Trust Metrics in Recommender Systems

Paolo Massa and Paolo Avesani

1 Introduction

Recommender Systems (RS) [25] have the goal of suggesting to every user the items that might be of interest for her. In particular, RSs based on Collaborative Filtering (CF) [2] rely on the opinions expressed by the other users. In fact, CF tries to automatically find users similar to the active one and recommends to this active user the items liked by these similar users. This simple intuition is effective in generating recommendations and is widely used [25].

However, RSs based on CF suffer some inherent weaknesses that are intrinsic to the process of finding similar users. In fact, the process of comparing two users with the goal of computing their similarity involves comparing the ratings they provided for items. And in order to be comparable, it is necessary that the two users rated at least some of the same items. However in a typical domain, for example in the domain of movies or books, the number of items is very large (in the order of the millions) while the number of items rated by every single user is in general small (in the order of dozens or less). This means that it is very unlikely two random users have rated any items in common and hence they are not comparable. Another important and underconsidered weakness is related to the fact that RS can easily be attacked by creating ad hoc user profiles with the goal of being considered as similar to the target user and influence the recommendations she gets. Other weaknesses refer to the fact that RSs are sometimes reported as difficult to understand and control and to the fact that most of the current real deployments of RSs have been as centralized servers, which are not under user control.

In order to overcome these weaknesses, we propose to exploit trust information explicitly expressed by the users. Users are allowed to state how much they consider

Paolo Massa
Fondazione Bruno Kessler, Via Sommarive 18, Povo (TN), Italy, e-mail: massa@fbk.eu

Paolo Avesani
Fondazione Bruno Kessler, Via Sommarive 18, Povo (TN), Italy, e-mail: avesani@fbk.eu

trustworthy each other user. In the context of RSs, this judgement is related to how much they consider the ratings provided by a certain user as valuable and relevant. This additional information (trust statements) can be organized in a trust network and a trust metric can be used to predict the trustworthiness of other users as well (for example, friends of friends). The idea here is to not search for similar users as CF does but to search for trustable users by exploiting trust propagation over the trust network. The items appreciated by these users are then recommended to the active user. We call this technique a Trust-aware Recommender System.

The goal of this chapter is to present a complete evaluation of Trust-aware Recommender Systems, by comparing different algorithms, ranging from traditional CF ones to algorithms that utilise only trust information with different trust metrics, from algorithms that combine both trust and similarity to baseline algorithms. The empirical evaluation is carried out on a real world, large dataset. We have also evaluated the different algorithms against views over the dataset (for example only on users or items satisfying a certain condition) in order to hightlight the relative performances of the different algorithms.

The chapter is structured as follows. Section 2 presents the motivations for our work in greater detail while Section 3 describes our proposal, focusing on the concept of trust, introducing the architecture of Trust-aware Recommender Systems, and commenting on related works. Section 4 is devoted to the experiments in which we compared different algorithms and the experimental results are then summarized and discussed in Section 5. Section 6 concludes the chapter.

2 Motivations

Ours is an Information society. The quantity of new information created and made available every day (news, movies, scientific papers, songs, websites, ...) goes beyond our limited processing capabilities. This phenomenon has been named "information overload" and refers to the state of having too much information to make a decision or to remain informed about a topic. The term was coined in 1970 by Alvin Toffler in his book Future Shock [1].

Recommender Systems (RS) [25, 3] are tools designed to cope with information overload. Their task is to pick out of the huge amount of new items created every day only the few items that might be of interest for the specific user and that might be worthy of her attention. Unsurprisingly, systems that automate this facility have become popular on the internet. Online Recommender Systems (RS) [25, 3], in fact, have been used to suggest movies, books, songs, jokes, etc. They have been an important research line because they promise to fulfill the e-commerce dream [3]: a different and personalized store for every single (potential) customer.

The most successful and studied technique for RSs is Collaborative Filtering [2] (CF). CF exploits a simple intuition: items appreciated by people similar to someone will also be appreciated by that person. While Content-based RSs require a description of the content of the items, Collaborative Filtering has the advantage of

relying just on the opinions provided by the users expressing how much they like a certain item in the form of a rating. Based on these ratings, the CF system is able to find users with a similar rating pattern and then to recommend the items appreciated by these similar users. In this sense, it does not matter what the items are (movies, songs, scientific papers, jokes, ...) since the technique considers only ratings provided by the users and so CF can be applied in every domain and does not require editors to describe the content of the items.

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

Table 1 An example of a small users × items matrix of ratings.

The typical input of CF is represented as a matrix of ratings (see Table 1), in which the users are the rows, the items the column and the values in the cells represent user rating of an item. In Table 1 for example, ratings can range from 1 (minimum) to 5 (maximum).

The CF algorithm can be divided into two steps. The *first step* is the *similarity assessment* and consists of comparing the ratings provided by a pair of users (rows in the matrix) in order to compute their similarity. The most used and effective technique for the similarity assessment is to compute the Pearson correlation coefficient [2]. The first step produces a similarity weight $w_{a,i}$ for every active user a with respect to every other user i.

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

The *second step* is the *actual rating prediction* and consists of predicting the rating the active user would give to a certain item. The predicted rating is the weighted sum of the ratings given by other user to that item, where the weights are the similarity coefficient of the active user with the other users. In this way the rating expressed by a very similar user has a larger influence on the rating predicted for the active user. The formula for the second step is the following

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

where $p_{a,i}$ represents the predicted rating that active user a would possibly provide for item i, r_u is the average of the rating provided by user u, $w_{a,u}$ is the user similarity weight of a and u as computed in step one, and k is the number of users whose ratings of item i are considered in the weighted sum (called neighbours).

However the Collaborative Filtering technique suffers from some key weaknesses we have identified and discuss in the following.

User similarity is often non computable. According to Equation 2, a user can be considered as neighbour for the active user only if it is possible to compute the similarity weight of her and the active user ($w_{a,u}$). In fact, in order to be able to create good quality recommendations, RSs must be able to compare the current user with every other user to the end 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, the rating matrix is usually very sparse because users tend to rate few of the available items (that can sum into the millions). Because of this *data sparsity*, it is often the case that two users don't share the minimum number of items rated in common required by user similarity metrics for computing similarity, and the system is not able to compute the similarity weight. As a consequence, the system is forced to choose neighbours from the small portion of comparable users and will miss other non-comparable but possibly relevant users. Moreover, even when two users share some commonly rated items, this number is usually very small and hence the computed user similarity is a very unreliable quantity: for example, deciding that two users are similar on the basis of the 3 movies they happened to both rate is not a very reliable measure. This problem is less serious for users who have already produced hundreds of ratings, but they are usually a small portion of the user base. Actually in most realistic settings most users have only provided a few or no ratings at all. They are called *cold start users* and it can be argued that they are the most important for RS since the system should be able to provide good recommendations, despite the small rating information available about them, in order to give them a reason to keep using the system and hence providing more ratings, which would allow better recommendation computation. However they are the most challenging due to the small quantity of information available about them. Often RSs fail on cold start users and are not able to produce recommendations for them with the consequence of driving them away. We believe that this is a serious weakness for RSs and that this aspect has been mainly neglected until now by the research efforts because the most used dataset for evaluating Recommender Systems didn't present these features. We will see in Section 4 how in real world datasets, both data sparsity and cold start users are the overwhelming reality.

