GePuTTIS: General Purpose Transitive Trust Inference System for Social Networks

Syavash Nobarany[1], Mona Haraty[1], Dan Cosley[2]
[1]School of ECE, University of Tehran, Tehran 14395-515, Iran
[2]HCI Lab, Cornell University, Ithaca, NY 14850, USA
{nobarany, haraty}@ gmail.com, drc44@cornell.edu

Abstract

Recent work has explored the idea of using trust networks to supplement ratings information in community-based information systems, including algorithms to infer missing values in the trust network. Current trust inference algorithms sometimes make undesirable inferences because they do not fully use information about distrust and sometimes make inferences based on weak support. Further, many algorithms do not consider the problem of trust scope, where one may trust someone's opinions about movies but not books. We present GePuTTIS, a trust inference system that reasons about support levels, distrust, and trust scope. We demonstrate that it improves prediction performance in a collaborative filtering dataset.

Introduction

Web-based social networks have attracted millions of people; the growth of these communities has created new social outlets but weakened modes of community (Putnam 2000). This gives computer scientists the opportunity and responsibility to design effective models of social living that bring value to these communities and their members.

One of the most valuable concepts in people's real world interactions and relationships is trust. System designers use trust-inspired algorithms in applications such as ad-hoc networking and electronic commerce where entities must interact in highly dynamic and unpredictable environments in which traditional security measures are not effective. For example, mapping between names and keys is a crucial issue in secure online transactions. Centralized key administration is costly and does not scale, so researchers have explored using webs of trust to certify participants as an alternative strategy (Jiang, Reeves, Ning 2004). Likewise, in mobile ad-hoc networks, trust-based security measures for admission control are proposed (Gray et al. 2003). In P2P commerce networks such as eBay, explicit trust ratings and extracted trust scores from the feedback comments can be used to predict and prevent undesirable transactions (O'Donovan et al. 2007).

Our focus is on using explicit trust expressions in social networks as a source of information for improving the quality of recommender systems. Trust can be used in a number of ways when making recommendations, including treating the ability of a partner profile to deliver more accurate recommendations in the past as implicit trust (O'Donovan and Smyth 2005) and combining explicit trust expressions with collaborative filtering algorithms (Massa and Avesani 2004)

The process of building trust is complicated and in most cases high levels of trust are achieved through repeated interactions. Almost always the basic level of trust (that leads to the first interaction) is made through referring of a trusted mediator, often called transitive trust. The great capability of collecting and managing information in web-based systems (Hill et al. 1992) may allow us to use transitive trust in web-based social networks more effectively than in the real world.

Trust modeling consists of designing of two basic elements: a trust metric and a transitive trust inference algorithm. TidalTrust is one such algorithm that has been recently used in building effective recommendation systems (Golbeck 2005). However, we believe it can be made even better by considering two key features of trust networks:

- Distrust. Sometimes people will express distrust in others. In those cases, inferences that involve distrusted partners should be discounted.
- Support. If the trust values on an inference chain are low, the algorithm should take into account the fact that the resulting inferences may be faulty.

We have developed an algorithm, GePuTTIS, that addresses the problems of insufficient information and negative trust arcs, making more rational trust estimates in these challenging situations. It does better than TidalTrust on a recommendation task using simulated trust values computed on the MovieLens collaborative filtering dataset.

Basic concepts and definitions

Trust transitivity means, for example, that if A trusts B, and B trusts C, then A will also trust C. This requires a number of considerations, in particular, thinking about the scope of a trust relationship and how trust is represented and expressed by users.

Trust scope

Jøsang, Hayward, and Pope present a nice consideration of the problem of trust scope (Jøsang, Hayward and Pope 2006). For example, the fact that A trusts B as a seller on eBay, and B trusts C to fix his car, does not imply that A trusts C as a seller on eBay, nor for fixing his car. In this

example trust transitivity fails because the scopes of A's and B's trust are different.

