Predicting Trust and Distrust in Social Networks

Thomas DuBois, Jennifer Golbeck, and Aravind Srinivasan
University of Maryland
College Park, MD 20742

***Abstract*—As user-generated content and interactions have overtaken the web as the default mode of use, questions of whom and what to trust have become increasingly important. Fortunately, online social networks and social media have made it easy for users to indicate whom they trust and whom they do not. However, this does not solve the problem since each user is only likely to know a tiny fraction of other users; we must have methods for inferring trust - and distrust - between users who do not know one another. In this paper, we present a new method for computing both trust and distrust (i.e., positive and negative trust). We do this by combining an inference algorithm that relies on a probabilistic interpretation of trust based on random graphs with a modified spring-embedding algorithm. Our algorithm correctly classifies hidden trust edges as positive or negative with high accuracy. These results are useful in a wide range of social web applications where trust is important to user behavior and satisfaction.**

I. Introduction

There are two billion people connected to the internet [1], and user-generated content is created and consumed at impressive rates. YouTube reports 24 hours of new video uploaded to their site every minute, and 2 billion videos watched every day. Facebook has over 600 million users who upload 2.5 billion photos per month, plus status updates, comments, videos, questions, and discussion posts. Twitter has over 200 million users creating over 90 million new tweets a day. The numbers continue for review sites, blogs and blog comments, more specialized social networks, and so on.

With so much user interaction and content created, the question of whom and what to trust has become an increasingly important challenge on the web. A user is likely to encounter dozens if not hundreds of pieces of user-generated content each day, and some of it will need to be evaluated for trustworthiness. Trust information can help a user make decisions, sort and filter information, receive recommendations, and develop a context within a community with respect to whom to trust and why. Fortunately, the rise of social networking on the web has allowed people to indicate whom they trust and distrust, creating links in the network (or edges in the graph to use graph-theory terminology). We can algorithmically use that information to make suggestions to other users about whom they in turn should trust.

In these contexts knowing whom to trust is important, however knowing whom to distrust is equally, if not more, useful. Unfortunately distrust is much trickier to compute in a satisfying way. While intuition and experimental evidence indicates that trust is somewhat transitive (if Alice trusts Bob and Bob trusts Chuck, there is a good chance that Alice will trust Chuck, too), distrust is certainly not transitive. If Alice distrusts Bob and Bob distrusts Chuck, Chuck may be closer to Alice than Bob is, or he may be even further away. Thus, when we try to propagate distrust through a network, questions about transitivity and how to deal with conflicting information abound.

While there has been some very nice work on distrust (e.g. [2]), there is a significant gap in the trust inference literature. Distrust information can be useful and plentiful but there are relatively few algorithms to compute it. For example, explicit distrust can distinguish between two factions who do not trust each other because of a lack of knowledge from those that are antagonistic. It can also expose subtleties in a trust network which are inexpressible with positive trust alone.

We contribute to this area a new algorithm for effectively predicting trust and distrust in web-based social networks. We combine a path-probability trust inference algorithm [3] with a novel technique using spring-embedding to infer network distance. We compute these two metrics for every pair of nodes in the network and train a classifier based upon the resulting two-dimensional data points. For each quantized connectivity probability estimate we find an embedding distance which minimizes the larger of the misclassified positive edges and the misclassified negative edges.

We evaluate the performance of our algorithm on three real-world datasets that include trust and distrust ratings from users. All of these datasets have edges which are either positive or negative (+1 or -1). Our algorithm can also be applied to networks with a scale of trust/distrust values, however we could not find a large dataset with both trust and distrust information in a finer level of detail. With each network, we apply our algorithm to the edge sign prediction problem. In this problem, some number of edges are chosen uniformly at random and hidden. Then we train our classifier on the remaining network, and use it to guess the signs of the hidden edges.

We find that when only a few edges are removed (which approximates guessing the trust between two nodes who have not rated each other) we consistently classify most of the positive and most of the negative edges correctly (81% for the Wikipedia dataset up to 89% for Epinions). This is comparable to the best existing methods which use substantially different techniques [2], [4]. We go a step further than existing works by exploring the result of removing many edges at once. We find that the accuracy rates do not diminish significantly until at least half of the edges have been removed. This implies not only that our algorithm captures inherent properties, but also that there is a high level of redundancy in the networks.

Not only can edge signs be predicted from the surrounding network, but they can be predicted from a random edge-induced subgraph of the surrounding network.

