Trust-Aware Collaborative Filtering for Recommender Systems

Paolo Massa and Paolo Avesani

ITC-iRST
Via Sommarive 14 - I-38050 Povo (TN) - Italy
{massa,avesani}@itc.it

Abstract. Recommender Systems allow people to find the resources they need by making use of the experiences and opinions of their nearest neighbours. Costly annotations by experts are replaced by a distributed process where the users take the initiative. While the collaborative approach enables the collection of a vast amount of data, a new issue arises: the quality assessment. The elicitation of trust values among users, termed "web of trust", allows a twofold enhancement of Recommender Systems. Firstly, the filtering process can be informed by the reputation of users which can be computed by propagating trust. Secondly, the trust metrics can help to solve a problem associated with the usual method of similarity assessment, its reduced computability. An empirical evaluation on *Epinions.com* dataset shows that trust propagation can increase the coverage of Recommender Systems while preserving the quality of predictions. The greatest improvements are achieved for users who provided few ratings.

1 Introduction

Recommender Systems (RS) [12] are widely used online (e.g.: in *Amazon.com*) to suggest to users items they may like or find useful. Collaborative Filtering (CF) [4] is the most widely used technique for Recommender Systems. The biggest advantage of CF over content-based systems is that explicit content description is not required. Instead CF only relies on opinions expressed by users on items. Instead of calculating the similarity between an item description and a user profile as a content-based recommender would do, a CF system searches for similar users (neighbours) and then uses ratings from this set of users to predict items that will be liked by the current user.

In contrast with a centralized content-based recommender, the CF technique distributes the work load involved in evaluating and marking up the items in its data base. For this reason, it has obvious advantages over a content based system where the knowledge expense to annotate millions of items is very high.

However CF suffers some weaknesses: problems with new users (cold start), data sparseness and difficulty in spotting "malicious" or "unreliable" users.

We propose to extend RS with trust-awareness: users are allowed to also explicitly express their web of trust [5] (i.e., users they trust about ratings and

R. Meersman, Z. Tari (Eds.): CoopIS/DOA/ODBASE 2004, LNCS 3290, pp. 492–508, 2004.

opinions on items). Using a technique to propagate trust throughout the global trust network, Trust-aware Recommender Systems are able to overcome the previously mentioned weaknesses. In fact, trust allows us to base recommendations only on ratings given by users trusted directly by the current user or indirectly, for example trusted by another trusted user. In this way it is possible to cut out malicious users who are trying to influence recommendation accuracy. Moreover, in RSs users typically have rated only a small portion of the available items, but user similarity is computable only on the few users who have rated items in common. This fact greatly reduces the number of potential neighbours, whose ratings are combined to create recommendations for the current user. This problem is exacerbated for "cold start users", users who have just expressed a few ratings. Instead, by allowing a user to rate other users, the system can quickly make recommendations using the explicit neighbour set. This means the new user will soon receive good recommendations and so she has an incentive to keep using the system and to provide more ratings.

The contributions of this paper are three-fold:

- We identify specific problems with current Collaborative Filtering RSs and propose a new solution that addresses all of these problems.
- We precisely formalize the domain and the architecture of the proposed solution: namely Trust-Aware Recommender Systems.
- We conduct experiments on a large real dataset showing how our proposed solution increases the coverage (number of ratings that are predictable) while not reducing the accuracy (the error of predictions). This is especially true for users who have provided few ratings.

The rest of the paper is structured as follows: firstly we introduce Recommender Systems and their weaknesses (Section 2). In Section 3 we discuss trust from a computational point of view and we argue how trust-aware solutions can overcome the weaknesses described in the previous section. Section 4 is devoted to formalizing the environment in which Trust-aware Recommender Systems can operate while Section 5 describes the architecture of the framework and its components. Our experiments are presented in Section 6 and Section 7 provides a discussion of the results. Section 8 concludes the paper with a discussion of future work.

2 Motivation

Recommender Systems (RS) [12] 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 suggests items that are similar to the ones the current users has shown a preference for in the past. Content-based matching requires a representation of the items in terms of features. For machine-parsable items (such as news or papers), such a representation can be created automatically but for other kind of items (such as movies and songs), it must be manually inserted by human editors. This activity is expensive, time-consuming,

error-prone and highly subjective. For this reason, content-based systems are not suitable for dynamic and very large environments, where items are millions and are inserted in the system frequently.

Collaborative Filtering (CF) [4], on the other hand, collects opinions from users in the form of ratings on items. The recommendations produced are based only on the opinions of users similar to the current user (neighbours). The advantage over content-based RS is that the algorithm doesn't need a representation of the items in terms of features but it is based only on the judgments of the user community.

Collaborative Filtering stresses the concept of community, where every user contributes with her ratings to the overall performances of the system. We will see in the following how this simple yet very powerful idea introduces a new concern about the "quality" and "reliability" of every single rating. In the rest of this paper we concentrate on RSs based on CF.