Trust scope is defined as the specific fields of trust assumed in a given trust relationship. In other words, the trusted party is relied upon to have certain qualities, and the scope is what the trusting party assumes those qualities to be. For recommendations, we might be interested in two scopes. One is the ability to make useful suggestions of items in the recommendation domain (*functional trust)*; the other is the ability to effectively convey transitive trust and help people reach other trusted people in the social network (*referral trust*). Further, one might want to consider multiple functional scopes—on Amazon, perhaps a given user is very useful for another when recommending books, but has completely opposite tastes in music. Transitive trust schemes do not usually separate these ideas because they lack a scoping mechanism; later, we discuss several ways one might represent these scopes usefully.

Trust metric

In addition to the scope of a trust relationship, there must also be a *trust metric*, a way to express how one member of a group trusts another. It is not required for the members to be humans, as in the PageRank algorithm where group members are web pages (Page et al. 1998). The first forms of trust metrics in computer software were in applications like eBay's Feedback Rating. There are two key points in designing a trust metric. First, users should be able to easily express their degree of trust. Second, the trust inference algorithm must be able to compute with them.

Among different trust inference systems, one of the most interesting metrics was TNA-SL. It considers three factors for expressing trust: trust, distrust, and certainty (Jøsang, Hayward and Pope 2006). We will later argue that a measure of trust certainty is very important for making effective trust recommendations; however, it is likely to be difficult for the user to separate this factor from the trust itself and accurately express both separately. We believe instead that algorithms should generate their own estimates of certainty, or support, for an inferred trust value, and use that information both in making recommendations and in informing users how confidently they might accept the recommendations—an important factor in whether users trust a recommendation system (McNee et al. 2003).

Typically, trust is expressed on a scale, say, from 0 to 1, where 0 is a lack of trust and 1 is complete trust. Such a scale, however, is flawed in that it does not distinguish between distrust and a lack of trust information. In TidalTrust, if A expresses his trust to B by 0, it means that he can't count on B's advice, but this expression is ambiguous. Sometimes, A might want to say "I know B, but I don't know whether to trust him yet." In this case, a trust inference algorithm might try to infer more about whether B should be trusted and use this inferred trust when making recommendations. Other times, A knows that B's advice is not valuable. In this case, the algorithm should not try to infer trust in B. Further, if B exists in paths from A to others, these paths should be ignored.

The GePuTTIS algorithm exploits distrust in order to ignore inference chains that might lead to poor recommendations. Another approach in dealing with distrust arcs, investigated in (Guha et al. 2004), is propagating them. Considering that "the enemy of your enemy is your friend" and "don't respect someone not respected by someone you don't respect", we preferred not to propagate distrust at all, but instead to find and ignore suspicious paths.

Inference Algorithms

The transitive trust inference algorithm is the algorithm that is used to infer trust values where arcs do not exist in the network. A thorough review of such algorithms is available in (Jøsang, Ismail and Boyd 2007). In principle, such algorithms might use information available in the whole network. For example, trust network analysis based on normalisation is used in algorithms such as PageRank (Page et al. 1998) and EigenTrust (Kamvar, Schlosser and Garcia-Molina 2003). Although such algorithms are obviously useful, they have drawbacks as trust inference algorithms for recommendations.

- They are slow compared to algorithms that use only local network information.
- They are not personalized. EigenTrust is used in peer-to-peer systems and takes trust to be a peer's performance. Since a single peer's performance is unlikely to differ much from one peer to another, each node essentially has a single, global trust value. This is not the case in recommendations, where Bob might provide good advice for Alice, but not Cynthia. Trust inference based on max flow, which is used in the Advogato project (Levin and Aiken 1998) also suffers from this problem.
- They do not consider negative trust values.