II. Related Work

There are many trust inference algorithms that take advantage of pair-wise trust values and the structure of a social network. All of these algorithms rely upon some notion of the transitivity of trust. In very small or dense networks most users may be close to each other and it is easy to argue that they should extend some trust to those their friends trust. However to be useful in large, sparse networks this transitivity must hold over paths longer than two or even three edges. TrustDavis [5] captures this transitivity economically. One user's direct trust in another takes the form of an insurance contract - for some fee c they will guarantee a trusted third party's debt of up to x. In this network the inferred trust between two parties for a debt x is the lowest cost network-flow with capacity x. Algorithms without this economic context must make the assumption that trust is somewhat transitive even over long distances. Finding the right balance where trust spreads far enough to be applicable for most pairs of users but not so far that it looses its effectiveness is part of these algortihms' parameter tuning.

Such algorithms include Advogato [6], Appleseed [7], Sunny [8], and Moletrust [9]. These algorithms use trust that is assigned on a fixed scale (e.g. 1-10). Other algorithms treat direct trust as a probability, including [3], [10]–[12]. The difficulty of generating these probabilities, using influence as a proxy for trust, was addressed in [13]. In our research, we work with probabilities that are given *a priori*, but those derived from other methods could also be used in our algorithms.

The results of trust inference have a wide range of applications. Recommender systems are a common one, where trust values are used in place of traditional user similarity measures to compute recommendations (e.g. [14]–[16]). Galland et al. present a technique for using trust to estimate the *truth* of information that is presented [17], which in turn has applications for assessing information quality, particularly on the Semantic Web. More specific applications of that idea include using trust for semantic web service composition [18].

Often these recommendation algorithms require, as an intermediate step, finding clusters of people who are more tightly connected to each other than to the remainder of the population [19], [20]. Trust recommendations are just one example application where clustering is useful; the art of finding useful sets of clusters has been well studied on a wide range of applications. Social tagging [21], web browsing on topic clusters [22], search classification [23], and music genre identification [24] are just a few examples that use clustering to improve performance and usability.

In some cases there is some (unknown) "ground-truth" clustering inherent in the data which we want to find, and the algorithms attempt to find a clustering that is "close" to the true one [25], [26]. Often, though, there is no reason to believe that the data has inherently correct clusters, and the goal becomes simply to produce a clustering which works well in practice for a particular application.

When each data point to be clustered consists of a vector of numerical values, one common technique is to choose a distance function between the elements (Euclidean, L1-norm, etc.) and look for clusters which minimize some optimization function. Examples of these algorithms include k-means [27] (which minimizes the mean squared-distance of elements from their cluster centers), and k-centers [28] (which minimizes the maximum distance from any point to the center of a cluster). Typically approximation algorithms, which find solutions close to optimal, are used because it is impractical to compute the optimal clustering for these problems. For a more extensive overview of various clustering algorithms, see [29].

Guha et al. [2] give one of the earliest studies that addresses both trust and distrust propagation in an algorithmic way. They treat trust propagation as a repeating sequence of matrix operations combining aspects of direct propagation, co-citation, and backwards propagation, and they consider both single-step and every-step propagation of distrust. They run a large number of trials with different sets of parameters to validate their approach using the edge sign prediction problem on the Epinions dataset. Their best results achieve 85% accuracy taken over an equal number of positive and negative hidden edges.

More recent work on edge sign prediction by Leskovec, Huttenlocher, and Kleinberg [4] is a direct predecessor of our work. They examine the same three networks as we do with a much more localized view. To predict the sign of an edge they look at the positive and negative edge counts of its endpoints, plus the number and type of triangles containing this edge. These local factors form a high dimensional space on which they perform standard machine-learning techniques to determine how to predict the sign of unknown edges. From a theoretical perspective they interpret their results through Heider's balance theory [30], which states that unbalanced triads (those with an odd number of negative edges) are unstable. Experimentally they show good edge prediction results for all three datasets (accuracy rates between 80-90% over all edges), with better results on edges with a higher embeddedness - those which are a part of a greater number of triangles.

III. Dataset Descriptions

We used three major social network datasets to test our methods. All are provided as part of the Stanford Large Network Dataset Collection[1]. The networks we use have both positive (trust) and negative (distrust) edges. These edges are unweighted, though our methods could easily support weighted trust and distrust.

- Wikipedia moderator elections - Wikipedia, the popular online encyclopedia created by users, has a set of elected moderators who monitor the site for quality and controversy and who help maintain it. These moderators receive

[1] http://snap.stanford.edu/data/

extra administrative privileges, and thus must be trusted by the community. When a user requests admin access, a public discussion page is set up for users to discuss and vote on whether to admit the moderator. Positive and negative votes are counted as positive and negative trust ratings. Note that in this network, if a user is not ultimately voted in, they will not appear in the graph. Thus, positive trust ratings (or positive votes) will be more common in the graph. The data was pulled from the discussion pages in January 2008 [4], [31]. It contains just over 7,000 nodes and 100,000 edges.

