Trust-Based Recommendation Based on Graph Similarity

Chung-Wei Hang and Munindar P. Singh

Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8206, USA
{chang,singh}@ncsu.edu

Abstract. Trust networks are directed weighted graphs whose nodes represent agents and edges represent trust between agents. This paper proposes a trust-based recommendation approach, which can recommend trustworthy agents to a requester in a trust network. We consider a good recommendation as one to an agent that the requester's trusted neighbors trust highly. We relate the recommendation problem to the graph similarity problem, and define the similarity measurement as a mutually reinforcing relation. By calculating the vertex similarity between the trust network and a structure graph (a path graph of length three), we can produce a recommendation based on similarity scores that reflect both the link structure and the trust values on the edges.

Key words: Trust, Graph Similarity, Recommendation

1 Introduction

Trust networks are directed weighted graphs whose nodes represent agents, edges represent trust relations, and weights represent trust values [1, 2]. An edge from node u to node v with trust value t means u trusts v to the extent of t. A similar concept is commonly seen in the real world. Examples include Facebook[1] where edges are the friendships between people; citation networks where nodes are papers and edges are citations; web graphs where nodes are webpages and edges are hyperlinks; FilmTrust [3] where edges are movie taste similarity between people; and, Epinions [4] and Advogato [5, 6], where the edges are trust relations.

There are two main challenges in trust networks: (a) *trust propagation*, and (b) *trust-based recommendation*. Trust propagation is about predicting the trustworthiness of nonadjacent agents by combining trust values through distinct indirect paths. To be more specific, trust propagation defines how trust values are aggregated and propagated through a trust network. It can help agents estimate a stranger's trustworthiness without assuming previous experience with the stranger. The other problem in trust networks is trust-based recommendation. Given a trust network and a target agent v, how to recommend a trustworthy

[1] http://www.facebook.com/

agent for v to interact with. Trust propagation is widely studied in the literature [1, 5, 7, 8, 6, 2, 9, 10], whereas trust-based recommendation has not drawn much attention from the research community.

A possible solution to trust-based recommendation is to apply trust propagation to estimate the trustworthiness of all agents that are not adjacent to the target v, and recommend the agents with high trust estimates. However, this solution is not promising because the complexity of the trust propagation grows quickly as the number of agents increases. Another possible solution is to recommend agents who share a fair number of common neighbors. For example, Facebook recommends friends based on the number of mutual friends between people. However, this approach fails to take the trust values (i.e., edge weights) into consideration.

Our approach aims to provide a trust-based recommendation approach, which recommends trustworthy relationships by considering not only the link structure (e.g., the number of common neighbors), but also the trust values associated with the links. Instead of considering one single potential neighbor separately, our trust-based recommendation processes the trust network around the target (i.e., the agent that requests a recommendation), thereby providing recommendations more efficiently. The idea behind our approach is based on graph similarity [11]. We show that by calculating vertex similarity between the trust network and a *structure graph*, the trust-based recommendation problem can be translated into a graph similarity problem, and the similarity scores can be viewed as a measurement of how many good connections (i.e., with high trust values) the agents shares with the target. Besides, instead of predicting how the trust network evolves from a *network*-level perspective, our approach departs from a *node*-level perspective, providing personalized recommendations, especially for the target.

Note that we can customize trust-based recommendation by using different structure graphs. In this paper, we study two basic structure graphs that facilitate making recommendation based on in-degree and friends of friends, respectively. However, our approach is not limited to these structure graphs. It can be extended to produce recommendations based on other criteria.

To summarize, our trust-based recommendation takes a trust network and a target agent (who requests recommendation in the trust network) as inputs, and outputs a list of trustworthy agents for the target to interact with as recommendation. This paper makes four key contributions. First, our approach provides personalized recommendations for a particular agent. Second, a recommendation produced is based not only on the network topology, but also on the trust values associated with the edges. Third, our approach is efficient because it considers only a small subgraph of the trust network, and processes potential candidates all at once. Fourth, our approach allows for customizing recommendation based on various criteria.