The traditional input to a CF algorithm is a matrix in which rows represents users and columns items. The entry at each element of the matrix is the user's rating of that item. Figure 1 shows such a matrix.

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

	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

In order to create recommendations for the current users, CF performs three steps:

- *It compares the current user's ratings against every other user's ratings.* CF computes a similarity value for every other user, where 1 means totally similar and -1 totally dissimilar. Usually the similarity measure is the Pearson correlation coefficient, but any other could be used [7]. The coefficient is computable only if there are items in common rated by both users. If this situation does not occur (as it is often the case), two users are not comparable.
- *Based on the ratings of the most similar users (neighbours), it predicts the rating the current user would give to every item she has not yet rated.*
- *It suggests to the user the items with highest predicted rating.*

The standard CF schema is simple but very effective, however it has some weaknesses which we discuss in the rest of the section.

RS are computationally expensive. The CF algorithm we have described is typical of a lazy, instance based learning algorithm. Such algorithms suffer can

be computationally very expensive at query time, since they need search all the user profiles to find the best set of neighbours. This problem means that current RS cannot scale to large environments with millions of users and billions of items (for example, the envisioned Semantic Web [1]). This is also a very slow step, in the sense it may take several minutes to find neighbours of one user. For this reason, it is not feasible to do it when a recommendation request is made by the user and hence this should be done periodically offline. However this means that recommendations are not always up to date and that user ratings do not take effect immediately.

User similarity is computable only against few users. The first step suffers another problem. In order to be able to create good quality recommendations, RSs should be able to compare the current user against every other user with the goal of selecting the best neighbours with the more relevant item ratings. This step is mandatory and its accuracy affects the overall system accuracy: failing to find "good" neighbours will lead to poor quality recommendations. However, since the ratings matrix is usually very sparse because users tend to rate few of the millions of items, it is often the case that two user don't share the minimum number of items rated in common required by user similarity metrics for computing similarity. For this reason, the system is forced to choose neighbours in the small portion of comparable users and will miss other non-comparable but relevant users. This problem is not as serious for users with hundreds of ratings but for users with few ratings. However it can be argued that it is more important (and harder) for an RS to provide a good recommendation to a user with few ratings in order to invite her to provide more ratings and keep using the system than to a user with many ratings that is probably already using the system regularly.

Easy attacks by malicious insiders. Recommender Systems are often used in e-commerce sites (for example, in *Amazon.com*). In those contexts, being able to influence recommendations could be very attractive: for example, an author may want to "force" *Amazon.com* to always recommend the book she wrote. However, subverting standard CF techniques is very easy [10]. The simplest attack is the copy-profile attack: the attacker can copy the ratings of target user and the system will think the attacker is the most similar user to target user. In this way every additional item the attacker rates highly will probably be recommended to the target user. Since currently RSs are mainly centralized servers, creating a "fake" identity is a time-consuming activity and hence these attacks are not currently heavily carried on and studied. However we believe that, as soon as the publishing of ratings and opinions becomes more decentralized (for example, with Semantic Web formats such as RVW [2] or FOAF [3]), these types of attacks will become more and more an issue. Basically, creating such attacks will become as widespread as spam is today, or at least as easy.

3 Web of Trust

In decentralized environments where everyone is free to create content and there is no centralized quality control entity, evaluating the quality of this content becomes an important issue. This situation can be observed in online communities (for example, slashdot.org in which millions of users posts news and comments daily), in peer-to-peer networks (where peers can enter corrupted items), or in marketplace sites (such as *eBay.com*, where users can create "fake" auctions).

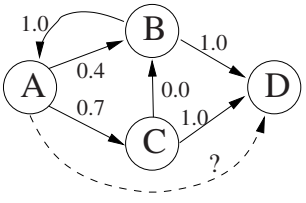

Fig. 1. Trust network. Nodes are users and edges are trust statements. The dotted edge is one of the undefined and predictable trust statements.

On these environments, it is often a good strategy to delegate the quality assessment task to users themselves. For example, by asking users to rate items, CF uses such a quality assessment approach.

Similarly, the system can ask the users to rate the other users: in this way, a user can express her level of trust in another user she has interacted with. For example, in Figure 1, user A has issued a trust statement in B (with value 0.4) and in C (with value 0.7); hence B and C are in the web of trust of A.

The webs of trust of all the users can then be aggregated in a global trust network, or social network (Figure 1), and a graph walking algorithm be used to predict the "importance" of a certain node of the network. This intuition is exploited, for example, by PageRank [11], one of the algorithm powering the search engine *Google.com*. According to this analysis, the Web is a network of content without a centralized quality control and PageRank tries to infer the authority of every single page by examining the structure of the network. PageRank follows a simple idea: if a link from page A to page B represents a positive vote issued by A about B, then the global rank of a page depends on the number (and quality) of the incoming links.

