002.