The rest of paper is organized as follows. Section 2 surveys the state of the art of the related research areas. Section 3 presents our approach by first introducing the graph similarity measurement used in our approach, and then demonstrating how the trust-based recommendation problem can be solved via graph similarity. Section 4 concludes this paper and identifies possible future directions.

2 Related Work

Here we categorize the literature into four areas: *graph similarity*, *link prediction*, *trust propagation*, and *recommender systems*.

Our trust-based recommendation approach is built on a graph similarity measurement. Graph similarity has been applied in various applications. For example, Melnik et al. [12] present a graph similarity approach, called *similarity flooding*, for database schema matching. Their approach takes two graphs as inputs, measures the vertex similarity between the inputs, and outputs a mapping—a subgraph consisting of similar nodes. This work differs from ours in many ways except both apply vertex similarity between two graphs. First, Melnik et al.'s input graphs have no weights, whereas our approach takes the edge weights into consideration. Second, their approach takes two graphs as input, and calculates the similarity between them. Our approach takes only one graph as input. Given that graph, our approach calculates the similarity between the graph and a *structure graph*, which reflects the features we care about in the recommendation. Third, Melnik et al.'s approach requires adjustment by humans, which ours does not.

Jeh and Widom [13] propose a domain-independent similarity measurement, *SimRank*. SimRank measures the similarity between objects. It follows the intuition that "two objects are similar if they are related to similar objects." Jeh and Widom first convert the graph to a *node-pair* graph, where each node represents a node-pair in the original graph. The node-pair (a, b) is connected to the node-pair (c, d) if a connects to c and b connects to d in the original graph. Then they calculate and propagate similarity score in the converted graph iteratively until convergence. Again, Jeh and Widom only consider graphs with no edge weights, whereas edge weights (i.e., trust values) play an important role in our approach.

Link prediction for large networks studies how to predict the edges that will be added in the future, given the current snapshot of a network. Liben-Nowell and Kleinberg [14] survey various link prediction methods from graph theory and social-network analysis. These methods measure the similarity between nodes with respect to the network topology, assign a weight to each pair of nodes, and generate a list sorted in decreasing order in terms of weights. Liben-Nowell and Kleinberg evaluate these link prediction methods in five collaboration networks where edges connect authors who coauthor papers. They indicate that the link prediction approaches can provide a network evolution model learned from the observed data. This learned network evolution model can infer how the network is going to evolve based on the network features. Unfortunately, Liben-Nowell and Kleinberg's approach only considers undirected graphs without edge weights.

Kunegis and Lommatzsch [15] propose a general link prediction approach, which applies machine learning techniques to reduce the learning parameters for link prediction, and then uses a curve-fitting method to estimate the parameters. Their approach can be applied to undirected, weighted, or bipartite graphs. Kunegis and Lommatzsch evaluate the approach on web graphs, trust networks, social networks, citation networks, and collaboration networks. Note that, in general, the link prediction methods provide *network*-level prediction, whereas our approach focuses on *node*-level recommendation. In other words, link prediction recommends links for the whole network, but our approach recommends trustworthy others for a particular node.

Trust propagation provides an alternative solution to trust-based recommendation from a node-level perspective. Recommendations can be made by first estimating the trustworthiness of all nonadjacent agents, and then listing the agents with high trust estimates. Hang et al. [10] model trust as a binary event. They define three operators for concatenating trust along a path, aggregating trust from distinct paths from the same witness, and selecting the most trustworthy path among all witnesses, respectively. Advogato [5] adopts a network flow algorithm where the flow capacity of edges is determined by the depth along the path. Appleseed [6] applies spreading activation, where *trust energy* is spread across the trust network. The energy is divided when the agent has more than one successor. All these trust propagation methods provide recommendation for a particular agent. However, they are not computationally efficient because the complexity grows quickly as the number of agents increases. Our approach does not treat each of the nonadjacent agents separately. Instead, it processes all agents at the same time, yielding better performance.

