Propagation of trust and distrust for the detection of trolls in a social network

F. Javier Ortega *, José A. Troyano, Fermín L. Cruz, Carlos G. Vallejo, Fernando Enríquez

Department of Computer Languages and Systems, University of Seville, Avda. Reina Mercedes s/n, 41012 Seville, Spain

Keywords:
Social networks
Trust and Reputation Systems
Graph theory
Ranking algorithms

A B S T R A C T

Trust and Reputation Systems constitute an essential part of many social networks due to the great expansion of these on-line communities in the past few years. As a consequence of this growth, some users try to disturb the normal atmosphere of these communities, or even to take advantage of them in order to obtain some kind of benefits. Therefore, the concept of trust is a key point in the performance of on-line systems such as on-line marketplaces, review aggregators, social news sites, and forums. In this work we propose a method to compute a ranking of the users in a social network, regarding their trustworthiness. The aim of our method is to prevent malicious users from illicitly gaining high reputation in the network by demoting them in the ranking of users. We propose a novel system intended to propagate both positive and negative opinions of the users through a network, in such way that the opinions from each user about others influence their global trust score. Our proposal has been evaluated in different challenging situations. The experiments include the generation of random graphs, the use of a real-world dataset extracted from a social news site, and a combination of both a real dataset and generation techniques, in order to test our proposals in different environments. The results show that our method performs well in every situations, showing the propagation of trust and distrust to be a reliable mechanism in a Trust and Reputation System.

1. Introduction

The relevance of Trust and Reputation Systems (TRSs) has grown in the past few years, in parallel to the huge expansion of social networks. The problem of managing the trustworthiness of the users of an on-line community is not new, and it has been faced with diverse methods depending on the context and the aim of the network where it was applied. For example, in on-line forums and social news sites is not rare to encounter a troll, that is a user who comments contents with the aim of focusing the attention on himself diverting the topic of discussion, or simply to cause an argument. These networks usually have a special group of users (*moderators*) who have the ability (and responsibility) of banning those users that they consider malicious or trolls. The main benefit of this kind of fully supervised systems is the ability of determining whether a user is a troll or not regarding all their actions, including their comments in the network.

On the other hand, a system based on moderators has two main disadvantages: scalability and subjectivity. The supervised system can be a reliable mechanism in networks with a medium-small number of users, but it is not scalable to social networks with a high amount of members and contents. Some social networks that have adopted this kind of TRS try to overcome this problem by delegating some of the moderation task to the users. They are provided with a mechanism to send notifications to the moderators whenever they detect an abusive content or behavior in the network. This system can help the moderators, but it also offers options to malicious users for

* Corresponding author. Tel.: +34 954556234; fax: +34 954557139.
E-mail address: javierortega@us.es (F.J. Ortega).

cheating the TRS in other ways. Concerning the subjectivity problem, the decisions about whether a user is a troll or not, or whether a comment is a controversial contribution to a discussion or just trolling, depends on the judgment of the moderators.

Other systems provide all the users with the ability of giving their opinions about other users, or even the contents generated by them. These voting systems are decentralized, so they avoid the subjectivity problem by delegating the decision to all the users of the network, and also the scalability issue because there is not an authority (or a group of them) that has to make all the decisions. There exist different techniques that processes this information together with the topology of the network of relationships formed by the users in order to propagate their opinions through the system with the aim of computing a trust score for each user.

In this work, we propose a novel approach to this problem, defining a general method that propagates not only the positive opinions of the users in a social network, but also the negative opinions. This is important due to the existence of several social networks where negative opinions are as determining as the positive ones or even more, such as in on-line marketplaces where the potential buyers not only evaluate the price of a product but also the reputation of the seller. In this case, a negative feedback from a customer can strongly influence the decisions of other users about that seller. The negative opinions are also important in other social networks, such as review aggregators, social news sites and recommender systems.

Our approach is intended to build a ranking of users according to their trustworthiness, demoting the users who present a dishonest behavior in the system. Our approach also avoids the negative effects that the actions of these malicious users can cause in the reputation of other users in the network. Unlike other proposed methods, in our proposal both positive and negative information are propagated in order to evaluate the scores of each user, being able to capture the transitivity of trust and distrust in a social network.

The rest of the work is organized as follows. In Section 2 we briefly talk about other works on trust and reputation analysis. Section 3 introduces our approach. The design and the results of the experiments performed to test our approaches are shown in Sections 4 and 5, respectively. Finally, in Section 6 we point out the conclusions and future work on this field.

2. Related work

Several works about TRS's have been carried out, studying the challenges that these systems must face in order to provide the users of social networks with reliable information about the trustworthiness of the rest of the users in the system. Common problems in the implementation of TRS in a social network are discussed in [1,2]. They point out the existence of a bias in the majority of the ratings toward giving positive or negative scores, depending on the topic of the network. They also talk about the absence of incentives that users usually have for providing reviews in the system. These incentives can consist in an improved service, such as a higher download rate in the case of streaming services or even economic rewards in the case of on-line marketplaces.

Other works study the nature of the relationships in a specific social network, and then propose methods intended to deal with them. This is useful in order to capture the different meanings of the relationships that can be established in each social network. For example, in those social networks where users usually provide each other with positive ratings as an expression of politeness, such as positively commenting back a positive review on eBay, or following back a new follower in Twitter. This phenomenon is called *nepotistic relationships* and it can be used for malicious users in order to gain a high number of followers by following a huge number of users who (hopefully) will follow them back. It is taken into account by some works [3,4] when it comes to compute the reputation of the users in those networks.

There exist in the bibliography several propagation algorithms intended to compute the trustworthiness of the users in a social network. One of the first works on this task was the EigenTrust algorithm [12], that aims to reduce the number of inauthentic file downloads in a P2P network. In the EigenTrust algorithm, each user calculates the local trust value for all peers voting to him. The global trust value is obtained by aggregating the normalized local trust values with respect to a peer. EigenTrust claims to take into account both positive and negative feedback from the users, but it actually assigns a trust score of 0 to every user whose local trust score is negative. So the algorithm only works with graphs whose edges have always positive weights.

Other technique that only works with positively weighted graphs is TidalTrust [19], a proposal intended to measure the trust between two users in a social network. It processes the trust graph, constructed by the relations between users. The algorithm computes the trust score for each pair of users, $t_{u,v}$, relying on the direct experience of all the nodes that constitute the shortest path from u to v in the network of trust. The algorithm follows the Eq. (1), to recursively compute the trust between u and v in terms of $t_{w,v}$, where w is a node of the path from u to v.

$$t_{u,v} = \frac{\sum_{w \in T_u \cap t_{u,w} > \epsilon} t_{u,w} t_{w,v}}{\sum_{w \in T_u \cap t_{u,w} > \epsilon} t_{u,w}} \quad (1)$$

where ϵ is the threshold that limits the strength of the paths to be considered. The strength of a path $\{a, b, c\}$ is computed as the minimum between the rating between a and b and the rating between b and c. In the cited work they show the performance of the algorithm using a dataset extracted from FilmTrust,[1] a social network formed by a system for user-generated movies reviews and a recommender system. The recommendation of movies is based on the combination of the user's trust network and the movie ratings created by trusted friends.