In practice, systems that make recommendations typically use local algorithms to compute personalized trust values For example, (O'Donovan and Smyth 2005) and (Massa and Avesani 2004) try to combine implicit and explicit trust expressions and collaborative filtering techniques to improve recommendation accuracy. TidalTrust's (Golbeck 2005) approach is the most analogous to GePuTTIS. TidalTrust is deployed in a real system, FilmTrust, a personalized movie rating system, and does better than IMDB average ratings. We will use TidalTrust as a representative transitive trust inference algorithm in the remainder of this paper.

Challenging Situations

The basic form of the TidalTrust formula is shown in equation 1. Given two nodes, TidalTrust uses a weighted average of trust along all paths from the first node to the second to infer the trust from the source to the destination. The basic formulation falls prey to two problems: dealing

with distrust arcs, and making inferences that rely on weak trust information. We discuss each in turn.

Distrust arcs problem

Many inference algorithms ignore referral distrust arcs, because the advice of a distrusted advisor is not important. However, knowing that an advisor is distrusted should be used when computing transitive trust. Most of the proposed algorithms use source to destination recursion: the source node asks its neighbors how much to trust the destination. This type of recursion cannot use the distrust arcs effectively, because for each advisor in a path, only the trust value of the advisor in that path is considered. His overall value for the main advisee is not calculated.

Consider figure 1.a. Forward recursion detects two paths: 1. S-C-D and 2. S-A-C-D. Path 1 does not tell us anything, because C is a distrusted advisor. However, path 2 suggests that we can trust D because no distrust arc is detected. But this is an inaccurate inference because C is distrusted! We should consider the distrust arc when computing trust values. Figure 1.b depicts a more subtle version of the same problem. While checking the S-A-C-D path, the forward recursion does not consider that one of S's most trusted neighbors (B) distrusts C. GePuTTIS addresses this problem by reversing the recursion process.

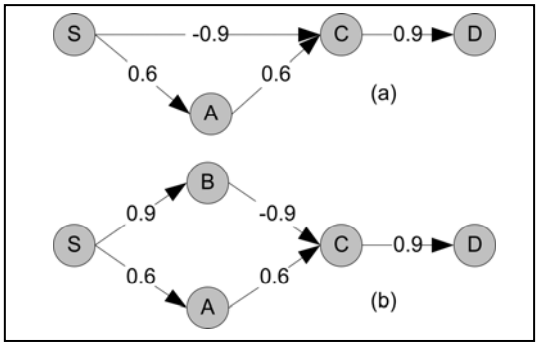

Figure 1. Distrust arcs problem. It is tempting to infer that D will be a useful advisor from the lower path, but C's judgment should be suspect in both cases because C is (part a) or should be (part b) distrusted by S.

Insufficient information problem

Trust inference algorithms that use weighted average over a series of advisors suffer in situations where sufficient information is not available. This is especially important when overestimated ratings (also known as false positive answers) are harmful. Consider figure 2. With weighted average inference, the value of non-existent arc S-D is computed below using the TidalTrust formula:

$$t_{SD} = \frac{t_{SA} \times t_{AD}}{t_{SA}} = \frac{0.1 \times 0.9}{0.1} = 0.9 \qquad (1)$$

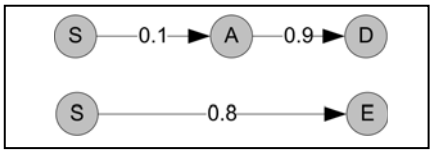

Figure 2. Insufficient data problem. Failing to consider the strength of the trust relationships leads to regarding D as more trusted than E.

The algorithm recommends S to trust D more than E. This is obviously irrational, because S himself knows E and trusts him, but a minimally-trusted neighbor A advises S that D is trustworthy. Recommendations from E are much more important than the advice of A. TidalTrust includes a max parameter that eliminates paths where some links express low trust; however, this eliminates some information while still treating all inferences equally.

GePuTTIS

GePuTTIS aims to be a general trust inference system that incorporates scope, a trust metric that allows expressions of distrust, and an algorithm that deals with distrust and low-support inferences. Our work so far is primarily around the algorithm, which we discuss in the next few sections. We then briefly discuss our ongoing work around the issues of scope and effective expression of trust values.