Now we discuss some related work of recommender systems. In general, recommender systems suggest *items* to *users*. There are two main categories of recommender systems: *content-based* and *collaborative filtering* systems [16]. Content-based approaches produce recommendations based on the similarity between items. Collaborative filtering approaches recommend the items chosen by the users with similar tastes. Our approach is closer to a collaborative filtering approach because some of the collaborative filtering approaches construct a trust network where nodes are users and edges represent the similarity between users' taste. For example, FilmTrust [3] is a social network where edges represent the similarity of movie taste. Ben-Shimon et al. [17] propose a recommendation approach that is quite similar to ours. They construct a *personal* social network containing friends of friends (up to six levels) of a user who needs recommendation. Ben-Shimon et al. then find the sum of all the ratings of a particular item, discounted by the distance from the rater to the user. If the sum is high, the item is recommended. There are no items involved in our case. Thus, rather than computing the sum of all ratings, our approach considers the link structure and the trust values on the edges. Fouss et al. [18] present a recommender system based on a similarity measurement between the nodes of a directed weighted graph. They compute similarity based on a Markov-chain random walk model, which assigns a transition probability to each edge. The distance required for a random walker to travel from one node to another defines the similarity between these two nodes. Our approach is different from Fouss et al.'s approach in two ways. First, our approach considers directed graphs rather than undirected ones. Second, we use the similarity measurement defined by a *mutually reinforcing relation* rather than the Markov-chain random walk model.

3 Approach

Now we introduce our approach. In Section 3.1, we briefly overview the graph similarity measurement applied in our approach. Section 3.1 also shows two applications of the graph similarity by customizing different structure graphs. In Section 3.2, we first define a trust network. Next we customize the graph similarity measurement by devising a structure graph that satisfies our claim for suggesting recommendations in a trust network. Then we show how trust values are considered, and how they affect the recommendations produced. We formalize our approach at the end.

3.1 Background: Vertex Similarity between Graphs

Blondel et al. [11] propose a vertex similarity measurement between graphs. Given two directed graph G_A with n_A vertices, and G_B with n_B vertices, a similarity matrix $\mathbf{S}$ is a $n_A \times n_B$ matrix where s_{ij} is the similarity score between node i in G_A and node j in G_B. $\mathbf{S}$ can be calculated by a convergent iterative process:

$$\mathbf{S}_{k+1} = \frac{B\mathbf{S}_k A^T + B^T \mathbf{S}_k A}{\|B\mathbf{S}_k A^T + B^T \mathbf{S}_k A\|_F}, \tag{1}$$

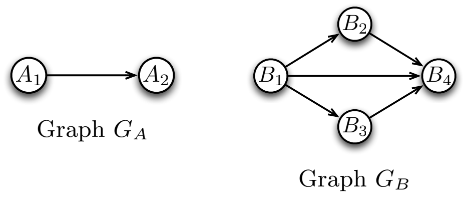
where A and B are the adjacency matrices of G_A and G_B, respectively, $\mathbf{S}_0$ has all entries equal to 1, and $\|.\|_F$ is the square root of the sum of the squares of all entries. The denominator normalizes $\mathbf{S}_{k+1}$ to $[0, 1]$. The limit of this convergent process is $\mathbf{S}$. The convergence can be determined by

$$\|\mathbf{S}_{k+1} - \mathbf{S}_k\|_F < \epsilon, \tag{2}$$

where ϵ is the error tolerance.

For example, Figure 1 shows the similarity matrix between two graphs: G_A and G_B. G_A contains two vertices: A_1 has out-degree of one, and A_2 has in-degree of one. After measuring the vertex similarity with G_B, one can observe that B_1, which has the largest out-degree, is the most similar vertex to A_1. B_4, which has the largest in-degree, has the highest similarity score to A_2. Notice that the greater the out-degree a vertex has, the higher its similarity score to A_1. An analogous observation applies to the in-degree and A_2. We can conclude that by comparing the similarity score to G_A, we can find the vertex that connects to the most others, and is connected by the most others.