[1] trust.mindswap.org/FilmTrust.

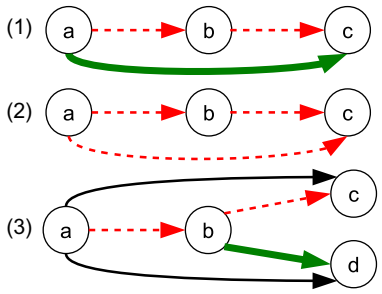

Fig. 1. Transitivity models: (1) Multiplicative distrust. (2) Additive distrust. (3) Neutral distrust. Dashed lines corresponds to negative opinions, while thick lines are positive votes, and thin ones are undecided opinions.

PowerTrust [18] is a TRS that aggregates the positive and negative opinions between the users into the *local trust scores*, similarly to EigenTrust. These scores are propagated through the network by a random-walk algorithm in order to obtain the *global trust scores* for each user in the network. The novelty in this system is that it establishes a method intended to select a set of *power* peers. The opinions from these users will be propagated through the network with more strength than the opinions from the rest of users. This method has the advantage of automatizing this process, but it presents a major disadvantage: since it is an unsupervised process, it is susceptible to being used by malicious users for their benefits, so it constitutes a potential weak point in the TRS.

In [9], the concept of *transitivity of distrust* is introduced, representing the way in which negative opinions of users can be spread over the social network. There are different assumptions that can be adopted when we try to estimate the trustworthiness of a node in a network, depending on the transitivity of distrust. The multiplicative distrust follows the assumption of *"The enemy of my enemy is my friend"*, it implies that if user *a* distrusts *b*, and *b* distrusts *c*, then *a* trusts *c*. In an additive distrust model, given the previous example, *a* should strongly distrust *c*, because *c* is not respected by *b*. In other words: *"Don't respect someone not respected by someone you don't respect"*. Finally, the neutral distrust can be stated as: *"Don't take into account your enemies' opinions"*. It implies that if *a* distrust *b*, then we cannot make any inference for neighbors of node *b*.

The transitivity models are graphically represented in Fig. 1. The dashed lines represent distrust between the related nodes, the thick edges represent trust, and the thin solid ones are undetermined relations.

Other work that studies a social network with positive and negative opinions is presented in [10]. They analyze the trust and reputation mechanism in Slashdot.org, where users can establish relations of friend or foe with other users in the network. They prove the multiplicative distrust transitivity model of this kind of networks. Furthermore, taking the network as dataset, some centrality measures are compared with the two proposals of the paper, Signed Spectral Ranking and Negative Ranking which take into account positive and negative opinions in the social network. Both techniques present a very good performance in this task, although they do not take into account the effects of the transitivity of distrust.

On the other hand, there are also many works that analyze the risks and difficulties of TRS's. A set of common security vulnerabilities for TRS's are identified in [7,8]: the initial window problem (or cold start) which occurs in TRS's that rely only on the user direct experiences, because new users do not have any information about the trustworthiness of the users in the system; the re-entry problem, which points out the impossibility of establishing the real identity of a user, allowing one user to create several accounts in the system to favor one to another; finally, the exit problem is the negative behavior of a user who is planning to leave the social network, and who has no further need for his good reputation. Most of them are difficult to totally avoid, so it is a good approach to try to minimize their negative effects.

Some works analyze other threats for the robustness of TRS's. In [6,2] the threats against a TRS are classified regarding the methods used in the attacks. They distinguish several types of attacks: *self-promotion*, where an attacker manipulates its own reputation by falsely increasing it; *whitewashing*, whose aim is to restore the reputation of an attacker after been penalized by a TRS; *slandering*, where a user votes against other users in order to decrease their reputation; and *orchestrated attacks*, where a group of users form a team to perform a coordinated attack.

Several threat models are presented in [11,12]. They take the example of a P2P network for file sharing, in order to explain the methods used by malicious users to achieve their goal. In many senses, a social network can be viewed as a P2P network, in terms of a decentralized network where users can share and rate different resources (texts, videos, images, etc.).

It is assumed that good users, in terms of trust and reputation, will always be honest, so they will receive positive links from the rest of good users. In this situation, the main threat models to interfere in the overall ranking of trust can be described as follows [11]:

- **Threat Model A:** *Individual Malicious Peers*. Malicious users always present a bad behavior, so they receive negative links from good users.
- **Threat Model B:** *Malicious Collectives*. Based on previous model, adding the possibility of bad peers to assign positive trust values to other malicious users. In this way, the ranking of malicious users can be increased due to the amount of positive in-links received.
- **Threat Model C:** *Camouflage behind good transactions*. Malicious peers can cheat some good users to vote positively for them. The effect in the network is that some bad users can sporadically receive a positive vote from a good user.
- **Threat Model D:** *Malicious spies*. There are two types of malicious users: some of them act as in threat model B and C; and the others, called *spies*, who make good users to vote positively for them, and assign positive trust values only to bad nodes.
- **Threat Model E:** *Camouflage behind judgments*. In this model, malicious peers assign negative trust values to good users. This strategy can cause the decrease of trust for good peers and, as a consequence, the promotion of the malicious peers in the ranking of trust.

3. Propagation of trust and distrust

In this section we introduce our proposal for the computation of the trustworthiness of the users in an on-line network by propagating their positive and negative opinions. Our proposal takes as input a weighted directed graph modeling a social network, where the nodes are the users of the network and the edges represent the relations between those users. The weights of the edges contain information about the type of the relation established between two users. Given i and j, two users of a social network, we define p_{ij} as the weight of the edge from user i to user j. If p_{ij} is positive, it means that i is a follower of j. In other case, i would be a detractor of j.

In the remainder of this section we briefly explain PolarityRank, a propagation algorithm introduced in [13,14] to propagate positive and negative information through a graph. Then, we present PolarityTrust, our trust and reputation algorithm that adapts PolarityRank in order to deal with the trust and reputation task, facing some common vulnerabilities in TRS's, such as the orchestrated attacks and the dishonest votes [2].