Easy attacks by malicious insiders. Another weakness is related to attacks on Recommender Systems [17]. Recommender Systems are often used in e-commerce sites (for example, on *Amazon.com*). In these 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. And in fact, gaming standard CF techniques is very easy. While this important aspect has been neglected until recently, recently some recent studies have started to look at attacks of Recommender Systems [17, 20]. The simplest attack is the copy-profile attack: the attacker can copy the ratings of target users and fool the system into thinking that the attacker is in fact the most similar user to the target user. In this way every additional item the attacker rates highly will probably be recommended to the target user. Currently RSs are mainly centralized servers, and it should be noted that in general it is easy for a person to create countless fake identities, a problem that is also known

as “cheap pseudonyms” [4]. E-commerce sites don’t have incentives to disclose this phenomena and hence the impact of it on real systems is not known. Note however that there is at least one disclosed precedent. An occurrence of this behavior has been revealed publicly because of a computer “glitch” that occurred in February 2004 on the Canadian Amazon site. For several days this mistake revealed

the real names of thousands of people who had posted customer reviews of books under pseudonyms [21]. By analyzing the real names, ti became evident that the author of a book had in fact created many different pseudonyms on the Amazon site and used all of them to write reviews about her book and rate it highly. The possibility seriously undermines the functioning of Recommender Systems sites, which rely on ratings provided by anonymous users.

Moreover, if the publishing of ratings and opinions becomes more decentralized, for example with Semantic Web formats such as hReview [5] (for items reviews) or FOAF [10] (for expressing trust statements about people), these types of attacks will increasingly become an issue. In fact, while registering on a centralized Recommender System site and providing ad-hoc attack ratings must generally be done by hand and is a time consuming activity, on the Semantic Web or in other decentralized architectures such as P2P networks this activity could easily be carried out with automatic programs (bots). Basically, creating such attacks will become as widespread as email spam is today, or at least as easy. We hence believe that coping with attacks is an important topic for the research community interested in Recommender Systems.

Current Recommender Systems are hard for users to understand and control. Another weakness is that RSs are mainly conceived and perceived as black boxes [6]: the user receives the recommendations but doesn’t know how they were generated and has no control in the recommendation process. For example, in [7], the authors conducted a survey with real users and found that users want to see how recommendations are generated, how their neighbours are computed and how their neighbours rate items. Swearingen and Sinha [6] analyzed RSs from a Human Computer Interaction perspective and found that RSs are effective if, among other things, “the system logic is at least somewhat transparent”. Moreover, it seems that, as long as RSs give good results, users are satisfied and use them, but, when they start recommending badly or strangely, it is very difficult for the user to understand the reason and to fix the problem; usually the user quits using the RS [8, 13]. Even if the RS exposes what it thinks of you (explicit or implicit past ratings on items) and allows the user to modify them, this is a complicated task, involving for example a re-examination of dozens of past ratings in order to correct the ones that are not correct [8]. It has been claimed that “few of the Amazon users revise their profiles” when the recommendations start becoming obviously wrong [13]. RSs use the step of finding similar users only in propedeutic ways for the task of recommending items, but they don’t make the results of these computations visible to users, such as possibly similar unknown users: CF automates the process of recommending items but doesn’t help in discovering like minded people for community forming.

RS are mainly deployed as centralized servers. Presently, the most used RSs are run as centralized servers where all the community ratings are stored and where the

recommendations are computed for all users. This fact is a weakness for Recommender Systems for more than one reason.

One consequence is that users profiles are scattered in many different, not cooperating servers (for example, different subsets of a single user preferences about books of can be stored with Amazon, Barnes and Nobles, and many other online bookstores); every single server suffers even more sparseness and has problems in giving good quality recommendations. Moreover, this means users cannot move from one RS to another without losing their profiles (and consequently the possibility of receiving good recommendations and saving time). In essence, this situation is against competition and can easily lead to a global monopoly because it is almost impossible for new RSs to enter the market while, for consolidated RS owning much user information, it is even possible to enter new correlated markets. Clearly, companies prefer to not allow interoperability or publicity of this information because it is their company value. Anyway, as long as this useful information will remain confined in silos, it will not unveil all its potentially disruptive power. This lack of portability is a serious weakness of current Recommender Systems [22].

Moreover, the entity running the centralized RS is usually a commercial product vendor and its interests could be somehow different from giving the best recommendations to the user [9]. In general, users are not free to test for biases of the RSs or to know the algorithm used for generating the recommendations or to adapt it to their preferences or to run a completely different one.

While it is theoretically possible to run current RSs in a decentralized way, for example on the small device under user control such as a mobile, in practice Collaborative Filtering requires a large memory to store all the ratings and, mainly, a great computation power to perform all the operations on the possibly very huge ratings matrix.

We want also to point out how these centralized architectures are one of the reasons behind the lack of datasets and real testbeds on which to apply and test new research hypotheses related to RSs. It would be a totally different scenario if researchers could have access to all the rating information collected by centralized Recommender Systems such as Amazon and other online bookstore for instance. In fact, there were only few freely available datasets of ratings on items and they were used for offline testing but, in order to run online experiments, researchers had to invest a lot of time into creating their own RS and gathering enough users. However, this is not an easy task: Grouplens working group at the University of Minnesota[1] is a notable exception in this since it was able to get enough users for its online Recommender System, Movielens, and they were very kind in sharing it as a dataset usable by researchers.

In this section we have highlighted the weaknesses we believe beset current Recommender Systems. In the next section, we describe our proposal and how it alleviates these weaknesses.

[1] Grouplens homepage is at www.cs.umn.edu/Research/GroupLens

3 Our proposal: Trust-aware Recommender Systems

In this section we summarize our proposal: Trust-aware Recommender Systems. We start by introducing basic concepts about trust networks and trust metrics. We then present the logical architecture of Trust-aware Recommender Systems. We conclude this section by comparing our proposal with related work in the literature.

3.1 Trust networks and trust metrics

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 post 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) [12]. In these environments, it is often a good strategy to delegate the quality assessment task to the users themselves. The system can ask users to rate other users: in this way, users can express their level of trust in another users they have interacted with, i.e. issue a trust statement such as "I, Alice, trust Bob as 0.8 in [0,1]". The system can then aggregate all the trust statements in a single trust network representing the relationships between users. An example of a simple trust network can be seen in Figure 1. As a consequence of the previously introduced properties of trust, such a network is a directed, weighted graph whose nodes are peers and whose edges are trust statements.

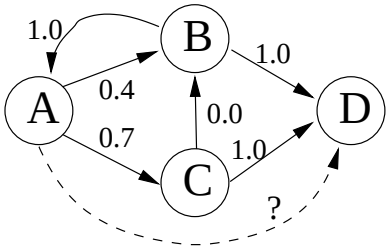


Fig. 1 Trust network. Nodes are users and edges are trust statements. The dotted edge is one of the undefined and predictable trust statements.

Since in most settings, a user has a direct opinion (i.e. has issued a trust statement) only about a small portion of the other peers, some computational tools can be designed for predicting the trustworthiness of unknown peers. These tools are Trust Metrics and Reputation Systems. The main idea behind these techniques is trust propagation over the trust network: if peer *A* trusts peer *B* at a certain level and

peer *B* trusts peer *D* at a certain level, something can be predicted about how much *A* should trust *D*.

These tools are starting to become more and more needed and useful because, thanks to the internet, it is more and more common to interact with unknown peers. Moreover, thanks to the internet, trust statements expressed by all peers can be published and made available to anyone so that aggregating them and reasoning on them is now becoming possible. This was not possible nor useful until a few years ago, and in fact, computational ways to exploit trust have begun to be proposed only very recently. In some sense, in order to get an idea of a certain peer's trustworthiness, we are relying on the judgments of other peers who have already interacted with them and shared their impression. There are many different proposed Trust Metrics [18, 10, 26, 15, 24].