- Slashdot - This is a technology news site where users can rate each other as friend or foe. We treat those as positive and negative trust ratings. The dataset contains over 77,000 nodes and just under 900,000 edges. Use used the version released in February 2009 [32]
- Epinions - This is product review site where users choose whether to trust or not trust one another based on their ratings and reviews of products. The network has over 75,000 nodes and 500,000 edges. The dataset was collected and released in 2003 [33].

Because the most efficient versions of our algorithms use undirected graphs, we must consider how to adapt them for use on directed datasets. The Wikipedia election data is topologically ordered by when the election takes place, so the graph is anti-symmetric. Therefore we can simply treat all edges as undirected without affecting the methodology. The Slashdot and Epinions networks have a large fraction of edges where each endpoint rates the other. For our first set of experiments, those most similar to our predecessors, our algorithms treat the network as an undirected multigraph. This means that if there are edges in both directions between two nodes, it is possible that exactly one of them is hidden. In this case we still check our result against only the edge going in the direction which was hidden. If there were a lot of disagreeing pairs of edges, this approach would be at a dissadvantage. For the second set we make the graphs undirected by taking paired edges and averaging them into a single undirected edge.

IV. Algorithm and Methodology

In prior work, we develop a method for computing trust based on path probability in random graphs [3]. For every pair of users (u, v), we place an edge between them with some probability that depends on the direct trust value between them, denoted by $t_{u,v}$. We then infer trust between two people from the probability that they are connected in the resulting graphs. Formally we choose a reversible mapping f from trust value to probabilities, and then construct a random graph G in which each edge (u, v) exists independently with probability $f(t_{u,v})$. This graph gives inferred trust values, $T_{u,v}$, such that $f(T_{u,v})$ equals the probability that there is a path from u to v in the random graph. In addition to having an intuitive appeal, we find this approach to work well in practice.

Distrust, however, is more complex. While trust can be considered transitive, distrust is not. Additionally with only positive trust, there are no inconsistencies in the data – paths can differ in strengh, but they are all additive. When we incorporate distrust, there can be paths which disagree as in Figure 1. We propose using a modified layout algorithm to find a low-dimensional embedding of the graph which tries to reconcile the conflicting information and transitivity. It is inspired by spring embedding graph layout algorithms [34]. This type of algorithm, which until now has not been used for trust inference, simulates the physics of springs in a 2D or higher dimensional space. Edges between nodes are treated as springs that pull nodes together, but reasonable space is maintained between nodes by making them repel one another. Nodes are randomly laid out in an initial configuration, and the system is simulated until a stable equilibrium is reached or some short-circuit condition happens (maximum iterations, changes per timestep below a threshold, etc).

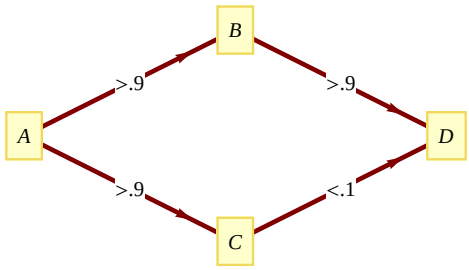

Fig. 1. A strongly trusts B and C. B and C disagree strongly on whether or not to trust D. It is not clear how much A should trust D.

Our first attempt to incorporate distrust involved resolving conflicting trust/distrust information through a nonlinear optimization. We would assume that all users' trust estimates are noisy, and we want to find the true ones. In this model, positive trust corresponds to edge probabilities, while negative trust corresponds to upper bounds on path probabilities. We then apply a cost function for each edge of the deviation between the "true" value and the "measured" value. We would then find a globally minimal cost solution which does not have any conflicting trust/distrust information and infer trust from it. Unfortunately this technique does not scale sufficiently well, and is therefore not suitable on large datasets.

This led us to develop a spring-embedding algorithm which we use in conjunction with our path probability technique to infer trust. First we compute path probabilities using only positive edges. Independently, we use an iterative spring embedding algorithm - where positive edges attract and negatives repel - to resolve competing trust/distrust information. Note that in the face of positive trust only, this results in all nodes very close to each other. A spring-embedding algorithm implicitly has the transitivity and conflict resolution properties we desire as well as the scalability necessary to handle very large datasets.

We modify the spring-emgedding layout algorithm to adapt it to our trust context. Instead of having all nodes repel, we add a repelling force only between nodes connected with a negative edge. Transitivity applies because two nodes with a shared friend are both pulled toward that friend. If they share

two friends who are co-located, they are pulled with twice as much force. If they have a shared enemy, they are both pushed away (which may or may not move them in the same direction). If one is friends with an enemy of the other, the forces will push them in different locations. This modified spring-embedding algorithm also deals well with conflicting information. If Node A has two friends who disagree about Node B, the friends will be pushed apart, and Node A will be partially pulled toward and partially pushed away from Node B.