The idea behind the similarity measurement is the *mutually reinforcing relation*, which is widely applied in web search [19, 20], and reputation management in peer-to-peer systems [21]. To illustrate the mutually reinforcing relation, we take G_A and G_B in Figure 1 as an example. For each vertex B_i in G_B, we associate two similarity scores, say s_{i1} (for A_1) and s_{i2} (for A_2), each of which



	A_1	A_2
B_1	0.5774	0
B_2	0.2887	0.2887
B_3	0.2887	0.2887
B_4	0	0.5774

Fig. 1. Example of two graphs, G_A and G_B, and their similarity matrix. The most similar vertex to A_1 is B_1, which has the largest out-degree; the most similar vertex to A_2 is B_4, which has the largest in-degree.

corresponds to the similarity between one vertex in G_A and B_i. Both scores are initialized to one. Then the scores are updated according to the mutually reinforcing relation:

$$\begin{cases} s_{i1} = \sum_{j:(i,j)\in E_B} s_{j2} \\ s_{i2} = \sum_{j:(j,i)\in E_B} s_{j1} \end{cases} \tag{3}$$

This mutually reinforcing relation says a vertex is similar to A_1 if it connects to many vertices that are similar to A_2, whereas a vertex is similar to A_2 if it is connected by many vertices that are similar to A_1. The update process is iterated. The scores s_{i1} and s_{i2} mutually reinforce each other. Blondel et al. show that this update process converges to a state, which corresponds to the similarity scores between A_1 and A_2, and B_i.

Now let us extend the similarity scores to all vertices in G_B. Suppose $\mathbf{s}_1$ and $\mathbf{s}_2$ are the similarity scores to A_1 and A_2, respectively, for all B_i in G_B

$$\mathbf{S}_{k+1} = \begin{bmatrix} \mathbf{s}_1 \\ \mathbf{s}_2 \end{bmatrix}_{k+1} = \begin{bmatrix} 0 & B \\ B^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{s}_1 \\ \mathbf{s}_2 \end{bmatrix}_k = \mathbf{S}_k, \tag{4}$$

where B is the adjacency matrix of G_B and $k = 0, 1, \ldots$. Blondel et al. further extend G_A to an arbitrary graph and simplify Equation 4 to Equation 1, where A is the adjacency matrix of G_A.

By using different G_A (called the *structure graph*), we can apply the vertex similarity to solve different problems and applications. Blondel et al. show that the web search algorithm, *HITS* [20], which searches for webpages based on a query, is an application of vertex similarity. HITS ranks webpages based on an *authority score* and a *hit score*. A webpage is a good authority if there are many hits that link to it. In contrast, a good hit is a webpage that points to many good authorities. The HITS algorithm is a special case that compares the vertex similarity between the cyberspace and G_A in Figure 1. Blondel et al. point out another application, synonym extraction. They construct a directed graph from a dictionary, where a node represents a word, and an edge from u to v means u is used in the definition of v. They first create a subgraph of the dictionary graph by extracting all words connecting to a word x or connected by x. Then they calculate the similarity score with the graph G_s shown in Figure 2. The vertices with higher similarity score to A_2 are chosen as synonyms of x. The similarity score to A_2 of a word y indicates how many common words occur in x and y's definitions, and how many definitions use both x and y. Thus, the similarity score to A_2 reflects the definition of a synonym.

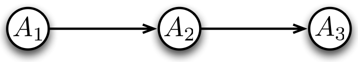


Fig. 2. Structure graph G_S.