An important distinction in Trust Metrics is in local and global [15]. Global Trust Metrics compute a single trust score for each peer of the trust network. This trust score is independent of the peer that is asking "How much should I trust this unknown peer?". Instead, Local Trust Metrics provide personalized scores. So a local Trust Metric might suggest to peer *Alice* to trust peer *Carol* and to peer *Bob* to distrust *Carol*. Global Trust Metrics compute a score for each peer that represents the average opinion of the whole community about that peer. Even if there is no agreement yet on definitions, in general, this global value is called "reputation" and "reputation systems" are what we called "global Trust Metrics". But the definitions are not that clear and very often the term "reputation" and "trust" are used synonymously just as "Reputation System" and "Trust Metric". PageRank [24], for example, is a global trust metric.

In the next section we will see how trust metrics can play a role in the context of Recommender Systems, essentially we propose them for replacing or integrating the users' similarity assessment of step 1.

3.2 An Architecture of Trust-aware Recommender Systems

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). There are two input informations: the trust matrix (representing all the community trust statements) and the ratings matrix (representing all the ratings given by users to items). The output is a matrix of predicted ratings that users would assign to items. The difference with respect to traditional CF systems is the additional input matrix of trust statements. The two logical steps of CF remain the same. The first step finds neighbours and the second step predicts ratings based on a weighted sum of the ratings given by neighbours to items. The key difference is in how neighbours are identified and how their weights are computed. The weight $w_{a,i}$ in Equation 2 can be derived from the user similarity assessment (as in traditional CF) or with the use of a trust metric. In fact in our proposed architecture for the first step there are two possible modules

able to produce these weights: a Trust Metric module or a Similarity Metric module. They respectively produce the Estimated Trust matrix and the User Similarity matrix: in both, row i contains the neighbours of user i and the cell of column j represents a weight in $[0,1]$ about how much user j is relevant for user i (trustable or similar). This is the weight $w_{a,i}$ in Equation 2 and represents how much ratings by user i should be taken into account when predicting ratings for user a (second step). A more detailed explanation of the architecture can be found in [14]. In Section 4 we are going to present experiments we have run with different instantiations of the different modules. For the Trust Metric module we have tested a local and a global trust metric. As local trust metric we have chosen MoleTrust [15], a depth-first graph walking algorithm with a tunable trust propagation horizon that allows us to control the distance to which trust is propagated. As global trust metric we have chosen PageRank [24], probably the most used global trust metric. For the Similarity Metric module we have chosen the Pearson Correlation Coefficient since it is the one that is reported to be performed best in [2]. Regarding the Rating Predictor module (second step), we experimented with selecting only weights from the Estimated Trust matrix or the User Similarity matrix and with combining them. For the purpose of comparison, we have also run simple and baseline algorithms that we will describe in next section.

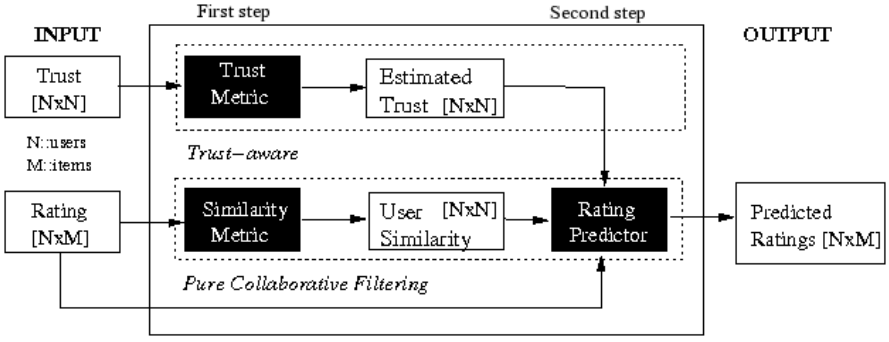


Fig. 2 Trust-Aware Recommender System Architecture.

3.3 How trust alleviates RS weaknesses

In the previous Section, we have presented our proposal for enhancing Recommender Systems by means of trust information.

In this section we discuss how Trust-aware Recommender Systems are able to alleviate the weaknesses besetting RSs that we have previously introduced. Section 4

will be devoted to empirical evidence confirming the claims, based on experiments run on a real world, large dataset.

The key new point of Trust-aware Decentralized Recommender Systems is to use trust propagation over the network constituted by trust statements in order to predict a trust score for unknown peers. This trust score can be used in the traditional Recommender System model in place of or in addition to the value of user similarity as a weight for the ratings of the specific peer.

The first weakness highlighted in Section 2 was that *often user similarity between two users is not computable* due to data sparsity. Instead, considering the trust statements expressed by all the peers, it is possible to predict a trust score for a larger portion of the user base. For example, by using a very simple global trust metric like the one used by eBay (see Section 4), it can be enough that a peer received at least one trust statement to be able to predict its trustworthiness. Even local trust metrics, which propagate trust starting from active users, are able to reach a large portion of the users in just a few propagation steps, considering that most social networks are highly connected and exhibit small world characteristics. Another important point related to the computability of user similarity that we mentioned when speaking about the weaknesses of current Recommender Systems is related to *cold start users*. Cold start users are the most critical users for standard CF techniques that are not able to generate any recommendation for them. Viceversa, they can benefit from the trust statements issued by other users. In particular, as soon as a cold start user provides at least one trust statement, it is possible to use a local trust metric. The local trust metric is able to propagate trust and predict trustworthiness for all reachable users, so that their ratings can be used in the rating prediction process. Issuing just one trust statement can be an effective mechanism for rapidly integrating new users, especially if compared with standard CF where users are usually required to rate at least 10 items before receiving a recommendation. A single trust statement can make the difference between an environment populated by users whose trustworthiness is totally uncertain to an environment in which it is possible to use a local trust metric and predict how much ratings provided by many other users can be taken into account.

With regard to *attacks on Recommender Systems*, considering trust information can be effective as well. For example, against shilling attacks [20] in which a user pretends to be similar to the user target of the attack. Trust-aware Recommender Systems can be used to consider only ratings provided by users predicted as trustworthy by the trust metric. Local Trust Metrics promise to be attack-resistant [18, 26], as long as there is no trust path from the active user to the users under control of the attacker. Essentially, while creating a fake identity is very easy and can be done by anyone [4], receiving a positive trust statement by a peer trusted by the user target of the attack is not as easy since it depends on judgments not under the control of the attacker. In our vision, exploiting of trust information allows being influenced only (or mainly) by “trustable” peers, either direct peers or indirect ones (friends of friends). This can reduce the user base used to find neighbours but surely keeps out malicious and fake peers. The sharing of opinions about peers is also a good way for detecting or spotting these attacks by virtue of a decentralized

assessment process. In this way, malicious ratings provided by users under attacker control are not taken into consideration when generating recommendations for the active user, just as if they didn't exist.

The possibility of only or mainly considering ratings provided by users whose predicted trust score is above a certain threshold would help in alleviating the weakness related to the fact that traditional RSs are *computational expensiveness*. In fact, reducing the number of considered users a priori could allow the algorithm to scale even in a domain with millions of peers and to be run on less powerful devices. For example, the trust metric might be designed to analyze just peers at distance 1 (on which the active peer issued direct trust statements) and peers at distance 2 significantly reducing the number of considered peers.

We reported in Section 2 how traditional *RS are often seen by users as black boxes* [7, 6] and thought hard to understand and control [8, 13]. RSs are considered more effective by users if, among other things, "the system logic is at least somewhat transparent" [6]. We believe that the concept of direct trust is easier to understand for users than the concept of user similarity. User similarity is computed out of a formula, usually the Pearson Correlation coefficient, which is not too easy to understand for a normal user [7]. A possible interface could communicate the reasons behind a certain recommendation explicitly referring to ratings provided by trusted users, with a text message like the following "this movie was recommended to you because 3 of the users you trust (Alice, Bob, Charlie) rated it as 5 out of 5. You can see their user profiles and, in case the recommendation is not satisfying to you, you can possibly revise your connections with them" with links to these users' profiles [10]. In fact, Sinha and Swearingen have found that people prefer receiving recommendations from people they know and trust, i.e., friends and family-members, rather than from online Recommender Systems [19]. By showing explicitly the users trusted by the active user, RSs may let the user feel that the recommendations are in reality coming from friends and not from some obscure algorithms. However we didn't conduct Human Computer Interaction analysis and survey with real users about different recommendation explanation interfaces.