One potential drawback is that two nodes may be placed close together by chance though they have little trust between them. This is why spring embedding alone is not enough - we need to consider path probabilities as well. We can independently compute path probabilities and spring embedding distances for our entire graph. For each edge or potential edge, we record the path probability between its endpoints as well as their embedded distance. Thus each edge corresponds to a two dimensional vector whose position indicates the amount of trust between its endpoints.

To assess our algorithm's quality, we use it to solve the edge sign prediction problem. For each of our three datasets, we remove a substantial number of edges (500 in Wikipedia and Slashdot, 1000 in Epinions) chosen uniformly at random. The removed edges make up the testing set and the kept edges make up the training set.

Using the training set with the test edges removed, we perform parameter tuning and compute path probabilities and spring embedding distances. For the path probability algorithm, this tuning consists of choosing a probability p that corresponds to a positive edge. In all three datasets we settled on $p = 0.05$, which gives path probabilities for the edge's endpoints spread nearly evenly across the range $[0, 1]$. For the spring embedding algorithm, tuning means selecting the force functions for both positive and negative edges and choosing the dimensions of the embedding space. We found through trial and error that edges of distance d having an attractive force proportional to d^2 and repelling force proportional to $1/d^2$ lead to good distributions of points. We also choose the embedding space to be the 4-dimensional unit cube, which helps reduce the instance of nodes being "stuck" at local minima compared to a lower-dimensional space. For every edge in the training and test sets we then record its sign as well as its endpoint path probability and embedded distance.

We bucket the list of training edges into intervals based upon their path probability, and for each interval we find the embedded distance which minimizes the maximum of the ratio of mislabeled positive edges and the ratio of mislabeled negative edges. We show this process for a single path probability interval in Figure 2.

We then use these values to classify edges in the testing set. For any edge in the testing set, we find the interval that corresponds to the connectivity probability of its endpoints. If they are embedded closer than that interval's cutoff, we guess that they are positively connected. And if they are embedded further than the cutoff, we guess that their edge is negative.

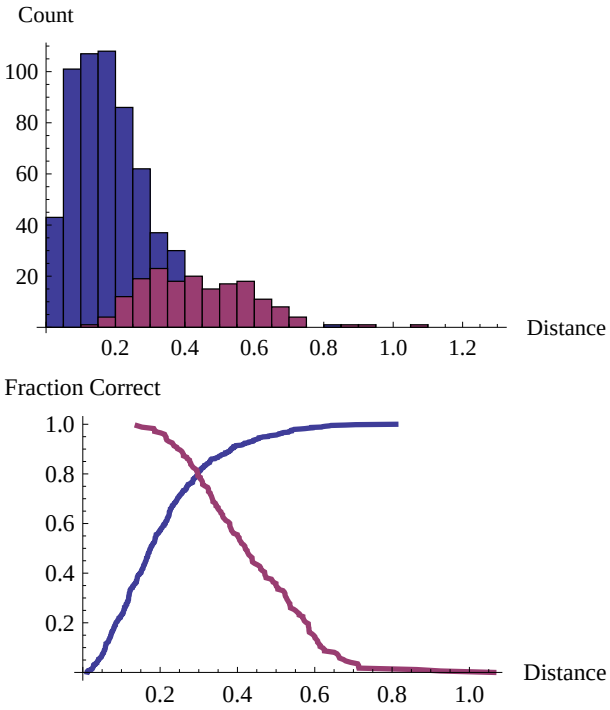

Fig. 2. The upper figure shows a histogram of the distances for all kept edges with estimated path probabilities of 0.55 (other path probabilities behave similarly). The lower figure plots the fraction of positive and negative edges classified correctly as a function of the cutoff. In both plots, positive edges are blue and negative ones are purple. We choose the point where the two lines cross in the lower figure – the point where equal ratios of positive and negative edges are classified correctly – to be the cutoff distance for this path probability.

Note that all of these networks are biased with many more positive than negative edges. Therefore our goal is not simply to have the highest ratio of correctly classified edges, but rather to correctly classify both positive and negative edges simultaneously. On such a biased dataset an algorithm which classifies edges randomly could perform quite well overall (by simply always choosing the dominant category), however the better it did on the positive edges, the worse it would do on the negatives. An algorithm which classifies edges as positive independently at random with probability p would be correct on an expected p fraction of positive edges and $1 - p$ fraction of negative edges. Using the metric of the minimum of the two ratios, such an algorithm could not do better than by choosing $p = 0.5$, so 50% correct is a minimum baseline score. This methodology eliminates the need to randomly sample the positive edges in order to "balance" the two cases – the mean behavior of a balanced set would produce the same minimum ratio for positive and negative edges.