Inference Algorithm

Transitive trust inference algorithms consist of two phases. First, the data gathering, also known as path finding, that is usually done through a recursive procedure. Second, trust is inferred from the gathered data. The proposed algorithm is unique in both phases.

In the proposed algorithm, the path finding is done in reverse. That is, the source doesn't ask trusted parties how much they trust the destination node; instead, destination nodes ask neighbors on the path back to the source node about their trustworthiness for the source node[1].

The next step is to process the gathered data and infer the trust value for non-existent arcs. This phase is also divided into two parts. One part is calculating the total value of the advisors—that is, how uncertain is the algorithm in its estimate of the trust value for this arc. The other part is calculating the actual trust value. In effect, the algorithm scales the trust value by the uncertainty in order to produce conservative trust inferences.

To calculate the total value of advisors—our measure of support for the inference—we assume that each advisor removes some uncertainty from the total inference. An incremental formula is designed for this calculation. Assume that a new advisor is added that has value v for

[1] This assumes that nodes can know how much other nodes trust them. This would be problematic in a distributed system—a node A is unlikely to directly tell another node B that A distrusts B—but is reasonable in a system where nodes tell a trusted central authority about their beliefs in each other's trustworthiness.

the source node and the current value of advisors is V_{old} (0, if this is the first advisor). The updated value of advisors is calculated as follows:

$$V_{new} = V_{old} + (1 - V_{old}) \times v \quad (2)$$

An important assumption that should be considered when using equation 2 is that in the real world, advisors that are distrusted by the source node are ignored for the decision-making process. When an advisor is distrusted by the source node, it means that the advisee cannot count on the advisor's recommendations, so his advices should not be considered as negative nor positive advice. So v is a positive real value between 0 and 1. The second part of equation 2 means that the new advice is removing the uncertainty (the complement of the previous value of advisors) and is increasing the total value of advisors.

The second part (equation 3) is to calculate the weighted average recommendation. This uses the advisor's value for the source node (how much the advisee trusts the advisor) as weight and the advice (advisor's opinion) as the value.

$$A_{sd} = \frac{\sum_{i \in adj(d)} A_{si} \times A_{id}}{\sum_{i \in adj(d)} A_{si}} \quad (3)$$

After calculating the total value of advisors and the recommendation, the algorithms multiplies them to get the value of the non-existent trust arc, shown in equation 4.

$$T_{sd} = A_{sd} \times V_{sd} \quad (4)$$

Scoping mechanism

GePuTTIS includes a scoping mechanism based on allowing users to freely label expressions of trust with descriptions of the intended scope of the trust, interpreting the descriptions as a set of tags. Limited-vocabulary systems are unsuitable for general use, while using the description of the scope as a whole would be very difficult because it would be hard to match detailed scope expressions. Tagging systems strike a useful intermediate ground, although the problem of ensuring a common vocabulary remains. An alternative to the closed-vocabulary system is to use algorithms that suggest but do not enforce a vocabulary. Such algorithms have been shown to influence tagging behavior and encourage convergence on common terms (Sen et al. 2006).

The properties of the tagging system can be chosen based on the trust network's specific application. For the movies dataset which is used in our evaluation process, the set of genres can be considered as a simple closed vocabulary for scoping. Trust between users is computed on a per-genre basis. This allows us to explore the effect of trust scope on GePuTTIS' performance using a simulated dataset.

Evaluation

To evaluate the proposed trust inference system, we first discuss the performance of the algorithm in challenging situations; we then explore its performance using simulated trust values derived from the MovieLens dataset.

Dealing with challenging situations

In the case where sufficient information from reliable neighbors is available, GePuTTIS' inferred trust values are very close to TidalTrust's. Thus, we focus here on how GePuTTIS deals with challenging situations. As with TidalTrust, GePuTTIS only considers the trust values in the network—the goal is to have a trust system that works on trust values (ratings of other people) without directly incorporating opinions (ratings of items in the domain).