3.1. PolarityRank

PolarityRank is based on the similar ideas as PageRank [15], but it extends its functionalities in order to handle graphs with positive and negative edges. PolarityRank defines two different ranking values for each node in the graph, *Positive PolarityRank* (PR^+) and *Negative PolarityRank* (PR^-). Formally, let $G = (V, E)$ be a directed weighted graph with the set of vertices V and the set of directed edges E. Given two nodes, $v_i,\ v_j \in V$, we define p_{ij} as a real valued attribute that represents the weight of the edge from v_i to v_j, with $p_{ij} \neq 0$. For a given vertex v_i, let $In^+(v_i)$ and $In^-(v_i)$ be the sets of vertices that point to it (predecessors) with positive and negative edges, respectively. And let $Out(v_i)$ be the set of vertices that v_i points to (successors). The scores can be obtained as it is shown in Eqs. (2) and (3).

$$PR^+(v_i) = (1-d)e_i^+ + d\left(\sum_{j\in In^+(v_i)} \frac{p_{ji}}{\sum_{k\in Out(v_j)}|p_{jk}|}\ PR^+(v_j) + \sum_{j\in In^-(v_i)} \frac{-p_{ji}}{\sum_{k\in Out(v_j)}|p_{jk}|}\ PR^-(v_j)\right) \quad (2)$$

$$PR^-(v_i) = (1-d)e_i^- + d\left(\sum_{j\in In^+(v_i)} \frac{p_{ji}}{\sum_{k\in Out(v_j)}|p_{jk}|}\ PR^-(v_j) + \sum_{j\in In^-(v_i)} \frac{-p_{ji}}{\sum_{k\in Out(v_j)}|p_{jk}|}\ PR^+(v_j)\right) \quad (3)$$

where d is the damping factor that represents the probability of choosing a neighbor of the current node in the next step of the random-walk. e^+ and e^- are the personalization vectors. They are intended to compute a biased ranking algorithm in which some nodes have higher probability of being visited than others. Intuitively, these vectors represent the *a priori*relevance of the nodes in the network.

The propagation algorithm iterates over the nodes in the network, applying the Eqs. (2) and (3). The process ends when the maximum difference between the scores in one iteration and the previous one, is lower than a given threshold, t. The algebraic and convergence proofs of PolarityRank are detailed in [13].

The basic idea behind PolarityRank is to classify the nodes in a network into positive and negative, regarding their relationships to a set of known positive and negative nodes, namely *seeds*. The algorithm is intended to propagate the information of these seeds to the rest of the nodes through the edges of the graph, in such way that the nodes are influenced by their closeness to positive or negative seeds.

Apart from the ability to process a graph with positive and negative relationships between its nodes, we have chosen PolarityRank as the basis of our system due to its flexibility when it comes to enrich the algorithm with suitable elements for the task of computing the reputation of users in a social network.

3.2. Our proposal

In this section we present our proposal, an adaptation of the ideas behind PolarityRank, intended to deal with the problem of estimating the trustworthiness of the users in a social network. In this case, the graph is built taking the users of the network as the nodes of the graph, the edges representing the opinions of some users about others, and the weights of the edges corresponding to the intensity of the relationship between the nodes. This *intensity* of the relations can be measured in different ways, depending on the type of social network that we are processing. For example, in an on-line marketplace (such as eBay.com), the weight of an edge, p_{ij}, can be an aggregation of all the opinions of user i about user j, computed as the difference between the amount of satisfactory and unsatisfactory transactions between these two users.

PolarityTrust computes an algorithm similar to PolarityRank in order to obtain two scores for each user: the first one representing the positive reputation of the user (PR^+), and the other one representing the negative reputation (PR^-). The final score for each user, $Trust(i)$, is obtained as follows:

$$Trust(i) = \frac{PR^+(i) - PR^-(i)}{PR^+(i) + PR^-(i)}$$

where $-1 \leqslant Trust(i) \leqslant +1$, being -1 the maximum distrust value and $+1$ the maximum trust value, corresponding to a totally untrustworthy and a totally trustworthy user, respectively.

As mentioned above, some social networks establish a group of authorities or moderators (Slashdot, Kuro5hin[2]). The opinions of these users are more relevant than the opinions from the rest of users. In our system, we can take into account this fact by considering a special group of nodes as *sources of trust*. The intuition behind this concept is that the opinions from these users must be propagated over the

[2] www.kuro5hin.org/.

network with more strength than the opinions from the rest of the users. In this way, their opinions can be more influential in the propagation algorithm than the opinions from the rest of users in the network. This intuition is included in our algorithm using the vector e^+, which contains a score for each user that is considered a *source of trust*. The vector is initialized as follows:

$$e_i^+ = \begin{cases} \frac{1}{|Sources^+|} & i \in Sources^+ \\ 0 & otherwise \end{cases}$$

where $Sources^+$ is the set of users taken as sources of trust. This intuition can be extended in order to obtain a set of *sources of distrust*, who can provoke in the algorithm the opposite effect of the sources of trust. In this way, the reputation of the users who are positively linked from a source of distrust, u, must be decreased, because the distrust score of u is propagated to them. Analogously to the previous approach, the users that are taken as sources of distrust, have an *a priori* score in the vector e^-.

This model implements as trust and distrust transitivity model the multiplicative distrust assumption: *"The enemy of my enemy is my friend"*, graphically represented in Fig. 1(1). So a node i gains high trust score in two cases, when:

1. A highly trusted node votes positively for i, because the high trustworthiness score of the voting node increases the trust score of i. In other words, node i is related to a trustworthy user, so it must be considered trustworthy as well.
2. A highly distrusted node votes negatively for i, because the inverse relation between both nodes makes the high untrustworthiness of the voting node rise the trustworthiness of i. This situation can be caused by the fact that node i has an opposite position (opinion) of an untrustworthy user, so it must be taken as a trustworthy user.

Analogously, a node i gains high distrust score if:

1. A highly trusted node votes negatively for i, because the inverse relation between the nodes transforms the high trust score of the voting node into a high distrust score for the target.
2. A highly distrusted node votes positively for i, because node i is considered to be closely related to an untrustworthy user.

Furthermore, PolarityTrust includes two mechanisms, *Non-Negative Propagation and Action–Reaction Propagation*, intended to specifically deal with the problem of trust and reputation in social networks. We discuss them in depth in next sections.

3.2.1. Non-Negative Propagation (PR_{NN})

It has been shown in Section 2 that malicious users have many ways to take advantage of the weaknesses in the trust and reputation algorithms. In order to avoid the influence of bad nodes in the final ranking, we integrate in the PolarityTrust algorithm the capability of deciding whether the opinion of the users will be taken into account or not.

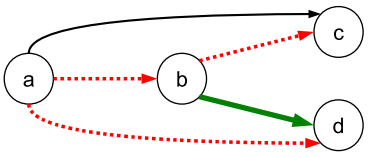

Fig. 2. Transitivity model adopted by the Non-Negative Propagation approach. Dashed lines corresponds to negative opinions, while thick lines are positive votes, and the thin solid ones are undetermined relations.

Thus we can minimize the influence of malicious peers in the ranking by allowing only some users to propagate their opinions over the network. In this case, we adopt an hybrid model for the transitivity of distrust, graphically explained in Fig. 2. We avoid the propagation of negative opinions from bad users, so they cannot harm the trust scores of good users. This feature can help the system to deal with possible attacks based on the use of negative votes, such as the threat model E. As it is shown in the figure, if user a distrusts b and b trusts d, then a distrusts d (multiplicative distrust), but if b distrusts c, a does not take this opinion into account (neutral distrust).