We also explore how the accuracy of our predictions changes as more and more of the network is hidden. Hiding only a small fraction of the edges closely approximates giving our algorithm the maximum information available and represents the best that our algorithm can do. High prediction rates indicate a significant amount of redundancy in the network – the predicted edges add relatively little information that was

not already present. For each network we remove a number of edges ranging from very few to almost all of them in order to take this idea further. If we can remove a substantial fraction of edges without degrading our predictions, then there is a much larger amount of redundancy present in the network.

V. Results

For each run, we compute a separator to classify positive and negative trust relationships. The details for one iteration are shown in Figure 3, with the separator shown as a red line. Positive edges are correctly classified when they are below the line and negative edges are correct when they are above the line. Note that the noisy nature of the datasets means that we are not able to correctly classify all the edges even in our training sets. The first two columns of Figure 3 show that there are a number of points that appear on the wrong side of the line.

The third and fourth columns of Figure 3 show the data points for the test set. The results for when we treat the directed networks as undirected multi-graphs are detailed in Table I. We are able to correctly classify between roughly 86-94% of both positive and negative edges in the training set. This gives us a baseline against which we can compare our results from the test set. This test is the most similar to the approaches of Guha et al. [2] and Leskovec et al [4], so we compare them directed to these prior results. On the Epinions dataset we achieve 89% accuracy on positive and negative hidden edges simultaneously which compares well to their results of 85.3% and 86% respectively for balanced test sets. For Slashdot, we match the 81% accuracy of Leskovec et al., and for Wikipedia we achieve 81% to their 82%.

We also consider results on sets containing only highly embedded edges – those sets $E_n \subseteq E$ of all edges which are a part of at least n undirected triangles. Using the sets E_{10} and E_{25} we perform slighly better than Leskovec et al. on the Epinions and Slashdor datasets, while they maintain their 1% advantage on Wikipedia. Figure 4 shows the performance of our algorithm when restricted to highly embedded edges .

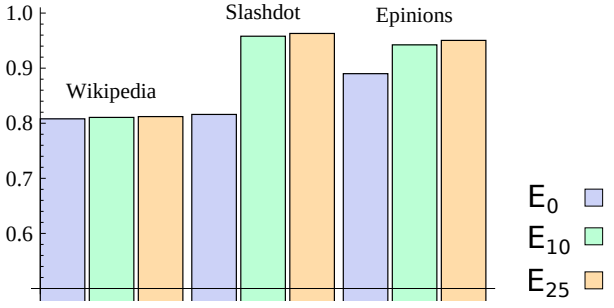

Fig. 4. This plot shows the overall accuracy rates for the sets of all edges (E_0) and edges which are a part of at least 10 (E_{10}) and at least 25 (E_{25}) triangles.

For Figure 5, we merge edges going in the opposite direction into a single undirected edge and show how our algorithm's prediction ratio changes as more edges are hidden. For each of the three networks we choose several edge removal ratios and perform ten iterations for each. For each iteration we hide edges uniformly at random, run the training algorithm, and record the minimum of the two correctness ratios. Figure 5 shows both the mean ratios and also their standard deviation.

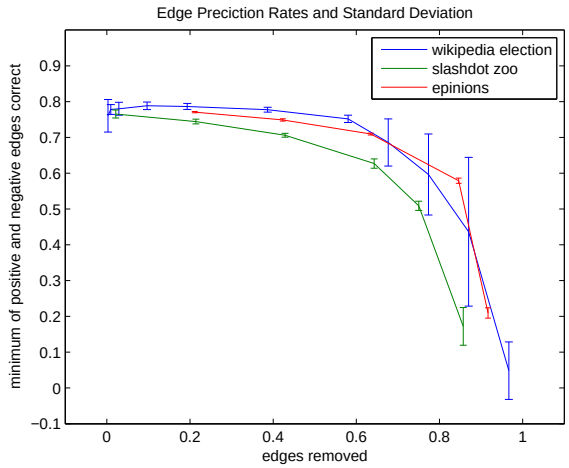

Fig. 5. Accuracy rates for the three networks as a function of the fraction of edges removed. For each fraction of edges removed we performed ten independent trials and plot the average and standard deviation.

From this result, our main observation is that accuracy rates decrease quite slowly until over 50% of the edges are hidden. This implies that these networks have a large amount of redundant information. Not only are edges highly predictable, but they are predictable even without the information from a large fraction of the edges. Once over 50% of the edges are hidden, performance degrades quickly. Eventually the endpoints of most hidden positive edges have no positive path pulling them together while most kept positive edges have few or no negative paths pushing them appart. This leads to very poor results because our training algorithm "learns" that positive edges have endpoints that are very close, but hidden positive edges having random endpoint distances. We do not view this result as a drawback of our algorithm, but rather it is a consequence of the degenerate graphs produced when most edges are hidden.