The same intuition can be extended from web pages to users: if users are allowed to cast trust values on other users, then these values can be used to predict the trustworthiness of unknown users. For example, the consumer opinion site *Epinions.com*, where users can express opinions and ratings on items, also allows users to express their degree of trust in other users. Precisely, the *epinions.com* FAQ suggests a user should add in her web of trust "reviewers whose reviews and ratings they have consistently found to be valuable"[1]

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

Using explicit trust statements, it is possible to predict trust in unknown users by propagating trust; precisely, if A trusts B and B trusts D, it is possible to infer something about how much A could trust D (the dotted arrow in Figure 1). It is important to underline what is perhaps an obvious fact: trust is subjective, the same user can be trusted by someone and distrusted by someone else. In Figure 1, for example, we can see how B is trusted by A as 0.4 and by C as 0. It is important to take this into account when predicting trustworthiness. Another self-evident fact is that trust is not symmetric (see users A and B, for instance).

Trust metrics [3,14,8] have precisely the goal of predicting, given a certain user, trust in unknown users based on the complete trust network. For example, in Figure 1, a trust metric can predict the level of trust of A in D.

Trust metrics can be divided into local and global. Local Trust metrics take into account the very personal and subjective views of the users and end up predicting different values of trust in other users for every single user. Instead global trust metrics predict a global "reputation" value that approximates how the community as a whole considers a certain user. In this way, they don't take into account the subjective opinions of each user but average them across standardized global values. PageRank [11], for example, is a global metric. However, in general, local trust metrics are computationally more expensive because they must be computed for each user whereas global ones are just run once for all the community.

In the following, we argue that trust-awareness can overcome all the weaknesses introduced in Section 2. Evidence supporting this claim will be given in Section 6 and 7. Precisely, trust propagation allows us to compute a relevance measure, alternative to user similarity, that can be used as an additional or complementary weight when calculating recommendation predictions. In [9] we have shown how this *predicted trust* value, thanks to trust propagation, is computable on much more users than the *user similarity* value.

CF systems have problems scaling up because calculating the neighbours set requires computing User Similarity of current user against every other user. However, we can significantly reduce the number of users which RS has to consider by prefiltering users based on their "predicted trust" value. For example, it would be possible to consider only users at a small distance in social network from current user or considering only users with a predicted trust higher than a certain threshold.

Moreover, trust metrics can be attack-resistant [8], i.e. they can be used to spot malicious users and to only take into account "reliable" users and their ratings. It should be kept in mind, however, that there isn't a global view of which user is "reliable" or "trustworthy" so that, for example, a user can be considered trustworthy by one user and untrustworthy by another user.

In the process to identify malicious users, a very relevant role can play the concept of distrust. However, studying the meaning of distrust and how to computationally exploit it is very recent topic (we are aware of just one paper researching this [6]) and much work is needed in order to fully understand it.

Moreover the dataset we run experiments on (see Section 6) did not contain distrust information.

A more detailed description of RS weaknesses and how Trust-awareness can alleviate them can be found in the paper by Massa et al. [9].

4 Formal Domain Definition

In this section we precisely formalize the environment in which Trust-aware Recommender Systems can operate.

The environment is composed by:

- A set P of n uniquely identifiable peers.

  $$P = \{p_1, p_2, p_3, ..., p_n\}$$

  In this abstract domain definition, we use the term "peer" because the proposed framework can work for users of an online community but also for intelligent web servers (willing to trade and share resources), nodes of a peer-to-peer network, software agents or every possible conceivable independent entity able to perform some actions. A peer must be uniquely identifiable. For instance on the web, a reasonable unique identifier for peers could be an URI (Uniform Resource Identifier).
- A set I of m uniquely identifiable items.

  $$I = \{i_1, i_2, i_3, ..., i_m\}$$

  For items identifiers, we can think about globally agreed ones (such as ISBN for books, for instance) or we can use some hashing of the content, if digital, or of the item description to produce a unique id.
- n sets of Trust Statements. Every peer is allowed to express a trust value in every other peer. This should represent how much a peer consider valuable the ratings of another peer. Every peer's trust statements can be formalized in a trust function whose domain is P and whose codomain is $[0, 1]$ where 0 means total distrust and 1 total trust. A missing value (function not defined) represents the fact that the peer does not express a trust statement about that peer, probably because it didn't have a direct evidence for deciding about that peer's trustworthiness.

  $$t_{p_i} : P \to [0, 1] \cup \perp \qquad \text{Trust function of peer } p_i$$

  For example, $t_{p_1}(p_2) = 0.8$ means that peer p_1 issued a trust statement expressing its degree of trust in peer p_2 as 0.8, a high trust value.
  In this model we do not consider the timing of trust statements, so that, for instance, if a peer expresses again another trust statement about the same user (probably updating the value based on last interactions), we just override the previous value.