The last weakness we introduced in Section 2 is related to *centralized architectures* that are at the moment the most adopted for current RSs. We think Trust-aware Recommender Systems demand a decentralized environment where all users publish 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 its users. In this way the rating prediction could be run locally on a device owned by the human user for whom ratings are predicted [22]. In this setting, the single peer can decide to retrieve and aggregate information from just a small portion of the peers, for example only ratings expressed by trusted peers. In this way, it is not necessary to build the complete ratings matrix or the complete trust network. This would reduce the computational power required for running the predictions and the bandwidth needed for aggregating the ratings. Trust-aware Decentralized Recommender Systems would not require very powerful computers, as is often the case for centralized service providers, but would work on many simple devices under the direct control of the human user.

In this section, we have argued how Trust-awareness alleviates some of the weaknesses of standard Recommender Systems. The next section discusses work related to our proposal.

3.4 Related work

There have been some proposals to use trust information in the context of Recommender Systems. We will give an account of the most significant ones here.

In a paper entitled "Trust in recommender systems" [16], O'Donovan and Smyth propose algorithms for computing Profile Level Trust and Item Level Trust. Profile Level Trust is the percentage of correct recommendations that this producer has contributed. Item Level Trust is a profile level trust that depends on a specific item. As the reviewers note, this quantity represents more of a "competence" measure and in fact reflects a sort of global similarity value. While in their work trust values are derived from ratings (of the Movielens dataset), in our proposal trust statements are explicitly expressed by users.

The PhD thesis of Ziegler [26] concentrates on RSs from different points of research. Regarding the integration of trust, he proposes a solution very similar to ours, i.e neighbour formation by means of trust network analysis. He has designed a local trust metric, Appleseed [26], that computes the top-M nearest trust neighbours for every user. He has evaluated algorithms against a dataset derived from AllConsuming (http://allconsuming.net), a community of 3400 book readers, with 9300 ratings and 4300 trust statements. Only positive trust statements are available. Ziegler found that hybrid approaches (using taxonomies of books and hence based on content-based features of books) outperforms the trust-based one which outperforms the purely content-based one. Performances on users who provided few ratings were not studied in detail.

Golbeck's PhD thesis [10] focuses on trust in web-based social networks, how it can be computed, and how it can be used in applications. She deployed an online Recommender System, FilmTrust (http://trust.mindswap.org/filmTrust/) in which users can rate films and write reviews and they can also express trust statements in other users based on how much they trust their friends about movies ratings. Trust statements in FilmTrust are weighted: users could express their trust in other users on a ten level basis. Golbeck designed a trust metric called TidalTrust [10] working in a breadth-first fashion similarly to MoleTrust [15]. We used MoleTrust in our experiments because it has a tunable trust propagation horizon parameter that lets us study how this parameter affects performances of the Recommender System. It is interesting to note that Golbeck's findings are similar to ours and that will be reported in the next section.

4 Empirical validation

In this Section we present experiments we have conducted for evaluating the performances of Trust-aware Recommender Systems. In particular we compare different instantiations of the modules of our proposed architecture (see Figure 2), so that the evaluated systems range from simple algorithms used as baselines to purely Collaborative Filtering ones, from systems using only trust metrics, both global and local, to systems that combine estimated trust and user similarity information. First we describe the dataset used and introduce our evaluation strategy, then we present the actual results of the experiments.

4.1 Dataset used in experiments: Epinions

The dataset we used in our experiments is derived from the *Epinions.com* web site. Epinions is a consumers opinion site where users can review items (such as cars, books, movies, software, ...) and also assign numeric ratings to them 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 and their Block list, i.e. a list of authors whose reviews they find consistently offensive, inaccurate, or not valuable [2]. Inserting users in the Web of Trust is tantamount issuing a trust statement of value 1, while inserting her in the Block List equals to issuing a trust statement of value 0 in their regard. Intermediate values such as 0.7 are not expressible on Epinions.

In order to collect the dataset, we wrote a crawler that recorded ratings and trust statements issued by a user and then moved to users trusted by that users and recursively did the same. Note, however, that the block list is kept private in Epinions in order to let users express themselves more freely, therefore it is not available in our dataset.

The Epinions dataset represents the most meaningful and large example where ratings on items and trust statements on users have been collected in a real world environment. We released the crawled dataset so that other researchers can validate their hypotheses and proposals on it. The crawled dataset can be found at www.trustlet.org/wiki/epinions.

Our dataset consists of 49,290 users who rated a total of 139,738 different items at least once. The total number of reviews is 664,824. The total number of issued trust statements is 487,181. Rating matrix sparsity is defined as the percentage of empty cells in the matrix users $\times$ items and in the case of the collected dataset is 99.99135%. The mean number of created reviews is 13.49 with a standard deviation of 34.16. It is interesting to look at what we have called "cold start users". They are the large majority of users. For example, 26,037 users expressed less than 5 reviews

[2] This definition is from the Epinions.com Web of Trust FAQ (http://www.epinions.com/help/faq/?show=faq_wot)

and represent 52.82% of the population. The mean number of users in the Web of Trust (friends) is 9.88 with a standard deviation of 32.85. Another interesting point is 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. Note that almost half of the ratings are a 5, i.e. the maximum possible value.

The characteristics we briefly described are very different from those of the Movielens dataset[3], the most commonly used dataset for RSs evaluation. In particular, in Movielens dataset all the users are guaranteed to have voted at least 20 items while in Epinions more than half of them have voted less than 5 items (cold start users). This also means that sparsity is much higher in Epinions and so finding overlapping on provided ratings between users and hence possible neighbours (step 1 of CF) is even harder. While on Epinions most of the rating values are 5 and 4, in Movielens all the different values are more balanced. This affects how different algorithms perform as we will see in the following sections.

4.2 New evaluation measures

The most used technique for evaluating Recommender Systems is based on *leave-one-out* [11]. Leave-one-out is an offline technique that can be run on a previously acquired dataset and involves hiding one rating and then trying to predict it with a certain algorithm. The predicted rating is then compared with the real rating and the difference in absolute value is the prediction error. The procedure is repeated for all the ratings and an average of all the errors is computed, the Mean Absolute Error (MAE) [11].

A first problem with MAE is that it weighs every error in the prediction of a rating in the same way. For example, let us suppose that our dataset contains only 101 users: one user provided 300 ratings while all the remaining 100 users provided just 3 ratings each. We call the first user a “heavy rater” and the other users “cold start users”. In this way our dataset contains 600 ratings. The leave-one-out methodology consists in hiding these 600 ratings one by one and then trying to predict them. Typically CF works well for users who have already provided numerous ratings and poorly on users who provided few ratings. A probable situation is that the error over the predictions of the heavy rater is small while the error over the predictions of the cold start users is high. However, in computing the Mean Absolute Error, the heavy raters weigh just as much as all the other users since they provided a very large number of ratings. This does not reflect the real situation in which there is actually one user who is probably satisfied with the prediction error (the heavy rater) and 300 users who are not satisfied (the cold start users). For this reason, the first additional measure we introduce is Mean Absolute User Error (MAUE). The idea is straightforward: we first compute the Mean Absolute Error for every single user independently and then we average all the Mean Absolute Errors related to every

[3] Distributed by Grouplens group at the University of Minnesota and available at http://www.cs.umn.edu/Research/GroupLens/

single user. In this way, every user has the same weight in the Mean Absolute User Error computation. This is very important since the Epinions dataset contains a large share of cold start users. In our experiments (see next section), this distinction was able to highlight different behaviours for different techniques that would otherwise have remained hidden inside the MAE value.