Dealing with distrust arcs: Reversing the recursion deals with this problem which rises because of the way that advice paths are discovered. Note that both types of recursion are used in real life. TidalTrust-style forward recursion is used when we ask our friends for recommendations. GePuTTIS' backward recursion is used when for example, an attorney tells you about satisfaction of his previous clients that you may know and you may trust him because of them.

Using backward recursion to compute how much S should trust D in figure 3 leads to calculating S's estimated trust for node C, then using the estimated trust value to temper his advice. C recommends D with a value of 0.9; however, C's value to S is calculated as follows:

$$\frac{0.9 \times (-0.9) + 0.6 \times 0.6}{0.6 + 0.9} = -0.3 \quad (5)$$

Because C's value is negative, no inference can be made. GePuTTIS also gracefully handles situations where conflicting advice should lead to small values of trust in an intermediate advisor. If the value of the arc that connects B to C changes to -0.2, then the total value of advisor will be 0.18 and the inferred trust from S to D will be 0.16. Again, this behavior seems logical. As the number or strength of trusted advisors increases, so does the inferred trust value.

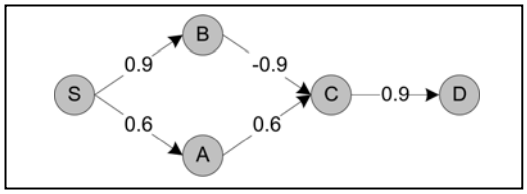

Figure 3. Revisiting the problem of negative trust.

Dealing with insufficient information: The example above shows how the algorithm deals with weak support for a trust inference by reducing inferred trust values for the arcs. Consider again figure 2. The inferred trust from S to D will be 0.09, which is far lower than the trust

expressed by S for E. This seems more logical and conservative than the inference from TidalTrust.
In both challenging situations, the proposed algorithm provides us with conservative and rational results. GePuTTIS is not designed to provide us the closest ratings to user's opinion, but to extract results that are not overestimated. False-positive recommendations that lead users to items they do not want, can cause users to not trust the system, and are often the most important errors in the e-commerce domain (Sarwar et al. 2000).

Evaluation using a simulated dataset

We compare the performance of TidalTrust and GePuTTIS using simulated trust ratings based on the ratings contained in the publicly available MovieLens dataset.[2] Simulated trust data are extracted from the similarity between users' ratings. To determine the trustworthiness of B for A, we use the mean absolute difference of their ratings on movies that are rated by both of them, scaled by the ratio of the number of movies that are rated by both to the total number of movies that are rated by A. We then choose the most trusted and most distrusted parties for the evaluation. Using these implicit trust arcs, other arcs have been guessed and predicted ratings are evaluated based on inference on implicit trust arcs. We use both TidalTrust and the proposed GePuTTIS algorithm.
We evaluated on two different splits. The first one uses 20% of users' ratings for testing and the other 80% for trust-extraction. In the second splitting, 10 ratings per user are chosen for testing, with all others used for trust-extraction. The results of the evaluation are presented in tables 1, 2 and 3. Each cell is the mean absolute error, normalized to be between 0 and 100, for the specified algorithm using specified settings. For each splitting four trust-extractions are considered to investigate how the number of explicitly represented trust arcs affects recommendation accuracy. Each column presents a run with a specified number of trust/distrust arcs. For example 5/0 means that 5 positive trust and 0 negative trust arcs are explicitly given for each user.
The first row of results in both tables is the average error of using IMDB average rating[3] as the predictor. The second row is the average error of the Slope One collaborative filtering algorithm presented in (Lamire and Maclachlan 2005), which we present as a baseline.