– n sets of Ratings. Every peer is allowed to express a rating on every item. Every peer's ratings can be formalized in a rating function whose domain is I and the codomain is $[0, 1]$ where 0 means total dislike and 1 maximum appreciation. A missing value (function not defined) means the user did not rate the item.

$$r_{p_i} : I \rightarrow [0, 1] \cup \perp \qquad \text{Rating function of peer } p_i$$

For example, $r_{p_1}(i_1) = 0.1$ means that peer p_1 rated item i_1 as 0.1, a low rating expressing its partial dislike for the item.
In this case too, we simply consider the last rating given by one user to the same item.

Discrete ratings scales (for example the integers from 1 to 5) for trust statements and ratings can be easily mapped in the $[0, 1]$ interval.

Similar models were proposed as Open Rating Systems [5] and in the context of Semantic Web Recommender Systems [13].

It is worth underlining how the trust and rating functions would always be very sparse (i.e., undefined for the largest part of the domain). This is so because no peer can reasonably experience and then rate all the items in the world (for example, all the books or songs). The same is true for trust: no peer can reasonably interact with every other peer (think of a community of 1 billion peers) and then issue a trust statement about them.

At present, this kind of environment has been created by some online companies, for example *epinions.com* or *amazon.com*. Likewise, many other environments are moving in this direction, for example, as we already mentioned, peer-to-peer networks, open marketplaces, and notably the Semantic Web [1] whose goal is to create a web of hyperlinked pages "understandable" automatically by machines. To this extend, two very interesting and promising semantic formats are FOAF [3] (for expressing friends and trusted users) and RVW [2] (for expressing reviews of items).

5 Trust-Aware Recommender Architecture

In this section we present the architecture of our proposed solution: Trust-aware Recommender Systems. Figure 2 shows the different modules (black boxes) as well as input and output matrices of each of them (white boxes).

The overall system takes as input the trust matrix (representing all the community trust statements) and the ratings matrix (representing all the ratings given by users to items) and produces, as output, a matrix of predicted ratings that the users would assign to the items. These matrix is used by the RS for recommending the most liked items to the user: precisely, the RS selects, from the row of predicted ratings relative to the user, the items with highest values. Of course, the final output matrix could be somehow sparse, i.e. having some cells with missing values, when the system is not able to predict the rating that

the user would give to the item. Actually the quantity of predictable ratings is one of the possible evaluation strategies.

Let us now explain in detail every single module. First, we define the task of the module and then we describe the algorithm we chose for the experiments. However the architecture is modular and hence different algorithms can be plugged for the different modules.

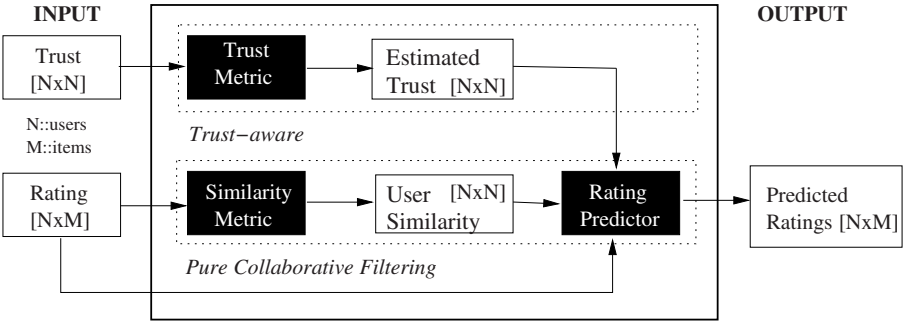

Fig. 2. Trust-Aware Recommender Systems Architecture.

Trust metric module. The Trust Metric Module takes as input the trust network (representable as a $N \times N$ trust matrix) and exploits trust propagation in order to predict, for every user, how much she could trust every other user. In this way, it produces an Estimated Trust matrix. The value in the cell i, j (if present) represents how much the metric predict peer p_i may trust peer p_j. This quantity can be used as a weight representing how much the user's ratings should be considered when creating a recommendation.

As already stated, trust metrics can be classified in local and global. In our framework, a global trust metric (for example, PageRank [11]) produces an Estimated Trust matrix with all the rows equal, meaning that the estimated trust in a certain user (column) is the same for every user (row).

While there are some attempts to propose trust metrics [3,14,8], this research topic is very recent and there aren't thorough analysis of which metrics perform better in different scenarios. Since the goal of this paper is to show that trust-awareness is useful in improving Recommender Systems, we use a simple trust metric in our experiments. More sophisticated ones can be deployed in our framework very easily.