Another important measure that is often not reported and studied in evaluation of RSs is coverage. Herlocker et al. in their solid review of Recommender Systems evaluation techniques [11] underline how it is important to go "beyond accuracy" in evaluating RSs and count coverage as one step in this direction but also note how few works have investigated it. Coverage simply refers to the fraction of ratings for which, after being hidden, the RS algorythm is able to produce a predicted rating. It might in fact be the case that some RS techniques are not able to predict the rating a user would give to an item. Again we believe that coverage was understudied by many research efforts because in Movielens, the most used dataset for evaluation of RSs, the coverage over ratings tends to be close to 100%. This is due to the fact that all the users are guaranteed to have voted at least 20 items and that there are some items that are rated by almost every user. Instead on a very sparse dataset that contains a large portion of cold start users and of items rated just by one user, coverage becomes an important issue since many of the ratings become hardly predictable. While the percentage of predictable ratings (*ratings coverage*) is an important measure, it has the same problem we highlighted earlier for Mean Absolute Error, it weighs heavy raters more. Following the same argument as before, we introduce also the *users coverage*, defined as the portion of users for which the RS is able to predict at least one rating. In fact, it is often the case that a RS is successful in predicting all the ratings for a user who provides many ratings and performs poorly for a user who has rated few items. Going back to the example introduced earlier, it might be that for the heavy rater who rated 300 items, the RS is able to predict all of them, while it fails on all the ratings provided by the 100 cold start users. In this case, the ratings coverage would be $\frac{300}{600} = 0.5$. Viceversa the users coverage would be $\frac{1}{100} = 0.01$.

A possibility given by a very large dataset of ratings is to study performances of different RS techniques on different portions of the input data (called "views") that, given the large numbers, remain significant. It is possible for example to compute MAE only on users who satisfy a certain condition. For example, as we already mentioned, while it might be very easy to provide good quality recommendations to a user who already provided 100 ratings to the system (heavy rater) and hence has given a very detailed snapshot of her opinions, it might be much more difficult to provide good quality recommendations to a user who has just joined the system and, for example, has entered only 2 ratings. With this regard, it is possible to compute evaluation measures such as MAE or users coverage only on these portions in order to analyze how a certain technique works on a particular subset of the data.

Views can be defined over users, over items and over ratings depending on their characteristics. We have already implicitly introduced many times the view over users based on the number of ratings that they have provided: users who provided

few ratings are called “cold start users” and users who provided many ratings are called “heavy raters”. As acknowledged also by [23], evaluating the performances of RSs in “cold start” situations has not been extensively covered in the literature. Our evaluations will concentrate on the relative performances of different techniques on these different classes of users, such as cold start users, who provided from 1 to 4 ratings; heavy raters, who provided more than 10 ratings; opinionated users, who provided more than 4 ratings and whose standard deviation is greater than 1.5; black sheep, users who provided more than 4 ratings and for which the average distance of their rating on item *i* with respect to mean rating of item *i* is greater than 1

Revealing views can be defined also over items. In this chapter we report evaluations performed on niche items, which received less than 5 ratings, and controversial items, which received ratings whose standard deviation is greater than 1.5. Making an error on a controversial item can be very serious and can mine the confidence the user places in the RS, for example, a user would be very unsatisfied to receive a recommendation for a movie about which she holds a clear and very negative opinion.

Additional views can be designed also on ratings. For example various measures can be computed only on ratings whose value is 1, in order to analyze how a certain technique performs on these ratings, or only on ratings whose value is greater or equal to 4.

We introduce these views because they better capture the relative merits of the different algorithms in different situations and better represent their weaknesses and strengths.

4.3 Results of the experiments

Every different instantiation of the Trust-aware Recommender System architecture is evaluated with regard to the measures we have defined (MAE, MAUE, ratings coverage, users coverage), also focusing the analysis on the different views previously introduced, such as, for example, cold start users and controversial items. In the following we discuss the results of the experiments condensed in Tables 2 and 3. Figures 4 and 5 graphically present just one of the measures reported in the tables, precisely the row labeled “Cold users” (i.e. MAE and ratings coverage on predictions for cold start users and MAUE and users coverage) in order to give the reader a visual grasp of the relative benefits of the different techniques.

4.3.1 Trivial algorithms seem very effective

As a first step in our analysis we tested a very simple algorithm that always returns 5 as the predicted rating a user would give to an item. We call this algorithm *Always5*.

$$prediction_{Always5}(a,i) = 5$$

This trivial algorithm is not meaningful from a RS point of view since, for instance, it does not allow to differentiate and prioritize the different items. However, it allowed us to start exploring which MAE a simple algorithm would achieve. The MAE over all the ratings is 1.008. This result is not too bad, especially if we compare it with more complex algorithms as we will do in the following.

Another trivial algorithm is the one we call *UserMean*. The idea of UserMean is simply to return the mean of the ratings provided by one user. Remember that we use leave-one-out as evaluation technique so we remove every single rating before computing the prediction.

$$prediction_{UserMean}(a,i) = \frac{\sum_{j=1}^{m}(r_{a,j})}{m}$$

where m is the number of items rated by user a.

The reason for such good performances is that in our dataset most of the rating values are in fact 5 and this is a notable difference with respect to other datasets, for instance MovieLens, on which these trivial algorithms work very badly. But in our case we have two very simple and not personalized algorithms that seem to perform well enough. This fact suggested to us that just presenting the Mean Absolute Error over all the ratings is not a useful way to compare different algorithms. We introduced the evaluation views explained in Section 4.2 in order to have an evaluation technique better able to capture the relative merits of the different algorithms in different situations and to better represent their weaknesses and strengths. In fact, on the controversial items view for instance, these trivial algorithms perform very badly.

4.3.2 Simple average better than Collaborative Filtering

Another trivial algorithm is the one that predicts - as a rating for a certain item - the unweighted average of all the ratings given to that item by all the users but the active user. It is a non-personalized technique that is like assigning 1 as similarity or trust weight to all the users in the second step of CF (Equation 2 with $w_{a,i}$ always equal to 1). For this reason we call it *TrustAll*.

$$prediction_{TrustAll}(a,i) = \bar{r}_a + \frac{\sum_{u=1}^{k}(r_{u,i} - \bar{r}_u)}{k} \tag{3}$$

To our surprise, TrustAll outperformed standard Collaborative Filtering algorithms, achieving a MAE of 0.821 (against 0.843 of standard *CF*). On the other hand, on MovieLens dataset, we observe the expected result: MAE of CF is 0.730 while MAE of TrustAll is 0.815. Moreover, the number of predictable Epinions ratings (the coverage) is 51.28% for CF and 88.20% for TrustAll, while on Movielens ratings they are both close to 100%. The reason for these important differences is in the datasets. The Epinions dataset contains mostly 5 as rating value and most of the users provided few ratings (cold start users). We believe these facts, not observed in other RS datasets, allowed us to study certain characteristics of RS algorithms that

Mean Absolute Error / Ratings Coverage					
Views	Algorithms				
	CF	MT1	MT2	MT3	TrustAll
All	0.843 51.28%	0.832 28.33%	0.846 60.47%	0.829 74.37%	0.821 88.20%
Cold users	1.094 3.22%	0.674 11.05%	0.833 25.02%	0.854 41.74%	0.856 92.92%
Heavy raters	0.850 57.45%	0.873 30.85%	0.869 64.82%	0.846 77.81%	0.845 92.92%
Contr. items	1.515 45.42%	1.425 25.09%	1.618 60.64%	1.687 81.01%	1.741 100.0%
Niche items	0.822 12.18%	0.734 8.32%	0.806 24.32%	0.828 20.43%	0.829 55.39%
Opin. users	1.200 50%	1.020 23.32%	1.102 57.31%	1.096 74.24%	1.105 92.80%
Black sheep	1.235 55.74%	1.152 23.66%	1.238 59.21%	1.242 76.32%	1.255 97.03%