The positive opinions from bad users are already included in the basic mechanism of PolarityTrust, taking this information into account depending on the target of the links. In this way, Non-Negative propagation approach aims to protect the global ranking from some actions of malicious users, because their negative opinions will not be propagated.

Given the scores PR^+ and PR^-, we can dynamically classify a user, v_i, as good or bad by checking the sign of $Trust(v_i)$, as follows:

$$Sign(v_i) = \begin{cases} -1 & Trust(v_i) < 0 \\ 1 & otherwise \end{cases}$$

So, the basic PolarityRank with the addition of Non-Negative propagation is computed as it is shown in the Eqs. (4) and (5).

$$PR^+_{NN}(v_i) = (1-d)e_i^+ + d\left(+\sum_{j\in In^+(v_i)} \frac{p_{ji}}{\sum_{k\in Out(v_j)}|p_{jk}|}\ PR^+_{NN}(v_j) + \sum_{j\in In^-(v_i),\ Sign(v_j)>0} \frac{-p_{ji}}{\sum_{k\in Out(v_j)}|p_{jk}|}\ PR^-_{NN}(v_j)\right) \quad (4)$$

$$PR^-_{NN}(v_i) = (1-d)e_i^- + d\left(\sum_{j\in In^+(v_i)} \frac{p_{ji}}{\sum_{k\in Out(v_j)}|p_{jk}|}\ PR^-_{NN}(v_j) + \sum_{j\in In^-(v_i),\ Sign(v_j)>0} \frac{-p_{ji}}{\sum_{k\in Out(v_j)}|p_{jk}|}\ PR^+_{NN}(v_j)\right) \quad (5)$$

This mechanism allows PolarityTrust to take a node as good or bad, depending on the scores obtained in each iteration of the algorithm. It means that, in a given iteration t, we can take node i as a bad user (and consequently do not propagate its opinions), and the next iteration, $t+1$, the same node could be taken as a good user due to the scores of its neighbors in that iteration.

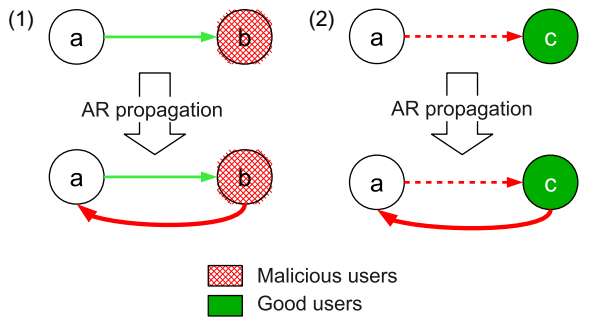

Fig. 3. Action–Reaction against incoherent judgments: node b is a malicious user, node c is a positive one, and node a is a dishonest user voting them. Dashed lines corresponds to negative opinions, while thick lines are positive votes. AR Propagation creates a negative vote from nodes b and c to node a, in order to penalize its incoherent judgments.

3.2.2. Action–Reaction Propagation (PR_{AR})

The main aspect of the behavior of malicious users is the set of relations that they establish, intended to gain some kind of benefits in terms of good reputation in the network. Most of the attacks that these users perform against a TRS consist in providing incoherent votes to other users in the network. In this section we explain the last extension of our TRS, that addresses the dishonest voting problem. This vulnerability is based on the concept of *malicious spies* that are users who seem to be good but only assign positive trust values to malicious peers. This kind of attacks can be blocked by studying the coherence or incoherence of the opinions in the network. This problem is called *Dishonesty* in [11], defining as *Positive Dishonesty* the situation in which a user assesses positively malicious peers, and analogously for *Negative Dishonesty*.

Action–Reaction Propagation is the method that we propose to deal with *Dishonesty*. It is graphically explained in Fig. 3. It consists in penalizing those users who have incoherent judgments, namely *dishonest users*, by the dynamic inclusion of virtual negative votes against incoherent opinions. In order to judge a vote as dishonest or not, we define the *polarity* of the edge from node i to node j as:

$$Polarity(i,j) = \begin{cases} -1 & p_{ij} < 0 \\ 1 & otherwise \end{cases}$$

where p_{ij} is the weight of the edge from i to j. The penalization depends on the level of dishonesty of each user, in other words, the total number of incoherent judgments. The penalization score, $AR(i)$, is computed as follows:

$$AR(i) = \frac{\sum_{j \in Out(i), Sign(j) \neq Polarity(i,j)} Trust(j)}{\sum_{k \in Out(i)} Trust(k)} \quad (6)$$

where $Sign(j) \neq Polarity(i,j)$ is true if node i assigns a positive vote to a negative node, or if it assigns a negative vote to a positive node. This penalization only affects the negative score of each node. The formula for PR^+ is the same discussed in Section 3.1, while PR^- is now obtained as shown in Eq. (7).

$$PR^-_{AR}(v_i) = (1-d)e^-_i + d \left(\sum_{j \in In^+(v_i)} \frac{p_{ji}}{\sum_{k \in Out(v_j)} |p_{jk}|} PR^-_{AR}(v_j) + \sum_{j \in In^-(v_i)} \frac{-p_{ji}}{\sum_{k \in Out(v_j)} |p_{jk}|} PR^+_{AR}(v_j) \right) + \frac{AR(i)}{\sum_{k \in V} AR(k)} \quad (7)$$

Note that the AR model does not reward the coherent votes, but penalizes the incoherent ones. This observation is important due to the possibility of an attack consisting in a variation of threat model E, in which malicious users vote negatively for good users. If the coherent votes are rewarded, a malicious user could gain high reputation by voting positively for good users. This approach also avoids this possibility by only penalizing the incoherent votes.

3.2.3. PolarityTrust

Our proposal, PolarityTrust, combines both extensions, Non-Negative Propagation and Action–Reaction Propagation, in order to take advantage of their ideas. In this way, the opinions of bad users are not propagated to their neighbors, and all the nodes with a dishonest behavior are penalized in the ranking of trust. This combined model follow the Eqs. (8) and (9).

$$PT^+(v_i) = (1-d)e^+_i + d \left(\sum_{j \in In^+(v_i)} \frac{p_{ji}}{\sum_{k \in Out(v_j)} |p_{jk}|} PT^+(v_j) + \sum_{j \in In^-(v_i), Sign(v_j) > 0} \frac{-p_{ji}}{\sum_{k \in Out(v_j)} |p_{jk}|} PT^-(v_j) \right) \quad (8)$$

$$PT^-(v_i) = (1-d)e^-_i + d \left(\sum_{j \in In^+(v_i)} \frac{p_{ji}}{\sum_{k \in Out(v_j)} |p_{jk}|} PT^-(v_j) + \sum_{j \in In^-(v_i), Sign(v_j) > 0} \frac{-p_{ji}}{\sum_{k \in Out(v_j)} |p_{jk}|} PT^+(v_j) \right) + \frac{AR(v_i)}{\sum_{k \in V} AR(v_k)} \quad (9)$$

4. Experimental settings

The experiments in this work are designed to evaluate the reliability of our proposal. With this aim, we detail in this section the configuration, datasets and evaluation metrics applied.