Because all three networks have many more positive edges than negative, next we describe the effects of this asymmetry. Because the algorithm optimizes the ratio of positive and negative edges guessed correctly, the results are the same regardless of the ratio of positive to negative edges. However any algorithm's performance appears skewed when viewed in the context of confidence in its predictions. When our classifier predicts a positive edge, we are quite certain (over 90%) sure of the results. However our confidence in a negative prediction lies between 50-60% (see Table I). This bias comes from the fact that the dataset is unbalanced, with many more positive edges than negative, and may be inherent in the problem. This follows from the fact that guessing 20% of the positive edges incorrectly produces roughly as many false negatives as there are correct negative edges, while guessing 20% of the negative

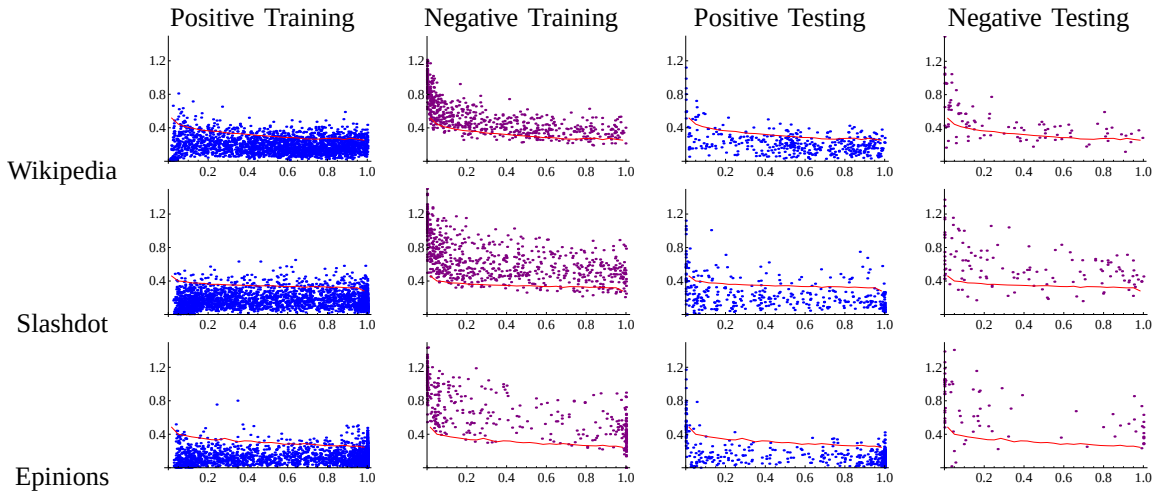

Fig. 3. This figure shows the positive and negative edges with the classification line for all three datasets. Each point in the figures corresponds to an edge in the graph. The horizontal axis is the path probability (larger values mean endpoints that are closer) and the vertical axis shows the embedded distance (smaller values mean endpoints that are closer). Points below the classification line are positive (or classified as such) and those above are negative. For clarity in viewing the larger datasets not every point is displayed, rather we display a random subset of several thousand of the points.

	Wikipedia	Slashdot	Epinions
Positive edges	0.78	0.77	0.85
Negative edges	0.22	0.23	0.15
Training edges correctly classified	0.86	0.92	0.94
Positive test edges correctly classified	0.81	0.81	0.89
Negative test edges correctly classified	0.78	0.84	0.89
Correct positive classifications	0.93	0.94	0.98
Correct negative classifications	0.51	0.60	0.61
Overall edges correctly classified	0.81	0.82	0.89
E_{10} edges correctly classified	0.81	0.96	0.94
E_{25} edges correctly classified	0.81	0.96	0.95

TABLE I
THE FRACTION OF CORRECT CLASSIFICATIONS FOR VARIOUS CRITERIA.

edges incorrectly produces far fewer false positives than true positives.

VI. CONCLUSIONS

In this paper, we presented a new algorithm for computing trust and distrust in social networks. We use a probabilistic treatment of trust combined with a modified spring-embedded layout algorithm to classify an edge. Using three different real world datasets, we showed that our method achieves very high accuracy, in the 80-90% range (from Table I), when classifying relationships as positive (trust) or negative (distrust). The results of these algorithms will be useful in many applications, including information sorting (like emails or product reviews so the most trusted can be shown first), filtering (like comments in online discussions where those from people who are distrusted should not be shown), and aggregation (as in social recommender systems, which have been shown to benefit from access to trust information).