Table 2 Accuracy and coverage measures on ratings, for different RS algorithms on different views.

were previously unexplored. The problem with CF in our dataset is that the Pearson correlation coefficient (similarity weight output of the first step of CF) is often not computable because of data sparsity and hence only the ratings of a small percentage of the other users can be utilized when generating a recommendation for the active user. Since there is not too much variance in rating values (most of them are 5), an unweighted average is usually close to the real value. On cold start users, the balance is even more for TrustAll. The coverage of CF on cold start users is only 3.22% while the coverage of TrustAll is 92.92% and the MAE of CF is 1.094 while the MAE of TrustAll in 0.856. Note that in the real-world Epinions dataset, cold start users make up more than 50% of total users. In fact, for a cold start user the first step of CF almost always fails since it is very unlikely to find other users which have rated the same few items and hence the similarity weight is not computable. However, these results are not totally dismissive of CF, in fact, on controversial items CF outperforms TrustAll (MAE of 1.515 against 1.741). In this case, CF is able to just consider the opinions of like minded users and hence to overcome the performances of TrustAll, a technique that - not being personalized - performs more poorly. This means that when it is really important to find like-minded neighbours CF is needed and effective. Also note that the error over ratings received by controversial items is greater than the error over all the ratings, meaning that it is harder to predict the correct ratings for these items.

4.3.3 Trusted users are good predictors

In this subsection we start comparing performances of RS algorithms that use only trust information (top box in Figure 2) with standard CF (bottom box). We start by

using only the users explictly trusted by the active user, i.e. not propatagating trust or setting the propagation horizon at 1 for the local Trust Metric MoleTrust. We call this algorithm *MT1*.

The Formula is very similar to Formula 2, the only difference being that users weights are derived from the direct trust statements.

$$prediction_{MT1}(a,i) = \bar{r}_a + \frac{\sum_{u=1}^{k} trust_{a,u}(r_{u,i} - \bar{r}_u)}{\sum_{u=1}^{k} trust_{a,u}} \tag{4}$$

where k is the number of users in which user u expressed a trust statement and $trust_{a,u}$ is the value of the trust statement explicitly issued by user a about user u. In the case of the analyzed dataset, k is the number of users in the Web of Trust (friends) and $trust_{a,u}$ has value 1.

In general, RSs based on trust propagation work better with cold start users. They don't use the (little) rating information for deriving a similarity measure to be used as weight for that user, but use the trust information explicitly provided by the user. In this way, even for a user with just one friend, it is possible that that friend has rated the same items and hence evaluating the accuracy of a prediction becomes possible. It is also possibly the case that that friend has very similar tastes to the current user and hence the error is small. In fact, the MAE of MT1 over cold start users is 0.674 while the MAE of CF is, as already discussed, 1.094. The difference in error is very high and particulary relevant since it is important for RSs to provide personalized recommendations as soon as possible to users who have not yet provided many ratings so that these users appreciate the system and keep using it, providing more ratings. Moreover, cold start users are a very large portion of the users in our dataset.

Mean Absolute User Error / Users Coverage					
Views	Algorithms				
	CF	MT1	MT2	MT3	TrustAll
All	0.938 40.78%	0.790 46.64%	0.856 59.75%	0.844 66.31%	0.843 98.57%
Cold users	1.173 2.89%	0.674 17.49%	0.820 30.61%	0.854 42.49%	0.872 96.63%
Heavy raters	0.903 86.08%	0.834 79.78%	0.861 88.42%	0.834 89.42%	0.820 100.00%
Contr. items	1.503 15.76%	1.326 11.74%	1.571 21.66%	1.650 27.85%	1.727 37.16%
Niche items	0.854 10.77%	0.671 10.27%	0.808 20.73%	0.843 32.83%	0.848 52.04%
Opin. users	1.316 61.20%	0.938 60.74%	1.090 76.51%	1.092 79.85%	1.107 100.00%
Black sheep	1.407 67.78%	1.075 60.83%	1.258 75.34%	1.285 77.70%	1.300 100.00%

Table 3 Accuracy and coverage measures on users, for different RS algorithms on different views.

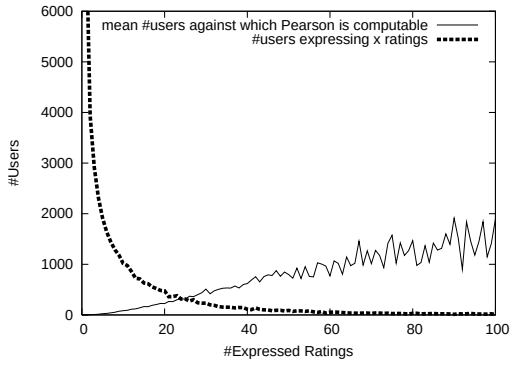
Let us now compare performances of CF and MT1 over all the ratings. The MAUE achieved by MT1 and CF is respectively 0.790 and 0.938. Regarding prediction coverage, while CF is able to predict more ratings than MT1 (ratings coverage is 51.28% vs. 28.33%), MT1 is able to generate at least a prediction for more users (users coverage is 46.64% vs. 40.78%). Summarizing, MT1 is able to predict fewer ratings than CF but the predictions are spread more equally over all users (which can then be at least partially satisfied) and, regarding errors, CF performs much worse than MT1 when we consider the error achieved over every single user in the same way and not depending on the ratings provided. These facts have the following reason: CF works well - both in terms of coverage and in terms of error - for heavy raters (users who already provided a lot of ratings) while it performs very poorly on cold start users. On many important views such as controversial items and opinionated users MT1 outperforms both CF and TrustAll.

4.3.4 Propagating trust with a Local Trust Metric

In the previous section we analyzed performances of RS algorithms that consider only trust information but don't propagate trust.

One of the weaknesses we highlighted in Section 2 was the fact that user similarity is often non computable and in this way the number of neighbours whose ratings can be considered in Formula 2 is small. We claimed this was especially the case for cold start users. We also claimed that, by using explicit trust statements and trust propagation, it was possible to predict a trust score for many more users and use this quantity in place of (or in combination with) the user similarity weight.

Here we analyze and compare the number of users for which it is possible to compute a user similarity value and a predicted trust one.



User base	No. of Users	Frac. of Total	Mean No. of Comp. Users
All	49,290	100.00%	160.73
5-	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. 3 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 3 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 those 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 3 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 each user against every other user; in this case the mean number of users would have been 49289 independently of the number of written reviews. Instead, Figure 3 makes evident how on the Epinions dataset the technique is far from ideal.

Let us now concentrate on "cold start users". For users who expressed less than 5 ratings (who are more than 50% of the users) Pearson Correlation Coefficient is computable on average only against 2.74 users over 49290 (as shown in the row labeled "5-" of Figure 3) 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 overlapping user, Pearson correlation coefficient is computable only against 9773 users that is just 19.83% of the entire population.

This plot is a stray evidence of how Pearson correlation coefficient is often not computable and hence ineffective.

Let us now analyze the computability of predicted trust and compare it with computatibilty of user similarity. We compute the number of users in which it is possible to predict trust starting from a certain user as the number of users at a certain distance from that user. In Table 4 we report the mean number of users reachable by propagating trust at different distances and the mean number of users for which user similarity weight is computable. The standard CF technique (Pearson correlation coefficient) on average allows computing 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 also considered 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, by propagating trust, it is possible to reach 94.54 users in just 2 steps and 9120.78 in 4 steps (see Table 4).