Regarding the two parameters of PolarityTrust algorithm, we have set d to 0.85, the common value used in most of the papers about PageRank and similar algorithms; and the error threshold has been set to 0.001.

In the reminder of this section, we present the datasets used in this work, consisting in a set of randomly generated graphs (Section 4.1) and a real-world dataset extracted from Slashdot.org, a social news site (Section 4.2). The graph models built from these datasets contain nodes representing the users in the social network and a set of edges representing the relationships between those users. In the case of the randomly generated graph models, the weights of these edges represent the aggregation of (positive and negative) opinions of the users. In the dataset from Slashdot.org we have two possible weights for a given edge, p_{ij} : 1 if i has tagged j as a friend, or -1 in the other case.

In Section 4.3 we explain the additional methods used in the experiments to compare the results. Finally, in Section

4.4 we briefly discuss the metrics used for the evaluation of the experimental results.

4.1. Graph generation

We use a method to randomly generate graphs with a topology similar to real-world networks. These methods are frequently used in works on network analysis [4,5] due to the difficulty in obtaining datasets from the real world, and also the necessity of creating random networks in order to study the behavior and mechanisms produced in the real ones.

One of the most cited studies on large real-world networks is presented in [16]. This work introduces a method for the generation of random graphs that models real-world networks. The main aspect of these graphs is that they present the *preferential attachment* property. It consists in the fact that the new users of a network attach preferentially to nodes that are already well connected. The work also introduces the Barabasi method, intended to generate graphs with this property. We use this method to generate the datasets for our experiments. The method begins with a network of at least two nodes, and the degree of each node should be at least 1, in order to generate a connected network. New nodes are added to the network one at a time. Each new node is connected to a number of existing nodes with a probability:

$$p_i = \frac{k_i}{\sum_j k_j}$$

where k_i is the out-degree of node i. In this way, the degrees follow a power-law distribution and the network shows the preferential attachment property.

Since we want to determine the performance of our proposal against different types of malicious behaviors, we need to generate a graph model for each attack. The Barabasi method is implemented to generate the nodes and edges modeling the good user behaviors. For the experiments, we have generated graphs with 10^4 nodes. Once the basic graph is generated modeling the behaviors of good users in a social network, we add the malicious users and their relations. The nodes and edges of malicious users are generated according to each attack. Given a randomly generated graph, we add the malicious nodes and create the negative links from good users to them regarding the intensity of threat model A. Then the edges that will form the desired attacks are created with a certain probability, depending on the intensity of each attack: the more intense a threat model is required to be, the more probable the creation of an edge modeling this attack.

4.2. Slashdot Zoo dataset

In addition to this randomly generated datasets, we have tested our proposal against a real-world dataset, namely *Slashdot Zoo*, presented in [10]. It is a crawling of the Slashdot.org network that is a social network founded in 1997, focused on the publication, recommendation and discussion of news mainly related to technology. The Slashdot.org system provides the users with the ability of establishing two types of relationships: friend (positive link) or foe (negative link). In the beginning, Slashdot established a group of 25 moderators. This amount was increased to 400 due to the growth in the number of users. Nowadays, the moderator team is automatically selected. The Slashdot TRS consists in three layers: the first one is for rating the content of the network (comments and articles), while the second one moderates the raters. In addition, Slashdot staff members are able to moderate any element in the system (comments, articles and participants).

The dataset crawled in [10] is formed by the users in the system and their friend-or-foe relations. It contains about 71500 users and more than 510K = 510,000 edges, being 24% negative links. The gold standard is given by a special user, called *NoMoreTrolls*, who has all the known trolls of Slashdot in its list of foes. Since we want to follow the same experimental settings of [10], we have considered the same set of 96 trolls.

4.3. Additionally implemented methods

In this section we present the techniques taken as baselines in the experiments (EigenTrust and Fans Minus Freaks) as well as two more sophisticated methods (Signed Spectral Ranking and Negative Ranking) that, like our proposal, use positive and negative links to compute trust values.

The first baseline method is the EigenTrust algorithm [12]. It aims to reduce the number of inauthentic file downloads in a P2P network. In the EigenTrust algorithm, each user calculates the local trust value for all peers voting to him. The global trust value is obtained by aggregating the normalized local trust values with respect to a peer. Formally, given C, a matrix where c_{ij} represents the opinion of i about j (local trust value), the EigenTrust algorithm computes the global trust values as:

$$\bar{t}_i = C^T \bar{c}_{ij}$$

where $\bar{t}_i$ is the vector of local trust values of i for each node in the network. Repeating this process, $\bar{t}_i$ will converge to a stable value, $\bar{t}$, that is the vector containing the EigenTrust values for each node. This vector is the *left principal eigenvector* of the matrix C.

The second baseline is the heuristic *Fans Minus Freaks*. It is a simple metric that relies only in the direct experience of the users, taking into consideration the detractors and followers of each one to compute its score. The trust score of node v_i is the difference between the number of positive and negative links pointing to i, obtained as follows:

$$FmF(v_i) = |In^+(v_i)| - |In^-(v_i)|$$

Finally, we have implemented the two methods proposed in [10], *Signed Spectral Ranking* (SR) and *Negative Ranking* (NR). The first one is a popularity measure consisting in applying PageRank algorithm directly to the graph, including negative edges. Negative Ranking includes both PageRank and SR in the computation, it is defined as follows:

$$NR = SR - \beta \cdot PR$$

where $\beta \geqslant 0$ determines the influence of PageRank on the final ranking. In [10], the best results are achieved by setting $\beta = 1$, so it is done in these experiments.

Unlike this method, in our proposal both positive and negative links are propagated in such way that they influence both positive and negative scores of the users, being able to implement the transitivity of trust and distrust in a social network.

4.4. Evaluation metrics

Since the aim of these techniques is to demote the bad users in the ranking and to promote the good ones, we use the *number of inversions* as one of the evaluation metrics. In other words, we evaluate the performance of the techniques in terms of the number of bad users that appear in the positions of the ranking corresponding to good users. The lower the metric value, the better is the performance of the technique. In order to make the results comparable, we normalize this metric obtaining the error rate of each technique, computed as follows:

$$\text{ErrorRate} = \frac{\text{NumberOfInversions}}{\text{TotalNumberOfBadUsers}}$$

On the other hand, we have included in the evaluation of the experiments the standard *nDCG* (Normalized Discounted Cumulative Gain) [17]. In this way, we do not only evaluate the fact of avoiding bad users in top positions, but also the specific ranking achieved by these malicious users. *nDCG* is obtained as follows:

$$nDCG_p = \frac{relevance_1 + \sum_{i=2}^{p} \frac{relevance_i}{log_2 i}}{IDCG_p}$$

where p is the given position of the ranking, $relevance = 1$ if a user is good, and 0 if not. Finally, *IDCG* is the Ideal Discounted Cumulative Gain, that is the best DCG that can be obtained for the given dataset. In our case, we have evaluated the positions of the bad users in the ranking, penalizing the results where good users have low positions.