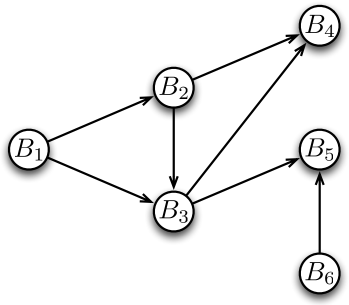
3.2 Trust-based Recommendation Based on Graph Similarity

A trust network is a graph where nodes represent agents and edges represent trust relations [1, 2]. A trust relation from agent u to agent v indicates how much trust u places in v. Thus, an edge in a trust network is associated with a trust value as its weight. Depending on the trust models, a trust value can be a single scalar, a Beta distribution, or follow another representation. The trust relations can be obtained from a direct interaction or from a referral via trust propagation [10]. For example, a social network such as Facebook is a trust network where all edges are modeled have the same trust values. Here we only consider a trust value as a single scalar. Other trust representations can be translated into a scalar, for example, a probability.

Definition 1. *A trust network TN is a directed weighted graph $TN(V, E)$, where V is a finite set of agents $\{v_1, \ldots, v_n\}$, and E is a set of trust relations $\{e_1, \ldots, e_m\}$.*

Consider the recommendation problem in trust networks: given a snapshot of a trust network, how can we recommend (i.e., predict) a trustworthy agent to an agent v? By intuition, we claim a good recommendation for v is an agent connected by many of v's neighbors. Let us start with a simple case where all edges have the same trust values 1 (i.e., no weights). For example, Figure 3 (left) shows a trust network TN_B, which contains the neighbors of the neighbors of agent B_1. Among all the agents except B_1's neighbors B_2 and B_3, B_4 is the most possible candidate, because it is connected by two neighbors of B_1. Consider the structure graph G_S in Figure 2, which illustrates our claim of producing good recommendations: a friend (A_3) of A_1's friend (A_2) is probably A_1's friend (i.e., a good recommendation for v is an agent connected by many of v's neighbors). Figure 3 (right) shows the similarity matrix between G_S and TN_B. The similarity score between A_3 and vertices indicates how the link structure of the vertices is similar to the link structure of A_3.

Now we consider the general case where each of the edges in TN_B is associated with a trust value. Instead of using the adjacency matrix, we define the *adjacency matrix with trust*, which is similar to the adjacency matrix for *multi-graphs* (permitted to have multiple edges between the same end nodes), except the entries can be non-integers. The entries in the adjacency matrix with trust



	A_1	A_2	A_3
B_1	0.4258	0.0901	0
B_2	0.3107	0.3562	0.0458
B_3	0.0828	0.4809	0.2270
B_4	0	0.1300	0.4258
B_5	0	0.0329	0.2940
B_6	0.0167	0.0971	0

Fig. 3. Example of a trust network with no edge weights, and its similarity matrix with the structure graph G_S in Figure 2. Among friends of B_1's friends (i.e., B_4, B_5, and B_6), B_4 is the best recommendation with the highest similarity score with A_3.

represent the trust values associated to the corresponding edges. One can regard a trust relation from u to v with a high trust value as there exist many edges from u to v.

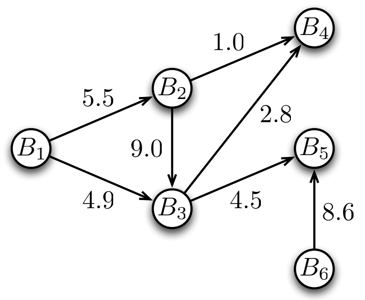
Definition 2. *An adjacency matrix with trust A of a trust network $TN(V, E)$ of n agents is an $n \times n$ matrix where the entry a_{ij} is the trust value v_i places in v_j.*

Figure 4 shows an example of a trust network TN'_B, and the similarity matrix between the structure graph G_S in Figure 2 and TN'_B. TN'_B shares the same topology as TN_B in Figure 3, except TN'_B has trust values as its edge weights (rather than 1). Unlike the result in Figure 3, although B_5 has fewer connections with B_1's neighbors than B_4, B_5 has the highest similarity score because the trust value of its only connection is much stronger than the trust values of B_4's connections. Note that B_3 (not considered as a recommendation because it is already a neighbor of B_1) also has a high similarity score because it is connected by B_2 (B_1's neighbor) with a high trust value.