We use the following local trust metric: given a source user, it assigns to every other user a predicted trust based on her minimum distance from the source user. Precisely, assuming trust is propagated up to the maximimum propagation distance d, a user at distance n from source user will have a predicted trust value of $(d - n + 1)/d$. Users that are not reachable within the maximum propagation distance have no predicted trust value (and cannot become neighbours). Our trust metric choice is guided also by the fact that the dataset we use for

experiments does not have weighted trust statements but only full positive trust statement: we only have access to "peer p_i trusts p_j as 1" (see a description of dataset and experiments on Section 6). As an example, we analyze the trust network in Figure 1, but considering the trust statements values as 1. We predict trust values for user A and choose 4 as maximum propagation distance: in this case, our trust metric would assign to users at distance 1 (B and C) a predicted trust value of $(4-1+1)/4 = 1$ and to users at distance 2 (D) a predicted trust value of $(4-2+1)/4 = 0.75$. In this way, we adopt a linear decay in propagating trust: users closer in the trust network to the source user have higher predicted trust.

Similarity metric. Computing the similarity of current user against every other user is one of the standard steps of Collaborative Filtering techniques. Its task is to compute the correlation between two users (represented as vectors of ratings), producing the output $n \times n$ User Similarity matrix in which ith row contains the similarity values of ith user against every other user. The correlation value is used in next steps as a weight for the user ratings, according to the intuition that, if a user rates in a similar way to current user, then her ratings are useful for predicting the ratings of the current user. The most used technique is Pearson Correlation Coefficient [7].

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

Note that this coefficient can be computed only on overlapping items. Moreover, if 2 users only have one item rated by both, then the coefficient is not meaningful. Hence, for a user, it is possible to compute the correlation coefficient only in users who share at least 2 co-rated items and this are usually a small portion as we described in [9]. In the experiments (see Section 6), we follow the most used strategy of keeping only positive similarities values because users with a negative correlation are dissimilar to current user and hence it is better to not consider their ratings.

Rating predictor. This step is the classical last step of Collaborative Filtering [7]. The predicted rating of item i for the current user a is the weighted sum of the ratings given to item i by the k neighbours of a.

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

Neighbours can be taken from the User Similarity matrix or from the Estimated Trust matrix and the weights $w_{a,u}$ are the cells in the chosen matrix. For example, in the first case, the neighbours of user i are in the ith row of the User Similarity matrix.

Another option is to combine the two matrices in order to produce an output matrix that embeds both the information (user similarity and estimated trust). Usually, these matrices are very sparse so that this strategy could be useful in

reducing sparseness and hence providing more neighbours for every single user. However, the goal of this paper is to evaluate separately the possible contributions of trust-awareness in RS and not to propose a combination technique that would require a dedicated evaluation.

In this section we have explained the architecture of Trust-aware Recommender Systems. In the following section, we present the experiments we have conducted and the dataset we have used.

6 Experiments

In this section we present experimental results that provide evidence supporting our claim that trust-awareness improve the performances of RSs.

The section is structured as follows: firstly we introduce details about the dataset we used, then we explain in detail the experiments we have run and discuss the chosen evaluation strategy.

We collected a dataset with the required features discussed in Section 4 from an online community, *epinions.com*. *Epinions.com* is a consumers opinion site where users can review items (such as cars, books, movies, software, ...) and also assign them numeric ratings in the range 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"[2] and their *Block list*, i.e. a list of authors whose reviews they find consistently offensive, inaccurate, or not valuable. We collected the dataset by crawling the *epinions.com* site on November 2003. We stored, for every user, the rated items (with the numeric rating) and the trusted users (friends). Note that we could only access the publically available positive trust statements ($t_{p_i}(p_j) = 1$), and not the private Block lists.

The collected dataset consists of 49290 users who rated a total of 139738 different items at least once. 40169 users rated at least one item. The total number of reviews is 664824. The sparseness of the collected dataset is hence more than 99.99%. The total number of trust statements is 486985. More details about the dataset (with, for instance, standard deviations and distributions of rating and trust statements) and the way we collected it can be found in [9].

We should underline how the majority of users are what are termed "cold start users" [9], users who provided few ratings. For instance in our collected dataset more than half of the users (52.82%) provided less than 5 ratings. We will see in the following how it is precisely with these users that traditional CF systems tend to perform poorly. We will also see that a trust-aware solution is especially powerful for these users.

We now explain the different experiments we have run on the *epinions.com* dataset. We have instantiated the architecture presented in Figure 2 in order to compare the contributions of the trust metric and of the user similarity metric to the performances of the system. Hence we have run separately two techniques: a pure Collaborative Filtering strategyand a Trust-aware one.

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

For the Trust Metric technique, we have used the one introduced in Section 5. We performed different experiments with different maximum propagation distances, precisely 1, 2, 3 and 4. Choosing 1 as max propagation distance means considering, for every user, only the users explicitly inserted in the web of trust (friends, in the *epinions.com* vocabulary). Since the further away the user is from current user, the less reliable is the inferred trust value, we choose to run experiments propagating trust only up to distance 4. As expected, increasing the propagation distance implies that the technique is able to consider, on average, an increasing number of potential neighbours for every single user. Intuitively, the higher the propagation distance, the less sparse the resulting Predicted Trust matrix.