Our results exhibit good self-consistency by performing well with respect to our classifier. Overall, the results are generally quite good, and compare well with [4] which uses more, but entirely local (path lengths of at most two) pieces of information. Our approach achieves a similar level of accuracy by reducing all interdependent paths between vertices regardless of length into two related trust values. Furthermore, [4] notes poorer results with edges that contribute to few or no triads. While this is expected in general (nodes with less direct information about their relationship should be harder to classify), our approach does a better job of using the information that is available in these cases - those trust paths of length greater than two.

At a higher level, the fact that we are able to achieve these good results speaks to the suitability of trust inference algorithms in general. Trust is a complex social relationship, and we are using realistic datasets with trust values created by real users. It is not unreasonable to question whether or not trust can be accurately computed at all, since it is so fuzzy and personal, or to question whether our probabilistic treatment is a proper one. Furthermore, distrust is a difficult addition to the range of trust values; how to properly treat it is an open question within the trust community, and one might expect that distrust would have a negative impact on our results. The fact that we were able to classify trust as positive or negative with such a high rate of success means not only that our algorithms work well, but that the underlying data is compatible with our

treatment of inferred trust.

Our work invites a number of natural extensions. We focus on trust inference in an undirected sense, which is limited in that there is no way to represent a relationship between two parties with different opinions of each other. There is no simple modification to spring-embedding which can pull one user towards a trusted second user in a way that is not symmetric. How to best apply these ideas in a directed graph is an open problem. We could also explore the relationship between network change over time and our metrics of path probability and spring-embedding distance. For example, are pairs with like paths and small distances more likely to form (or strengthen) positive connections over time than those with unlikely paths or far distances.

ACKNOWLEDGMENT

This work was supported by NSF Award CNS 1010789, NSF Award CNS-0626636, and U.S. Army Research Office grant W911NF1010350. We thank the reviewers for their help shaping the final version of this paper.

REFERENCES

[1] "Internet usage statistics - the internet big picture: World internet users and population stats," http://www.internetworldstats.com/stats.htm, 2010.

[2] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins, "Propagation of trust and distrust," in *Proceedings of the 13th international conference on World Wide Web*. ACM, 2004, pp. 403–412.

[3] T. DuBois, J. Golbeck, and A. Srinivasan, "Rigorous probabilistic trust-inference with applications to clustering," in *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology-Volume 01*. IEEE Computer Society, 2009, pp. 655–658.

[4] J. Leskovec, D. Huttenlocher, and J. Kleinberg, "Predicting positive and negative links in online social networks," in *Proceedings of the 19th international conference on World wide web*. ACM, 2010, pp. 641–650.

[5] D. do B. DeFigueiredo and E. T. Barr, "Trustdavis: A non-exploitable online reputation system," in *Proceedings of the Seventh IEEE International Conference on E-Commerce Technology*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 274–283. [Online]. Available: http://portal.acm.org/citation.cfm?id=1097108.1097189

[6] R. Levien and A. Aiken, "Attack-resistant trust metrics for public key certification," in *7th USENIX Security Symposium*, 1998, pp. 229–242. [Online]. Available: citeseer.ist.psu.edu/levien98attackresistant.html

[7] C.-N. Ziegler and G. Lausen, "Spreading activation models for trust propagation," in *Proceedings of the IEEE International Conference on e-Technology, e-Commerce, and e-Service*. Taipei, Taiwan: IEEE Computer Society Press, March 2004. [Online]. Available: citeseer.ist.psu.edu/ziegler04spreading.html

[8] U. Kuter and J. Golbeck, "Using probabilistic confidence models for trust inference in web-based social networks," *ACM Trans. Internet Technol.*, vol. 10, no. 2, pp. 1–23, 2010.

[9] P. Avesani, P. Massa, and R. Tiella, "Moleskiing.it: a trust-aware recommender system for ski mountaineering," *International Journal for Infonomics*, 2005.

[10] C. Hang, Y. Wang, and M. Singh, "An adaptive probabilistic trust model and its evaluation," in *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 3*. International Foundation for Autonomous Agents and Multiagent Systems, 2008, pp. 1485–1488.

[11] J. Patel, W. Teacy, N. Jennings, and M. Luck, "A probabilistic trust model for handling inaccurate reputation sources," *Trust Management*, pp. 193–209, 2005.

[12] A. Jøsang, S. Marsh, and S. Pope, "Exploring different types of trust propagation," *Trust Management*, pp. 179–192, 2006.

[13] A. Goyal, F. Bonchi, and L. V. Lakshmanan, "Learning influence probabilities in social networks," in *WSDM '10: Proceedings of the third ACM international conference on Web search and data mining*. New York, NY, USA: ACM, 2010, pp. 241–250.