We formalize our trust-based recommendation approach. Given a trust network $TN(V, E)$, to find recommendations for agent v, we construct a subgraph $TN'(V', E')$ where $V' \subset V$ contains v, all the neighbors of v, and the neighbors of v's neighbors, and $E' \subset E$ are all trust relations between any two of v, v's neighbors, or the neighbors of v's neighbors. Then the similarity matrix between the structure graph G_S (Figure 2) and TN' is calculated. The nodes that are not neighbors of v and have high similarity scores to A_3 are recommended. The reason of taking the subgraph is because if the whole TN is considered, the result will not be a recommendation for the agent v. Instead, the agents with the high similarity scores are just similar to A_3 in G_S, i.e., these agents are connected by many other agents that connected by many others.

We can summarize the main steps of our approach as follows:

1. Given an agent v in a trust network $TN(V, E)$ ($v \in V$).



	A_1	A_2	A_3
B_1	0.3950	0.1307	0
B_2	0.4044	0.3444	0.0669
B_3	0.0089	0.4802	0.3478
B_4	0	0.0271	0.1571
B_5	0	0.0045	0.3550
B_6	0.0036	0.1926	0

Fig. 4. Example of a trust network where edge weights represent trust values, and its similarity matrix with the structure graph G_S in Figure 2. Unlike the result in Figure 3, although B_4 is connected by B_1's both friends, B_5 is the best recommendation because of its strong connection with B_3.

2. Construct $TN'(V', E')$, where $V' \subset V$ contains (a) v, (b) v's neighbors, and (c) the neighbors of v's neighbors; $E' \subset E$ contains all trust relations between $(u', v') \in V'$.
3. Calculate the similarity matrix $\mathbf{S}$ between G_S (Figure 2) and TN' by Equation 1.
4. Recommend the vertices that are not neighbors of v with high similarity scores to A_3 in G_S.

4 Conclusion

In this paper, we present a trust-based recommendation approach, which provides recommendations to a requester in a trust network. The approach is built on a vertex similarity measurement between graphs. The similarity measurement is defined by a mutual reinforcing relation. We show that by calculating the similarity between the trust network and a structure graph (a path graph of length three), the similarity score can be viewed as a indicator that the agent is strongly connected by the strong neighbors of the requester.

To further validate our approach, a possible future direction is to evaluate on real datasets, for example, FilmTrust [3], Epinions [4], and Advogato [5, 6]. There are three tentative experiment settings. First, we can use cross-validation by removing some of the edges of the requester from the trust network, applying the trust-based recommendation, and comparing the recommendation list with the removed edge list ordered by their trust values. Second, we can trace the evolution of a trust network over time. We can compute the recommendation list based on the past snapshot of the network, and then compare it with the current snapshot to see if the trust network does evolve as predicted. Third, we can compare the recommendation list with the agent list ordered by the estimated trust values calculated by trust propagation [10].

Another direction is to study how our approach can be extended to provide different recommendation. For example, Hang et al. [22] design a trust-based service composition model for estimating trustworthiness of the subservices underlying a composition. However, their model fails to provide a mechanism for selecting the subservices—recommending compositions. Based on our approach, we can construct a structure graph that satisfies their scenario.

Acknowledgment

This work is supported by the U.S. Army Research Office (ARO) under grant W911NF-08-1-0105 managed by NCSU Secure Open Systems Initiative (SOSI).