5. Experimental results

In this section we present the results of the experiments performed following the settings explained above. First, in Section 5.1 we test our proposal against an incremental combination of the basic attacks reviewed in Section 2. We also show the performance of our approaches with a real-world social network using the Slashdot Zoo dataset (Section 5.2). Then, in Section 5.3, we test our proposal with a combination of a real and synthetic data, by modifying the Slashdot Zoo dataset in such way that the tagged trolls perform the attack models introduced in Section 2. Finally, we present in Section 5.4 some experiments intended to test the usefulness of the inclusion of the *sources of distrust* in our algorithm.

5.1. Basic attack models

The first set of experiments is intended to show the performance of our approaches against the basic attacks explained in Section 2, and incremental combinations of them. In these experiments, we have generated graphs with 10^3 malicious users. These nodes receive negatives votes from the good users with a probability of 0.8 (intensity of threat A). The intensity of threat model B has been set to 100%, meaning that every bad node votes positively for the rest of malicious users in the graph. Model C intensity has been set to 25%, given that a value near 50% or higher will transform any bad node into a good one. Finally, the 50% of the bad nodes vote negatively for good users, performing the model E. With these parameters, the resulting graphs have about 10^5 edges, of which 25% are negative links. The graphs implementing the threat model D additionally have 100 spy nodes. Finally, we have used 10 nodes as sources of trust for all the experiments.

The error rate for each technique is shown in Table 1 where we can see that our system obtains very good results, improving EigenTrust and Fans-Minus-Freaks. SR and NR results are between PT and PT + AR, improving EigenTrust and FmF as well. This fact shows the relevance of using the negative links in the computation of trust.

In Table 2 we show the nDCG, measuring the demotion of bad users in the ranking of trust. In this case, both extensions Non-Negative and Action-Reaction propagation present better performance than the baselines, while the complete technique is the best method in accordance to these results.

5.2. Slashdot dataset experiments

Up to this point we have performed experiments with a set of randomly generated datasets, using the Barabasi and Albert generation method in order to obtain graphs that model real-world networks. In this section we show the results of our proposals on a real-world dataset: the Slashdot Zoo (see Section 4.2). The results of each technique have been evaluated using the error rate and the nDCG. In both cases we have evaluated the number of bad users in positions above the last 96 of the ranking. In Table 3 we show the results.

For these experiments, the set of sources of trust is formed by the user *CmdrTaco*, corresponding to the founder of Slashdot.org Rob Malda, and the users that he has tagged as friends. There are 6 sources of trust in total.

We show the results of these experiments in Table 3. The scores in the error rate for every technique are higher than those in previous section due to the proportion of bad nodes in the Slashdot Zoo dataset (0.0013) in contrast to

Table 1
Error rate for each technique against incremental attacks: EigenTrust (ET), Fans Minus Freaks (FmF), Signed Spectral Ranking (SR), Negative Ranking (NR), Non-Negative propagation approach (PR_{NN}), Action–Reaction approach (PR_{AR}) and PolarityTrust (PT). For the attacks including the model D, we insert 100 spy nodes in the graph. The best results for each experiment are highlighted in bold.

Threats	ET	FmF	SR	NR	PR_{NN}	PR_{AR}	PT
A	0.535	0.446	0.354	0.271	0.265	0.175	**0.087**
A, B	0.535	0.446	0.355	0.272	0.265	0.175	**0.087**
A, B, C	0.526	0.649	0.345	0.272	0.256	0.166	**0.106**
A, B, C, D	0.528	0.650	0.344	0.272	0.255	0.166	**0.116**
A, B, C, D, E	0.527	0.527	0.345	0.282	0.261	0.169	**0.110**

Table 2
nDCG for each technique against incremental attacks: EigenTrust (ET), Fans Minus Freaks (FmF), Signed Spectral Ranking (SR), Negative Ranking (NR), Non-Negative propagation approach (PR_{NN}), Action–Reaction approach (PR_{AR}) and PolarityTrust (PT). For the attacks including the model D, we insert 100 spy nodes in the graph. The best results for each experiment are highlighted in bold.

Threats	ET	FmF	SR	NR	PR_{NN}	PR_{AR}	PT
A	0.833	0.843	0.599	0.749	0.876	0.906	**0.987**
A, B	0.833	0.844	0.811	0.920	0.876	0.906	**0.987**
A, B, C	0.842	0.719	0.816	0.920	0.877	0.903	**0.984**
A, B, C, D	0.823	0.723	0.818	0.937	0.879	0.903	**0.984**
A, B, C, D, E	0.753	0.777	0.877	0.933	0.966	0.862	**0.982**

the randomly generated datasets of previous section (10%). In this case, *nDCG* is a more appropriate metric to compare the techniques because their values are normalized.

In accordance with the data in Table 3, EigenTrust algorithm does not demote any of the 96 trolls in the last positions of the ranking, while FmF and SR perform slightly better and NR gets a very good result. Finally, our approaches outperform NR, presenting the best performance with this dataset, achieving a 10% of improvement over the rest of methods. This experiment highlights the usefulness of our proposal in a real environment.

5.3. Trolling Slashdot Zoo dataset

In this section we adopt the role of trolls in Slashdot.org. We try to perform some attacks against the network through the 96 tagged trolls in the Slashdot Zoo dataset. In this way, we can measure the impact of the different trolling techniques discussed in Section 2, and compare the behavior of the implemented techniques.

In order to create the datasets for these experiments we apply the same mechanism explained in Section 4.1 for the generation of bad users and their out-links and in-links. Instead of taking a randomly generated graph as the network under attack, we have used the Slashdot Zoo dataset. The different attacks have been applied to this graph by adding a number of edges modeling the required behavior of the bad nodes. No new nodes have been added to the original dataset, so the amount of good and bad users is the same as in previous experiments.

The experiments are evaluated applying the same metrics previously mentioned. In Table 4 we present the values of the rate of errors for each technique against each threat model, counting the number of good users that appear in the last positions of the rankings. We can see that the

Table 3
nDCG and error rate for each technique processing Slashdot Zoo dataset. EigenTrust (ET), Fans Minus Freaks (FmF), Signed Spectral Ranking (SR), Negative Ranking (NR), Non-Negative propagation approach (PR_{NN}), Action–Reaction approach (PR_{AR}) and PolarityTrust (PT). The best results for each experiment are highlighted in bold.

	ET	FmF	SR	NR	PR_{NN}	PR_{AR}	PT
Error %	0.990	0.901	0.881	0.881	**0.861**	**0.861**	**0.861**
nDCG	0.310	0.460	0.479	0.477	**0.593**	0.570	0.588

Table 4
Error rate for each technique processing the Slashdot Zoo dataset with synthetic attacks: EigenTrust (ET), Fans Minus Freaks (FmF), Signed Spectral Ranking (SR), Negative Ranking (NR), Non-Negative propagation approach (PR_{NN}), Action–Reaction approach (PR_{AR}) and PolarityTrust (PT). For the attacks including the model D, we insert 100 spy nodes in the graph. The best results for each experiment are highlighted in bold.