(trust/distrust) arcs	5/0	5/5	10/0	10/10
IMDB Average	20.44			
CF (Slope One)	18.88			
TidalTrust	15.82	15.81	15.48	15.46
GePuTTIS	15.19	15.11	14.88	14.93

Table 1. Evaluation using 80% - 20% splitting

[2] Available at http://www.grouplens.org/taxonomy/term/14. We were unable to get access to the FilmTrust dataset with explicit trust opinions.
[3] IMDB ratings are extracted from movies dataset available at http://had.co.nz/data/movies/.

(trust/distrust) arcs	5/0	5/5	10/0	10/10
IMDB Average	19.61			
CF (Slope One)	18.88			
TidalTrust	16.08	16.08	15.66	15.59
GePuTTIS	15.06	15.24	14.80	15.00

Table 2. Evaluation using 10 ratings per user splitting

The proposed system shows better performance than TidalTrust in every run, and more trust arcs led to more accurate recommendations. We were surprised that adding the negative trust arcs reduced GePuTTIS' performance. We speculate that this is because we used simulated trust values based on similarity of movie ratings. In this situation, the main advantage of GePuTTIS comes from its weighting of advice, rather than its ignoring of negative trust. Presumably, here the distrust arcs were still useful for helping to find neighbors who could make positive contributions to recommendations. We would like to study how actual expressions of distrust would affect the algorithm, if a dataset with explicit trust values were available.
Table 3 shows the effect of incorporating trust scopes. Genres are considered as movies' applicable trust-scopes and five or ten positive trust arcs for each genre are assigned to each user based on the similarity of users' ratings of that genre's movies.

	Without scopes	Using scopes
GePuTTIS-5	15.06	14.87
GePuTTIS-10	14.80	14.90

Table 3. Effect of using trust scopes

Using the scopes had a minimal effect on this simulated dataset. Again, we think this is likely to be related to the nature of the data: if you trust someone's recommendations about horror movies, their advice about action movies is likely to be at least somewhat useful. As the overlap between scopes decreases—for example, if the scopes are food and books—considering scope when inferring trust is likely to become more important.

Other aspects of GePuTTIS/future work

Our first goal with GePuTTIS is to evaluate it with actual trust arcs and in an actual social network. Using GePuTTIS in a real community can provide us more details about its performance. Similarity of interests was the best approximation we could make to trust, but especially in cases of distrust, we believe real expressions of trust will have very different meanings and effects on trust inference.

We would also like to refine how GePuTTIS thinks about users' trust ratings. In recommender systems, people use the same rating to mean different things—for instance, some people rate any movie they enjoy as five stars, while others reserve those ratings for only their most favorite

movies. Techniques such as z-score adjustment are often used to make users' ratings more comparable. It is an open question whether trust ratings should be scaled in similar ways, and whether that would lead to more useful trust inferences. One approach we are trying is to use reinforcement learning methods to interpret users' trust ratings; but it is hard to find appropriate parameters and there are many choices in applying reinforcement learning on trust networks. Likewise, it might be interesting to try to infer certainty values for a user's trust ratings, for example, by looking at how long one user has trusted another, by looking at the results of recommendations made through trusted advisors as in (O'Donovan and Smyth 2005), and by looking at changes in the trust network over time.

Conclusion

Improving social aspects of the web and designing better models of social concepts through creating better online interaction environments is one of the most important concerns of HCI researchers. Trust is an important commodity that shows promise as a computational tool for supporting online interaction. The GePuTTIS approach for inferring transitive trust in a social network provides a practical method for expressing and deriving trust between members within a community or network. It addresses the problems of distrust and weak support that can affect trust inference algorithms, and can be used in a wide range of applications and different types of recommender systems.
All in all the proposed trust inference system is different in application and approach from most of the previously suggested trust inference systems. GePuTTIS is intended to use a better model of the real world's trust inference process and extract more logical and more reliable results than previously suggested trust inference systems with lower rate of overestimated results that are considered harmful in many recommender systems.

Acknowledgements

We would like to thank Dr. Farhad Oroumchian & Sam Vakil for their comments.