This table tells that on a dataset of real users (Epinions), trust propagation is potentially able to predict a trust score in many more users than the traditional RS technique of computing user similarity over the ratings matrix using the Pearson

Userbase	Propagating Trust (up to distance)				Using Pearson
	1	2	3	4	
All users	9.88	400	4386	16334	161
Cold start users	2.14	**94.54**	1675	9121	**2.74**

Table 4 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).

Correlation Coefficient. Note also that, because of the sparsity of the rating data, the Pearson Correlation Coefficient is usually computed only based on a small number of overlapping items, producing a noisy and unreliable value. This difference in the number of users in which it is possible to compute similarity and trust is even exacerbated for cold start users. These users are usually the largest portion of users and also the ones to benefit most from good quality recommendations.

Since by propagating trust it is possible to reach more users and hence to compute a predicted trust score in them and to count them as neighbours, the prediction coverage of the RS algorithm increases. In fact the larger the trust propagation horizon, the greater the coverage (see columns MT1, MT2 and MT3 of Table 2 and 3). For instance, on all ratings, the ratings coverage increases from 28.33% for MT1, to 60.47% for MT2, to 74.37% for MT3. By continuing to propagate trust (i.e. expanding the trust propagation horizon) it is possible to consider more and more users as possible neighbours and hence to arrive at 88.20%, the ratings coverage of TrustAll which considers every user who provided a rating. The downside of this is that the error increases as well. For example, on cold start users, the MAUE is 0.674 for MT1, 0.820 for MT2 and 0.854 for MT3. These results say that by propagating trust it is possible to increase the coverage (generate more recommendations) but that it also considers users who are worse predictors for the current user so that the prediction error increases as well. The trust propagation horizon basically represents a tradeoff between accuracy and coverage.

4.3.5 Global Trust Metrics not appropriate for Recommender Systems

An additional experiment we performed is about testing the performance of global Trust Metrics as algorithms for predicting the trust score of unknown users. A global trust metric predicts the same trust scores in other users for every user. This technique, like TrustAll, is hence not personalized. We have chosen to run PageRank [24] as global trust metric and to normalize the output value in [0,1]. We call the Recommender System that uses PageRank for its Trust Metric module, *PR*. PR performs similarly to TrustAll, even slightly worse (MAE of 0.847 and 0.821 respectively). This means that a global Trust Metric is not suited for Recommender Systems whose task is to leverage individual different opinions and not to merge all

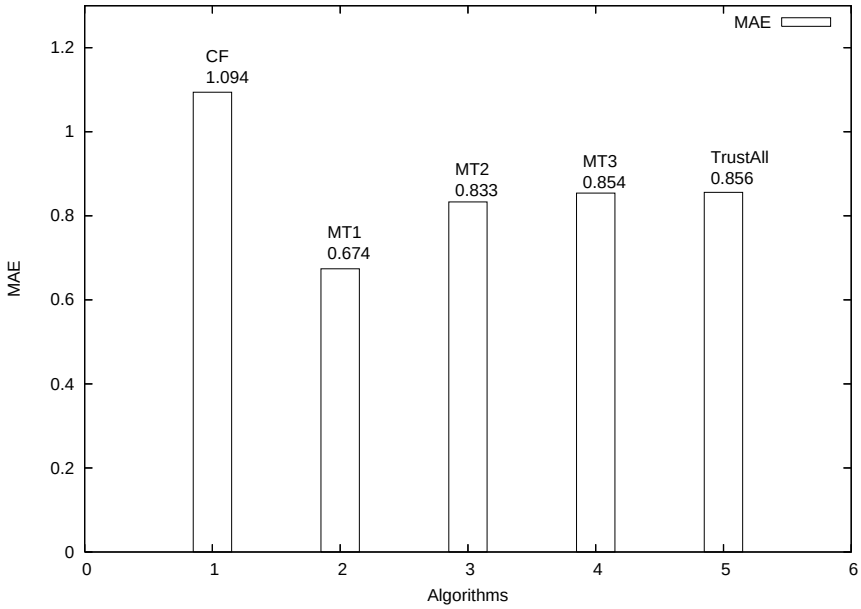


Fig. 4 MAE on cold start users for some representative algorithms.

of them into a global average. We also tried to restrict the neighbours to just the first 100 users as ranked by PageRank but this algorithm (called *PR100*) - while of course reducing the coverage - reports even larger errors (MAE of 0.973). The reason behind these bad performances is that globally trusted users (as found by PageRank) tend to be peculiar in their rating patterns and provide more varied ratings so that averaging them generates larger errors. In contexts such as understanding which is the most relevant web page about a certain topic or the most relevant scientific paper, global trust metrics such as PageRank can be highly effective. However global trust metrics are not suited for finding good neighbours, especially because the task of RSs is to provide personalized recommendations while global trust metrics are unpersonalized. We also showed in [15] that also in social contexts local trust metric performs better. This is especially true for controversial users, for which a common agreement cannot exist. We suggested how it might be important for the healthiness of a society to favor diversity of opinons and not to force everyone to suffer from the tyranny of the majority [15] and hence to adopt local trust metrics.

4.3.6 Combining Estimated Trust and User Similarity

In the architecture of Trust-aware Recommender Systems (Figure 2), the “rating predictor” module takes as input both the Estimated Trust matrix and the User Sim-

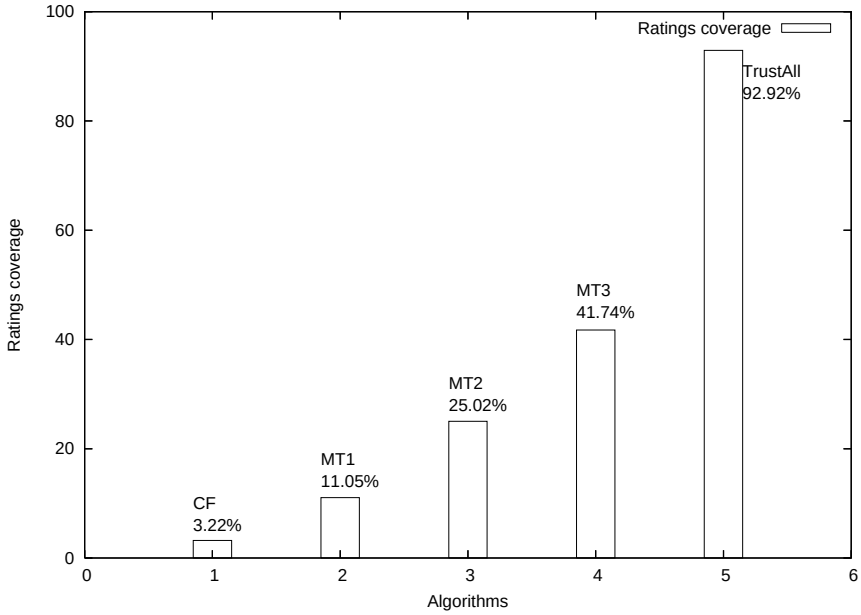


Fig. 5 Ratings coverage on cold start users for some representative algorithms.