References

1. Yu, B., Singh, M.P.: Distributed reputation management for electronic commerce. Computational Intelligence **18**(4) (November 2002) 535–549
2. Wang, Y., Singh, M.P.: Trust representation and aggregation in a distributed agent system. In: Proceedings of the 21st National Conference on Artificial Intelligence (AAAI), Menlo Park, AAAI Press (2006) 1425–1430
3. Kuter, U., Golbeck, J.: Sunny: A new algorithm for trust inference in social networks using probabilistic confidence models. In: Proceedings of the 22st National Conference on Artificial Intelligence (AAAI), Menlo Park, AAAI Press (2007) 1377–1382
4. Guha, R., Kumar, R., Raghavan, P., Tomkins, A.: Propagation of trust and distrust. In: WWW '04: Proceedings of the 13th international conference on World Wide Web, ACM Press (2004) 403–412
5. Levien, R.: Attack Resistant Trust Metrics. PhD thesis, UC Berkeley (2003)
6. Ziegler, C.N., Lausen, G.: Spreading activation models for trust propagation. In: EEE '04: Proceedings of the 2004 IEEE International Conference on e-Technology, e-Commerce and e-Service, Washington, DC, USA, IEEE Computer Society (2004) 83–97
7. Richardson, M., Agrawal, R., Domingos, P.: Trust management for the semantic Web. In: The Semantic Web: Proceedings of the 2nd International Semantic Web Conference (ISWC). Volume 2870 of LNCS., Springer (2003) 351–368
8. Gray, E., , Seigneur, J.M., Chen, Y., Jensen, C.: Trust propagation in small worlds. Lecture Notes in Computer Science **2692** (2003) 239–254
9. Quercia, D., Hailes, S., Capra, L.: Lightweight distributed trust propagation. In: Seventh IEEE International Conference on Data Mining (ICDM 2007). (2007) 282–291
10. Hang, C.W., Wang, Y., Singh, M.P.: Operators for propagating trust and their evaluation in social networks. In: Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS), Columbia, SC, IFAAMAS (2009) 1025–1032
11. Blondel, V.D., Gajardo, A., Heymans, M., Senellart, P., Dooren, P.V.: A measure of similarity between graph vertices: Applications to synonym extraction and web searching. SIAM Review **46**(4) (2004) 647–666
12. Melnik, S., Garcia-Molina, H., Rahm, E.: Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In: Proceedings of the 18th International Conference on Data Engineering, Washington, DC, USA, IEEE Computer Society (2002)

13. Jeh, G., Widom, J.: SimRank: a measure of structural-context similarity. In: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, New York, NY, USA, ACM Press (2002) 538–543
14. Liben-Nowell, D., Kleinberg, J.: The link-prediction problem for social networks. Journal of the American Society for Information Science and Technology **58**(7) (May 2007) 1019–1031
15. Kunegis, J., Lommatzsch, A.: Learning spectral graph transformations for link prediction. In: Proceedings of the 26th Annual International Conference on Machine Learning, New York, NY, USA, ACM Press (2009) 561–568
16. Shani, G., Chickering, M., Meek, C.: Mining recommendations from the web. In: Proceedings of the ACM conference on Recommender systems, New York, NY, USA, ACM Press (2008) 35–42
17. Ben-Shimon, D., Tsikinovsky, A., Rokach, L., Meisels, A., Shani, G., Naamani, L.: Recommender system from personal social networks. In: Proceedings of the 5th Atlantic Web Intelligence Conference, Springer Berlin / Heidelberg (2007) 47–55
18. Fouss, F., Pirotte, A., Renders, J.M., Saerens, M.: Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. IEEE Transactions on Knowledge and Data Engineering **19**(3) (2007) 355–369
19. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. Computer Networks and ISDN Systems **30**(1–7) (1998) 107–117
20. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. Journal of the ACM **46**(5) (1999) 604–632
21. Kamvar, S.D., Schlosser, M.T., Garcia-Molina, H.: The EigenTrust algorithm for reputation management in p2p networks. In: WWW '03: Proceedings of the 12th international conference on World Wide Web, ACM Press (2003) 640–651
22. Hang, C.W., Singh, M.P.: Trustworthy service selection and composition. Technical Report 18, North Carolina State University (July 2009)