[14] J. O'Donovan and B. Smyth, "Trust in recommender systems," in *Proceedings of the 10th international conference on Intelligent user interfaces*. ACM, 2005, pp. 167–174.

[15] P. Avesani, P. Massa, and R. Tiella, "A trust-enhanced recommender system application: Moleskiing," in *Proceedings of the 2005 ACM symposium on Applied computing*. ACM, 2005, p. 1593.

[16] J. Golbeck, "Generating predictive movie recommendations from trust in social networks," *Trust Management*, pp. 93–104, 2006.

[17] A. Galland, S. Abiteboul, A. Marian, and P. Senellart, "Corroborating information from disagreeing views," in *WSDM '10: Proceedings of the third ACM international conference on Web search and data mining*. New York, NY, USA: ACM, 2010, pp. 131–140.

[18] U. Kuter and J. Golbeck, "Semantic web service composition in social environments," in *8th International Semantic Web Conference (ISWC2009)*, October 2009. [Online]. Available: http://data.semanticweb.org/conference/iswc/2009/paper/research/137

[19] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering," in *Proceedings of the Fifth International Conference on Computer and Information Technology*, 2002. [Online]. Available: citeseer.ist.psu.edu/sarwar02recommender.html

[20] T. DuBois, J. Golbeck, J. Kleint, and A. Srinivasan, "Improving recommendation accuracy by clustering social neworks with trust," in *Proceedings of the ACM RecSys 2009 Workshop on Recommender Systems and the Social Web*, October 2009.

[21] D. Ramage, P. Heymann, C. D. Manning, and H. Garcia-Molina, "Clustering the tagged web," in *WSDM '09: Proceedings of the Second ACM International Conference on Web Search and Data Mining*. New York, NY, USA: ACM, 2009, pp. 54–63.

[22] S. Xu, T. Jin, and F. C. M. Lau, "A new visual search interface for web browsing," in *WSDM '09: Proceedings of the Second ACM International Conference on Web Search and Data Mining*. New York, NY, USA: ACM, 2009, pp. 152–161.

[23] D. Xing, G.-R. Xue, Q. Yang, and Y. Yu, "Deep classifier: automatically categorizing search results into large-scale hierarchies," in *WSDM '08: Proceedings of the international conference on Web search and web data mining*. New York, NY, USA: ACM, 2008, pp. 139–148.

[24] L. Chen, P. Wright, and W. Nejdl, "Improving music genre classification using collaborative tagging data," in *WSDM '09: Proceedings of the Second ACM International Conference on Web Search and Data Mining*. New York, NY, USA: ACM, 2009, pp. 84–93.

[25] S. Dasgupta, "Learning mixtures of gaussians," in *FOCS '99: Proceedings of the 40th Annual Symposium on Foundations of Computer Science*. Washington, DC, USA: IEEE Computer Society, 1999, p. 634.

[26] M. F. Balcan, A. Blum, and A. Gupta, "Approximate clustering without the approximation," in *SODA '09: Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2009, pp. 1068–1077. [Online]. Available: http://portal.acm.org/citation.cfm?id=1496886

[27] J. A. Hartigan and M. A. Wong, "A K-means clustering algorithm," *Applied Statistics*, vol. 28, pp. 100–108, 1979.

[28] D. S. Hochbaum and D. B. Shmoys, "A best possible heuristic for the k-center problem," *Mathematics of Operations Research*, vol. 10, no. 2, pp. 180–184, May 1985. [Online]. Available: http://dx.doi.org/10.1287/moor.10.2.180

[29] R. Xu and D. Wunsch, "Survey of clustering algorithms," *IEEE Transactions on Neural Networks*, vol. 16, no. 3, pp. 645–678, May 2005. [Online]. Available: http://dx.doi.org/10.1109/TNN.2005.845141

[30] F. Heider, "Attitudes and cognitive organization," *Journal of Psychology*, vol. 21, no. 2, pp. 107–112, 1946.

[31] J. Leskovec, D. Huttenlocher, and J. Kleinberg, "Signed networks in social media," in *Proceedings of the 28th international conference on Human factors in computing systems*. ACM, 2010, pp. 1361–1370.

[32] J. Leskovec, K. Lang, A. Dasgupta, and M. Mahoney, "Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters," *Internet Mathematics*, vol. 6, no. 1, pp. 29–123, 2009.

[33] M. Richardson, R. Agrawal, and P. Domingos, "Trust management for the semantic web," *The SemanticWeb-ISWC 2003*, pp. 351–368, 2003.

[34] P. Eades, “A heuristic for graph drawing,” *Congressus Numerantium*, vol. 42, pp. 149–160, 1984.