For the Similarity Metric technique, we have used the Pearson Correlation coefficient (Equation 1), the most commonly used similarity metric in RSs [7].

For the Rating Predictor module, we have used the standard CF technique (Equation 2). In one experiment we have generated the neighbourhood set using the User Similarity matrix, and in the others we used the Estimated Trust matrices.

In order to compare the performances of the two different approaches (pure CF and trust-aware), we need to choose a Recommender System evaluation technique. We use *leave-one-out* technique with Mean Absolute Error (MAE) as our error metric, since it is the most appropriate and useful for evaluating prediction accuracy in offline tests [7]. Leave one out involves hiding a rating and then trying to predict it. The predicted rating is then compared with the real rating and the difference (in absolute value) is the prediction error. Averaging this error over every prediction gives the overall MAE.

Another important way to discriminate between different recommender techniques is coverage. The RS may not be able to make predictions for every item. For this reason, it is important to evaluate also the portion of ratings that an RS is able to predict (*ratings coverage*).

However this quantity is not always informative about the quality of an RS. In fact, it is often the case that an RS is successful in predicting all the ratings for a user who provide many ratings and perform poorly for a user who has rated few items. For example, let us suppose that we consider one user with 100 ratings and 100 users with 1 rating. In this case, a probable situation is the following: the RS is able to predict all the 100 ratings given by the "heavy rater" and none of the other ones. So we have 100 predicted ratings over 200 possible ones, corresponding to 50% of the ratings, but we only have one "satisfied" user over 101 users, corresponding to less than 1%! For this reason, we also compute the *users coverage*, defined as the portion of users for which the RS is able to predict at least one rating.

A similar argument applies for Mean Absolute Error as well. Usually RSs produce small errors for "heavy raters" and higher ones for "cold start users". But, since heavy raters provide many ratings, in computing MAE, we are going to count these small errors many times, while the few big errors made for cold start users count few times. For this reason, we define another measure we call

Mean Absolute User Error (MAUE), for which we first compute the mean error for each user and then we average these user errors over all the users. In this way every user is taken into account once and a cold start user is influential as much as a heavy rater.

Since our argument is that trust-awareness is especially useful for cold start users, in the next section we will analyze the performances (coverage and error) of the different techniques, also focusing particularly on users who provided few ratings. This analysis is important because these users make up more than 50% of our dataset. In particular, we constructed three different views, considering only users who provided 2, 3 or 4 ratings.

In this section we discussed the dataset we use, our experiments and the chosen evaluation technique. In the next section, we will present and discuss the results.

7 Discussion of Results

The results of our experiments are summarized in Table 2. The rows of the table represent the evaluation measures we computed for the different techniques. The reader is referred to previous Section for an explanation of the different evaluation measures: ratings coverage, users coverage, MAE and MAUE. The different techniques we used in our experiments are *UserSim* and *Trust-x*. *UserSim* refers to the pure Collaborative Filtering strategy (bottom dotted box in Figure 2) while *Trust-x* refers to the trust-aware strategy (top dotted box), where x is the max propagation distance.

The columns of the table represent different views of the data. In the first column (labelled "ALL") we show the evaluation measures computed over all the users and this gives a picture of the average performances of the different techniques. Instead in the next columns we concentrate on cold start users (which are a large portion of the users). For example, in the second column (labelled "2"), only the users who expressed 2 ratings are considered. For every view of the data (i.e. column), we also indicate the number of users who satisfy the conditions on number of expressed ratings ("User Population Size"). For example, the users who expressed 2 ratings are 3937, almost 10% of the users. For every view of the data, we also present the mean number of users in the web of trust of the considered users. This is done in order to show the quantity of information available to the different techniques (number of ratings for *UserSim*, mean number of friends for *Trust-x*).

We now discuss results of experiments presented in Table 2.

User Similarity performs well with heavy raters and poorly with cold start users. As we have shown in [9], for the heavy raters (i.e. users who rated many items), the Pearson coefficient is computable on many other users that are potential neighbours. This means there is a high probability that some of these neighbours have rated the item under prediction. The opposite situation occurs with cold start users (users who rated few items). This is represented by the fact that

Table 2. Results of experiments. Rows represents different evaluation measures we collected for the different evaluated techniques. Columns represents different views of the data (e.g., in the column labelled "2", we present evaluation measures computed only on users who have rated exactly 2 items). For every column we also show the number of users in the specific view and the mean number of users in their webs of trust (friends).