Threats	ET	FmF	SR	NR	PR_{NN}	PR_{AR}	PT
A	0.990	0.901	0.881	0.881	**0.861**	**0.861**	**0.861**
A, B	1.000	0.901	0.880	0.880	**0.861**	**0.861**	**0.861**
A, B, C	1.000	0.901	0.881	0.881	**0.861**	**0.861**	**0.861**
A, B, C, D	1.000	0.925	0.935	0.940	0.851	0.856	**0.841**
A, B, C, D, E	1.000	0.925	0.940	0.935	0.851	0.856	**0.841**

Table 5
nDCG for each technique processing the Slashdot Zoo dataset with synthetic attacks: EigenTrust (ET), Fans Minus Freaks (FmF), Signed Spectral Ranking (SR), Negative Ranking (NR), Non-Negative propagation approach (PR_{NN}), Action–Reaction approach (PR_{AR}) and PolarityTrust (PT). For the attacks including the model D, we insert 100 spy nodes in the graph. The best results for each experiment are highlighted in bold.

Threats	ET	FmF	SR	NR	PR_{NN}	PR_{AR}	PT
A	0.310	0.460	0.479	0.477	**0.593**	0.570	0.588
A, B	0.308	0.460	0.478	0.477	**0.593**	0.570	0.588
A, B, C	0.311	0.460	0.474	0.484	**0.593**	0.570	0.588
A, B, C, D	0.370	0.476	0.501	0.501	0.580	0.570	**0.586**
A, B, C, D, E	0.370	0.475	0.501	0.496	0.580	0.574	**0.588**

Table 6
Error rate for our approaches using some known trolls as sources of distrust, taking as input the Slashdot Zoo dataset with synthetic attacks. Results for Non-Negative propagation approach (PR_{NN}), Action–Reaction approach (PR_{AR}) and PolarityTrust (PT) using sources of trust only, and sources of trust and distrust. The best results for each experiment are highlighted in bold.

Threats	Sources of trust			Sources of trust & distrust		
	PR_{NN}	PR_{AR}	PT	PR_{NN}	PR_{AR}	PT
A	0.861	0.861	0.861	0.475	0.505	**0.465**
A,B	0.861	0.861	0.861	0.475	0.505	**0.465**
A,B,C	0.861	0.861	0.861	0.475	0.505	**0.465**
A,B,C,D	0.851	0.856	0.841	0.657	0.677	**0.642**
A,B,C,D,E	0.851	0.856	0.841	0.652	0.677	**0.637**

Table 7
nDCG for our approaches using some known trolls as sources of distrust, taking as input the Slashdot Zoo dataset with synthetic attacks. Results for Non-Negative propagation approach (PR_{NN}), Action–Reaction approach (PR_{AR}) and PolarityTrust (PT) using sources of trust only, and sources of trust and distrust. The best results for each experiment are highlighted in bold.

Threats	Sources of trust			Sources of trust & distrust		
	PR_{NN}	PR_{AR}	PT	PR_{NN}	PR_{AR}	PT
A	0.593	0.570	0.588	**0.846**	0.790	**0.846**
A, B	0.593	0.570	0.588	**0.846**	0.790	**0.846**
A, B, C	0.593	0.570	0.588	**0.846**	0.790	**0.846**
A, B, C, D	0.580	0.570	0.586	0.775	0.739	**0.782**
A, B, C, D, E	0.580	0.574	0.588	0.774	0.741	**0.781**

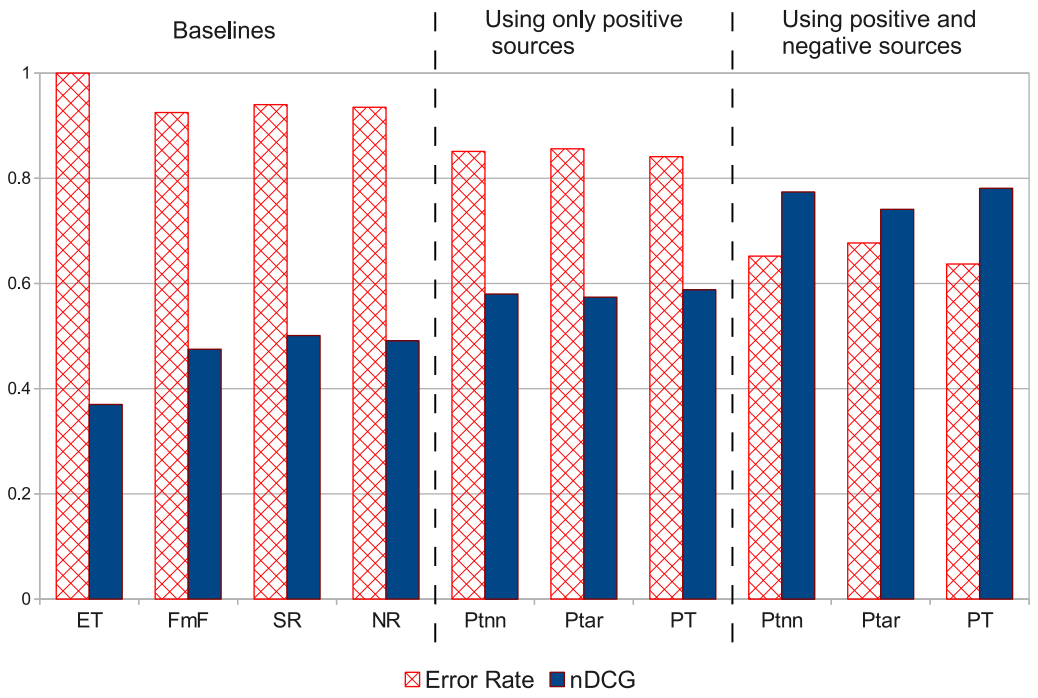

Fig. 4. Error rate and *nDCG* results obtained by all the techniques processing a network under a combination of all the reviewed attacks. EigenTrust (ET), Fans Minus Freaks (FmF), Signed Spectral Ranking (SR), Negative Ranking (NR), Non-Negative propagation approach (*PRnn*), Action–Reaction approach (*PRar*) and PolarityTrust (PT). The results of PolarityTrust with and without negative sources are represented in the chart.

addition of the attacks provoke an increasing number of errors using any of the studied methods but our proposals, which performance remains reasonably stable.

Regarding the positions where bad users are placed in the ranking, we show in Table 5 the *nDCG* values for each experiment. The results of our methods achieve an improvement of about 10% over the rest of techniques. It means that the misclassified bad users are demoted in the ranking so their ranks, according to our approaches, are lower than the ones computed by the other methods.

5.4. Experiments with sources of distrust

All the experiments presented below have been performed with a set of sources of trust, and no *sources of distrust*. In this section we test this feature by taking as input the same datasets of previous section, the Slashdot Zoo with the addition of synthetic attacks. As long as we just try to test whether this feature can be useful in our schema, we have randomly chosen 5 known trolls (foes of the special user *No More Trolls*) as sources of distrust, without following any specific heuristic.