ilarity matrix. The idea is that the weight of a neighbour used in Equation 2 can be derived both from the user similarity value computed by the Similarity Metric (Pearson Correlation Coefficient in our case) and the predicted trust value computed by a Trust Metric. We have already commented on the number of users for which it is possible to compute a similarity weight or a predicted trust in the previous subsection [14]. However, in order to devise a way of combining these two matrices, it is interesting to analyze how much they overlap. As previously reported, the number of users reachable in one step (the ones used by MT1) are on average 9.88 and the number of users in which a user similarity coefficient is computable are on average 160.73. The two matrix rows overlap only on 1.91 users on average, that is only for 1.91 users we have both a predicted trust and a user similarity. The number of users reachable propagating trust up to distance 2 is 399.89. Comparing it again with the number of users in which a similarity coefficient is computable (160.73), the average number of users present in both lists is 28.84. These numbers show how Pearson Correlation coefficient and MoleTrust address different portions of the user base in which they are able to compute a weight. So, in order to combine these weights, we tested the simple technique of computing a weighted average when there are two weights available and, in case only one is available, of using that. We call this technique *CF+MTx*: for example the system that combine CF and MT1 is called CF+MT1. The results are not very good. When comparing CF+MT1 with CF and MT1 for example, we see that the coverage is greater than the coverage of the two techniques. This is of course to be expected since CF+MT1 considers both the users

for which it is possible to predict a trust score (as MT1 does) and the users for which it is possible to compute a user similarity (as CF does). However, the error of CF+MTx generally lies in between of CF and MTx, that is worse than MTx and better than CF. The problem is that, as we reported earlier, CF is almost never able to find good neighbours and hence making an average of the users who are similar and of the users that are trusted produces worse results than just considering trusted users. Since techniques that used only trust were superior in previous tests to CF-based ones, we also try to just use the predicted trust score when both the weights were available but the results are very similar.

5 Discussion of results

In this section we summarize and discuss the most important results of the presented experiments. The first important result it that considering only the ratings of directly trusted users is the technique that, in general, achieves the smallest error with an acceptable coverage. The comparative improvement over the other techniques is particularly evident with regard to controversial items and black sheep, two of the most important and challenging views. With regard to cold start users, standard CF techniques totally fail and are not able to generate any recommendation. Instead, by considering ratings of trusted users we achieve a very small error and are able to produce a recommendation for almost 17% of the users. We can therefore state that providing a single trust statement is an easy, effective and reliable way of bootstrapping the Recommender System for a new user. It is important to underline that the evidence is based on experiments carried out on a real world, large dataset. In particular the Epinions datasets allowed us to explore topics which were not addressed before in research papers, such as cold start users and other views. Using our local Trust Metric MoleTrust in order to propagate trust allows users trusted by trusted users (at distance 2 from active user in the directed trust network), or even further away users, to be considered as possible neighbours. In this way, the coverage increases significantly, but the error increases as well. This means that ratings of users at distance 2 (or more) are less precise and less useful than ratings of users at distance 1, i.e. directly trusted by the active user. However it is an open issue to see if different local trust metrics are able to extract just some of the other users such that their ratings are really effective in improving the recommendation accuracy. In fact, this method can be used to evaluate the quality of different trust metrics, i.e. a better trust metric is the one that is able to find the best neighbours and hence to reduce the prediction error. As a last point we would like to highlight how Collaborative Filtering, the state of the art technique, performed badly in our experiments, especially on cold start users (which in fact are more than 50% in our dataset). The reason for this lies in the characteristics of the datasets used for evaluation. In previous research evaluations the most used dataset was MovieLens, while we used a dataset derived from the online community of Epinions.com. As we have already explained they present very different characteristics. It is still an open point to understand how

much the different datasets influence the evaluation of different algorithms' performances. In order to help this process, we released the dataset we crawled from Epinions. The dataset is downloadable at http://www.trustlet.org/wiki/epinions.

6 Conclusions

In this chapter we have presented our proposal for enhancing Recommender Systems by use of trust information: Trust-aware Recommender Systems. We have presented a deep empirical evaluation on a real world, large dataset of the performances of different algorithms ranging from standard CF to algorithms powered with local or global trust metrics, from the combination of these to baseline algorithms. We have also segmented the evaluation only on certain views (cold start users, controversial items, etc.) over the dataset in order to better highlight the relative merits of the different algorithms. The empirical results indicate that trust is very effective in alleviating weaknesses inherent to RSs. In particular, the algorithms powered with MoleTrust local trust metric are always more effective than CF algorithm, which surprisingly performs even worse than simple averages when evaluated on all the ratings. This difference is especially large when considering cold start users, for which CF is totally uneffective. The trust propagation horizon represents a trade-off between accuracy and coverage, i.e. by increasing the distance to which trust is propagated by the local trust metric the prediction coverage increases but the error increases as well. Results also indicate that global trust metrics are not appropriate in the context of RSs. Given that the user similarity assessment of standard CF is not effective in finding good neighbours, the algorithms that combine both user similarity weight and predicted trust weights are not able to perform better than algorithms that just utilize trust information.

References

1. A. Toffler. Future shock, 1970. Random House, New York.
2. J. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artifi cial Intelligence*, Madison, WI, July 1998. Morgan Kaufmann.
3. J.B. Schafer, J. Konstan and J. Riedl. *Recommender Systems in E-Commerce*. Proceeding of the ACM Conference on Electronic Commerce, Pittsburgh, PA, USA, 1999
4. E. J. Friedman and P. Resnick (2001). The Social Cost of Cheap Pseudonyms. *Journal of Economics and Management Strategy*, 10(2):173–199.
5. Microformats.org. hreview. Retrieved December 28, 2007, from http://microformats.org/wiki/hreview.
6. 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.
7. J.L. Herlocker, J.A. Konstan, and J. Riedl. Explaining Collaborative Filtering Recommendations. In *Proc. of CSCW 2000.*, 2000.

8. J. Zaslow. If TiVo Thinks You Are Gay, Here's How to Set It Straight. The Wall Street Journal, 26 November 2002
9. N. Wingfield and J. Pereira. Amazon uses faux suggestions to promote new clothing store, December 2002. Wall Street Journal.
10. J. Golbeck. *Computing and Applying Trust in Web-based Social Networks*. PhD thesis, University of Maryland, 2005.
11. J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, 2004.
12. P. Massa. A survey of trust use and modeling in current real systems, 2006. Chapter in "Trust in E-Services: Technologies, Practices and Challenges", Idea Group, Inc.
13. L. Guernsey. Making Intelligence a Bit Less Artificial. New York Times, 5 January 2003.
14. P. Massa and P. Avesani. Trust-aware collaborative filtering for recommender systems. In *Proc. of Federated Int. Conference On The Move to Meaningful Internet: CoopIS, DOA, ODBASE*, 2004.
15. P. Massa and P. Avesani. Trust metrics on controversial users: balancing between tyranny of the majority and echo chambers, 2007. International Journal on Semantic Web and Information Systems.
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, New York, NY, USA, 2005. ACM Press.
17. M. P. O'Mahony, N. J. Hurley, and G. C. M. Silvestre. Recommender systems: Attack types and strategies. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI-05)*, Pittsburgh, Pennsylvania, USA, 9–13, Jul 2005. AAAI Press.
18. R. Levien. *Advogato Trust Metric*. PhD thesis, UC Berkeley, USA, 2003.
19. R. Sinha and K. Swearingen. Comparing recommendations made by online systems and friends, 2001. In Proceedings of the DELOS-NSF Workshop on Personalization and Recommender Systems in Digital Libraries. Dublin, Ireland.
20. S. K. Lam and J. Riedl. Shilling recommender systems for fun and profit. In *Proceedings of WWW04*, 2004.
21. Ctv.ca (2004). Amazon glitch outs authors reviewing own books. Retrieved December 28, 2007, from http://www.ctv.ca/servlet/ArticleNews/story/CTVNews/1076990577460_35.
22. B. N. Miller, J. A. Konstan, and J. Riedl. Pocketlens: Toward a personal recommender system. *ACM Trans. Inf. Syst.*, 22(3):437–476, 2004.
23. 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.
24. L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford, USA, 1998.
25. P. Resnick and H. Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.
26. C.-N. Ziegler. *Towards Decentralized Recommender Systems*. PhD thesis, Albert-Ludwigs-Universität Freiburg, Freiburg i.Br., Germany, June 2005.