# Expressed Ratings		ALL	2	3	4
User Population Size		40169	3937	2917	2317
Mean Web of Trust Size		9.88	2.54	3.15	3.64
Ratings	UserSim	**51%**	N/A	4%	**8%**
Coverage	Trust-1	**28%**	10%	**11%**	12%
	Trust-2	60%	23%	26%	31%
	Trust-3	74%	39%	45%	51%
	Trust-4	**77%**	45%	**53%**	59%
Users	UserSim	**41%**	N/A	6%	**14%**
Coverage	Trust-1	**45%**	17%	**25%**	32%
	Trust-2	56%	32%	43%	53%
	Trust-3	61%	46%	57%	64%
	Trust-4	**62%**	50%	**59%**	**66%**
MAE	UserSim	**0.843**	N/A	1.244	**1.027**
	Trust-1	0.837	0.929	**0.903**	0.840
	Trust-2	0.829	1.050	0.940	0.927
	Trust-3	0.811	1.046	0.940	0.918
	Trust-4	**0.805**	1.033	**0.926**	0.903
MAUE	UserSim	**0.939**	N/A	1.319	1.095
	Trust-1	0.853	0.942	0.891	0.847
	Trust-2	0.881	1.041	0.935	0.905
	Trust-3	0.862	1.033	0.942	0.915
	Trust-4	0.850	1.019	0.927	0.899

UserSim is able to cover 51% of the ratings (340906 out of 664824) but only 41% of the users (16378 out of 40169). The reason is that in the small percentage of users for which a prediction is possible, there are mainly heavy raters that provided a big percentage of the ratings. This is especially relevant if compared with *Trust-1* that has an opposite behaviour: in fact, it is able to cover only 28% of the ratings (187513 out of 664824) but 45% of the users (17897 out of 40169). This means *Trust-1* is able to predict at least one rating for many users but, for every user, it is able to predict a small portion of the ratings (on average, almost 10). We have already stated that, while predicting for heavy raters is reasonably easy, the real challenge for RS is in making recommendations to new users with few ratings so that they find useful the system and keep using it. *Trust-2*, *Trust-3*, *Trust-4* significantly increase both the users coverage and the ratings coverage.

Another fact confirming that *UserSim* works well with heavy raters and not with cold start users is the difference between MAE and MAUE. Averaging the

prediction error over every rating gives a MAE of 0.843, while, considering the average error for every user, we obtain a MAUE of 0.939. This latter measure tells us that the error is higher on predictions for cold start users: in fact, they expressed few ratings and hence the high errors produced for their predictions contribute less to the overall Mean Absolute Error computed over every predictable rating. Instead, for *Trust-x*, MAE and MAUE values are in general close, meaning that errors (and performances) are more similar for every type of user and more uniformly distributed.

On average, Trust-x achieves better coverage without loss of accuracy. We now compare the global MAE for the different techniques over all the users (third row, first column of Table 2). The highest error is obtained in the experiment with *UserSim* (0.843). Instead every *Trust-x* technique has smaller error. Every additional allowed trust propagation step decreases the error, however the difference in error at different max propagation distances is small (the MAE of *Trust-4* is 0.805). Hence we can say that, even if *Trust-x* performs better than *UserSim*, the decrease in error is small. On the other hand, coverage is significantly higher for Trust-aware strategies; for example propagating trust up to distance 4 (*Trust-4*), we are able to cover 77% of the ratings and to make at least a prediction for 62% of the users. These values were, respectively, 51% and 41% for *UserSim* technique.

For cold start users, Trust-x achieves also better accuracy. As we have already argued, the most important and challenging users for an RS are the cold start users, users who provided few ratings. For this reason we analyze in detail the performances (both in term of coverage and error) of the different techniques only considering users who rated a small number of ratings.

For example, in the second column of Table 2, we consider only users who have rated 2 items. It is worth noting that there are 3937 out of 40169 users (10 %) who provided at least one rating and hence a significant portion. For users who have rated just 2 items, no prediction with *UserSim* is possible. In fact, since *leave-one-out* hides one rating, the Pearson Correlation coefficient is not computable and hence every user has zero neighbours.

It is important to note that, Trust-aware techniques and pure CF ones use different input (respectively, trust statements and ratings). However, we must compare their performances when they use comparable quantity of input information. Considering that *leave-one-out* hides one rating but does not affect the number of trust statements, we compare *UserSim* computed over users who expressed n ratings with *Trust-x* computed over users who expressed $n - 1$ trust statements (friends). In particular, in the following we compare the performances of *UserSim* over users with 4 ratings (because of leave one out, 3 ratings are used as input) with performances of *Trust-x* over users with 3 ratings (on average, 3.15 trust statements are available and used as input).

Considering coverage over ratings, *UserSim* is able to cover 8% of the ratings (750 out of 9268) and *Trust-1* 11% (976 out of 8751). A significant improvement in the ratings coverage is obtained increasing the trust propagation horizon, for

example *Trust-4* is able to cover 53% of the ratings. Considering coverage over users, we observe a similar pattern with *UserSim* able to predict at least a rating for 14% of the users (318 out of 2317); the percentages are 25% (728 out of 2917) for *Trust-1* and 59% for *Trust-4*.