The results of our approaches with the inclusion of the sources of distrust are shown in Tables 6 and 7, corresponding to the error rate and the nDCG, respectively. For better understanding, we include in these tables also the scores of the techniques using only sources of trust.

The improvement achieved by all the approaches in relation to the same techniques without the sources of distrust is evident, outperforming all the previous results. In Fig. 4 we show the results of all the techniques dealing with the combination of all the reviewed attacks.

Regarding these results, this feature can be very useful in a system where some malicious users have been identified, penalizing those users who interact with them (creating a positive edge to any of them, for example). It can be also applicable together with some simple mechanisms that help to (vaguely) identify users who can be considered *a priori* as bad users according to their links, their comments or their behavior in general.

6. Conclusions

In this work we have presented PolarityTrust, a Trust and Reputation System based on the propagation of the positive and negative opinions of the users in a social network in order to compute their trustworthiness. Our proposal uses this information in order to obtain two scores: a positive one indicating the goodness of a user, and a negative one corresponding to its badness. Unlike other approaches, in our method positive and negative links influence both positive and negative scores, being able to implement the transitivity of trust and distrust in a social network.

The reliability of PolarityTrust has been proved by testing its behavior under some common attacks against TRS's, and also with a real-world dataset from the social news site Slashdot.org. The attacks consist in dishonest behaviors of malicious users who can take advantage of the vulnerabilities of a TRS in order to gain a high reputation in the network. We have also introduced a extension to the basic model of PolarityTrust intended to penalize those users who present a dishonest behavior, according to the intensity of this behavior. The experiments on synthetic datasets show that the performances of our approaches

are not affected by the implemented attacks, clearly outperforming the results of other systems. Finally, the experiments with the Slashdot Zoo dataset shows the reliability of PolarityTrust in a real-world social network, even when a few new edges are added in order to perform a set of more sophisticated attacks against the network.

The concept of sources of trust allows to include in our TRS the different levels of users existing in social networks (administrators, moderators, normal users, etc.). With PolarityTrust we can easily assign different weights to the opinions of the users regarding their category in the network. Otherwise, the use of sources of trust and distrust have proved to be a highly reliable method. It could be useful in a semi-supervised schema, serving as an assisting tool for experts or moderators who would only have to assess the (un-) trustworthiness of a small amount of users and delegating the rest of the process to our system.

We plan to further our research by studying other types of attacks against TRS's, including the use of *playbook sequences* [8], consisting in a sequence of actions intended to gain high trustworthiness. There is an infinite set of possible playbook sequences, and they can be influenced by other users playbooks, making this attacks really hard to detect and to avoid. The intuition behind playbook attacks is that it cannot be assessed that a TRS is effective just because the potential attackers do not know how it works. It is also interesting to test some methods for the selection of sources of distrust in the system, such as link-based heuristics.

References

[1] A. Jøsang, R. Ismail, C. Boyd, A survey of trust and reputation systems for online service provision, in: Decision Support Systems, vol. 43, 2007, pp. 618–644.

[2] S. Marti, H. Garcia-molina, Taxonomy of trust: categorizing p2p reputation systems, in: International Journal of Computer and Telecommunications Networking, vol. 50, 2006, pp. 472–484.

[3] D. Gayo-Avello, Nepotistic relationships in twitter and their impact on rank prestige algorithms, CoRR abs/1004.0816, 2010.

[4] Baeza-Yates, Ricardo, Castillo, Carlos, López, Vicente, pagerank increase under different collusion topologies, in: Proceedings of First International Workshop on Adversarial Information Retrieval on the Web, 2005. <http://airweb.cse.lehigh.edu/2005/baeza-yates.pdf>.

[5] Pandit, Shashank, Chau, Duen Horng, Wang, Samuel, Faloutsos, Christos, NetProbe: a fast and scalable system for fraud detection in online auction networks, in: Proceedings of the 16th international conference on World Wide Web, New York, NY, USA, 2007.

[6] K. Hoffman, D. Zage, C. Nita-Rotaru, A survey of attack and defense techniques for reputation systems, in: ACM Computing Surveys, vol. 42, 2009, pp. 1–31.

[7] A. Jøsang, J. Golbeck, Challenges for robust trust and reputation systems, in: the 5th International Workshop on Security and Trust Management, Saint Malo, France, 2009.

[8] R. Kerr, R. Cohen, Smart cheaters do prosper: defeating trust and reputation systems the security of trses, in: Proceedings of the 8th International Joint Conference on Autonomous Agents and Multiagent Systems, Budapest (Hungary), 2009.

[9] R. Guha, R. Kumar, P. Raghavan, A. Tomkins, Propagation of trust and distrust, in: WWW '04: Proceedings of the 13th International Conference on World Wide Web, ACM, New York, NY, USA, 2004, pp. 403–412.

[10] J. Kunegis, A. Lommatzsch, C. Bauckhage, The Slashdot Zoo: mining a social network with negative edges, in: the 18th International World Wide Web Conference, 2009.

[11] D. Donato, M.P. Stefano Leonardi, Combining transitive trust and negative opinions for better reputation management in social networks, in: Workshop on Social Network Mining and Analysis (SNA-KDD), 2008.

[12] S.D. Kamvar, M.T. Schlosser, H. Garcia-molina, The eigentrust algorithm for reputation management in P2P networks, in: Proceedings of the Twelfth International World Wide Web Conference, ACM Press, 2003, pp. 640–651.

[13] F.L. Cruz, C.G. Vallejo, F. Enríquez, J.A. Troyano, Polarityrank: finding an equilibrium between followers and contraries in a network, Journal of Information Processing and Management (2011).

[14] F.L. Cruz, J.A. Troyano, F.J. Ortega, F. Enríquez, Automatic expansion of feature-level opinion lexicons, in: Proceedings of the 2nd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis (WASSA 2011), Association for Computational Linguistics, Portland, Oregon, 2011, pp. 125–131.

[15] L. Page, S. Brin, R. Motwani, T. Winograd, The pagerank citation ranking: bringing order to the web, 1999.

[16] A.-L. Barabási, R. Albert, Emergence of scaling in random networks, Science 286 (1999) 509–512.

[17] M.A. Najork, H. Zaragoza, M.J. Taylor, Hits on the web: how does it compare? in: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information retrieval (SIGIR'07), ACM, New York, NY, USA, 2007, pp. 471–478.

[18] R. Zhou, K. Hwang, Powertrust: a robust and scalable reputation system for trusted peer-to-peer computing, IEEE Transactions on Parallel and Distributed Systems 180 (4) (2007) 460–473. doi:10.1109/TPDS.2007.102. ISSN:1045-9219.

[19] J.A. Golbeck, Computing and Applying Trust in Web-Based Social Networks, PhD thesis. University of Maryland at College Park, College Park, MD, USA, 2005.