References

Golbeck, J. 2005. Computing and Applying Trust in Web-based Social Networks. Ph.D. diss., Dept. of Computer Science, University of Maryland, College Park, Maryland.

Gray, E., Seigneur, J.M., Chen, Y., Jensen, C. 2003. Trust Propagation in Small Worlds. In *Proc. First Intl. Conf. on Trust Management, iTrust 03*, 239–254. Heraklion, Crete, Greece: Springer.

Guha, R., Kumar, R., Raghavan, P., Tomkins, A. 2004. Propagation of Trust and Distrust. In *Proc. WWW '04*, 403–412. New York: ACM Press.

Hill, W. C., Hollan, J. D., Wroblewski, D., McCandless, T. 1992. Edit Wear and Read Wear. In *Proc. CHI 92*, 3–9. California: ACM Press.

Jiang, Q., Reeves, D.S., Ning, P. 2004 Certificate Recommendations to Improve the Robustness of Web of Trust, in *Proc. ISC '04*, 292–303. California: Springer.

Jøsang, A., Hayward, R., Pope, S. 2006. Trust Network Analysis with Subjective Logic. In Proc. Australasian Computer Science Conference, 85–94. TAS, Australia: Australian Computer Society.

Jøsang, A., Ismail, R., Boyd, C. 2007. A Survey of Trust and Reputation Systems for Online Service Provision. *Decision Support Systems*, 43(2), 618–644.

Kamvar, S.D., Schlosser, M.T., Garcia-Molina, H. 2003. The Eigentrust Algorithm for Reputation Management in P2P Networks. In *Proc. WWW '03,* 640–651. Budapest, Hungary: ACM Press.

Lemire, D., Maclachlan, A., 2005. Slope One Predictors for Online Rating-Based Collaborative Filtering. In *Proc. SIAM Data Mining*, 21–23. California: MIT Press.

Levin, R., Aiken, A. 1998. Attack Resistant Trust Metrics for Public Key Certification In Proc. 7th USENIX Security Symposium, 229–242. Texas: USENIX Association.

Massa, P., Avesani, P. 2004. Trust-aware Collaborative Filtering for Recommender Systems. In *Proc. Federated International Conference On The Move to Meaningful Internet: CoopIS, DOA, ODBASE*, 492–508. Agia Napa, Cyprus: Springer.

McNee, S.M., Lam, S.K., Guetzlaff, C., Konstan, J.A., Riedl, J. 2003. Confidence Displays and Training in Recommender Systems. In *Proc. INTERACT '03*, 176–183. Zürich, Switzerland: IOS Press.

O'Donovan, J., Smyth, B., Evrim, V., McLeod, D., 2007. Extracting and Visualizing Trust Relationships from Online Auction Feedback Comments. In *Proc. IJCAI '07*, 2826–2831. Hyderabad, India, AAAI Press.

O'Donovan, J., Smyth, B. 2005. Trust No One: Evaluating Trust-based Filtering for Recommenders. In *Proc. IJCAI '05*, 1663–1665. Edinburgh, Scotland: Morgan Kaufmann.

Page, L., Brin, S., Motwani, R., Winograd, T. 1998. The PageRank Citation Ranking: Bringing Order to the Web. Technical Report. Stanford Digital Library Technologies.

Putnam, R. 2000. *Bowling Alone: The Collapse and Revival of American Community*: Simon & Schuster.

Sarwar, B., Karypis, G., Konstan, J., Riedl, J. 2000. Analysis of Recommendation Algorithms for E-commerce. In *Proc. EC '00*, 158–167, Minnesota: ACM Press.

Sen, S., Lam, S.K., Rashid, A., Cosley, D., Frankowski, D., Osterhouse, J., Harper, F.M., Riedl, J. 2006. Tagging, Communities, Vocabulary, Evolution. *In Proc. CSCW '06*, 181–190, Alberta, Canada: ACM Press.