Also when considering the error on predictions, *Trust-x* techniques improve over *UserSim* technique. For example, *UserSim* produces a Mean Absolute Error (MAE) of 1.027 compared to a MAE of 0.903 for *Trust-1*. *Trust-4* presents a slight increase in MAE (0.926) if compared with *Trust-1*. This is so because *Trust-4* is able to predict much more ratings (4618 versus 976 of *Trust-1*), i.e. has a much higher ratings coverage, as already noted. These latter results are important: with a Trust-aware technique we are able to generate a prediction for more that half of the users with 3 ratings, while keeping the error low. Similar results can be observed for users who rated 2 and 4 items. Note that, as expected, for every technique, the average error for users with 2, 3 or 4 ratings is higher than the average error over all the users. This is because it is more difficult to predict ratings for users with small rating profiles.

In order to bootstrap the system for new users, it is better to promote the acquisition of few trust statements. Our experiments suggest that for cold start users (who are a sizeable portion of the dataset), a few trust statements (and the use of our trust metric) can achieve a much higher coverage and reduced error with respect to the comparable amount of rating information. For example, the last column of Table 2 shows that for users with 4 ratings (that have on average 3.64 friends), the best Trust-aware technique is able to make a prediction for 66% of the users while the technique based on User Similarity only to 14% and with a higher error. This fact suggests that, in order for an RS to be able to provide recommendations to a new user, collecting few trust statements is more effective, both in term of coverage and error, than collecting an equivalent number of ratings.

8 Conclusions and Future Work

The goal of this paper is to analyze the potential contribution of Trust Metrics in increasing the performances of Recommender Systems. We have argued how Trust-awareness can solve some of the traditional problems of RSs and we have proposed an architecture for Trust-aware Recommender Systems. We have shown, through a set of experiments on a large real dataset, that Trust Metrics increase the coverage (number of predictable ratings) and decrease the error when compared with traditional Collaborative Filtering RS. This effect is especially evident for new users who have rated few items. Hence, based on the experiments results, we are able to suggest a RS bootstrapping strategy for new users: collecting few trust statements is more effective than collecting an equivalent amount of ratings.

Our future work goes in several directions. We need to evaluate the performances of different trust metrics in RSs. In particular we are interested to

test whether local trust metrics [3,14] performs better than global ones (for instance, PageRank [11]). Another direction deals with distrust [6], i.e. negative trust statements. We have recently obtained the complete and anonymized *epinions.com* dataset containing also the users' Block list. Considering distrust is a very recent topic and in our context it is especially useful in order to conduct a deep evaluation of possible RSs attacks [10] and to test wheter trust-aware solutions are able to detect malicious or not reliable users.

Our final goal is to create Trust-Aware Recommender Systems. In this paper we kept cleanly separated the technique using trust information (*Trust-x*) and the technique using rating information (*UserSim*) and conducted a comparative analysis. However in future we want to propose and study algorithms for combining trust and rating information, in order to take the advantages of both strategies. A specific analysis will be made also on understanding when this information are conflicting (for example, when a user predicted as trustable issues dissimilar ratings on the same items or viceversa).

References

1. T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 2001.
2. A. Eaton. RVW module for syndicating and aggregating reviews, 2004. http://www.pmbrowser.info/rvw/0.2.
3. J. Golbeck, J. Hendler, and B. Parsia. Trust networks on the Semantic Web. In *Proceedings of Cooperative Intelligent Agents*, 2003.
4. 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.
5. R. Guha. Open rating systems. Technical report, Stanford University, CA, USA, 2003.
6. R. Guha, R. Kumar, P. Raghavan, and A. Tomkins. Propagation of trust and distrust. In *Proceedings of WWW2004*, 2004.
7. 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.
8. R. Levien. *Advogato Trust Metric*. PhD thesis, UC Berkeley, USA, 2003.
9. P. Massa and B. Bhattacharjee. Using trust in recommender systems: an experimental analysis. In *Proc. of 2nd Int. Conference on Trust Management*, 2004.
10. M. O'Mahony, N. Hurley, N. Kushmerick, and G. Silvestre. Collaborative recommendation: A robustness analysis. In *Proceedings of Int'l Semantic Web Conf. (ISWC-03)*, 2003.
11. L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford, USA, 1998.
12. P. Resnick and H.R. Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.
13. C. Ziegler. Semantic web recommender systems. In *Joint ICDE/EDBT Ph.D. Workshop*, 2004.
14. C. Ziegler and G. Lausen. Spreading activation models for trust propagation. In *IEEE International Conference on e-Technology, e-Commerce, and e-Service (EEE'04)*, 2004.