A Distributed Trust-based Recommendation System on Social Networks

Karan Sarda, Priya Gupta, Debdoott Mukherjee, Smruti Padhy, Huzur Saran
Department of Computer Science and Engineering
Indian Institute of Technology, Delhi

ABSTRACT
In everyday life, we seek suggestions from people we know for deciding the best place to buy a particular good or service. In this work, we put forth a framework of an automated distributed recommendation system on a social network that exploits the widely studied concept of trust, to get personalized responses. The main contribution of our model is to combine two forms in which trust is perceived, the friendship trust and domain-expertise based trail levels, to efficiently propagate a query on a social network. We empirically validate the role of trust in online social networks by crawling the online social networking site Orkut and evaluate our recommendation system against one in which trust has not been used. We also analyze the results by varying different parameters of our model. Our framework is designed to work in a distributed environment and can thus be useful with upcoming technologies such as Mobile Ad Hoc networks, P2P networks and Semantic Web.

1. INTRODUCTION
In everyday life, we rely on recommendations from others to decide the best place to buy a desired product or avail a service. In recent times, we have become increasingly dependent on the Internet to aid us in taking such decisions. With the Internet containing a vast repository of information on almost all products and services, it becomes very difficult to find suggestions that would suit our needs. Moreover, even the best search engines and discussion forums do not give us personalized responses to queries and we end up getting inputs from people who often have disparate tastes. In such a scenario, our social network becomes crucial, as we can rely on recommendations coming from our friends. The degree of trust we have in our recommender friend plays an important role in ascertaining how much we would value her recommendation. This paper illustrates how trust may be used to build an effective recommendation system over social networks.

Recommendation systems have long been used on the Internet to answer user queries. Sinha et al. [1] note that people prefer recommendations from friends rather than general recommendation systems and further show in [2] that users prefer recommendations from systems they trust. Ziegler and Lausen [3] indicate a correlation between trust and user similarity in an empirical study of a real online community. Such evidence motivates the use of social networks to drive a trust-based recommendation system.

The real life situation that we want to model can be illustrated with the following example. Say, for instance, Alice wants to buy a new camera and seeks advice from her friends. They respond with their shop preferences, if they have any, or else they further ask their friends about the best camera shop in town and get back to Alice. Finally, Alice receives many recommendations from friends or friends-of-friends and so on. After assessing the recommendations based on how much faith she has in the recommender and the recommender’s rating for that shop, Alice goes ahead to make a choice.

Modeling the above described word-of-mouth scenario over online social networks is hard for a number of reasons. Firstly, one is faced with the subjective problem of quantifying interpersonal trust between users and obtaining initial values. Secondly, a sound estimation of transitive trust between two users who are not directly connected in the social network is imperative to build a successful recommendation system. Lastly, one has to devise an efficient (on bandwidth usage) yet effective query propagation mechanism in which the queries may be selectively routed to users who have sufficient knowledge to respond to them. These difficulties become manifold when we want to design the recommendation system in a distributed setting and have it work even if some of the nodes of the network are down.

Due to the inherent fuzziness associated with trust, it’s not surprising that there exists no single definition for trust across all disciplines and contexts where it has been studied. In our model, we have assimilated two aspects of trust. The first notion of trust, what we call friendship-trust, is the faith or belief a person has in the recommendations of her friend irrespective of the context. This can be due to closeness of relationship or a conviction that the friend would have thoroughly delved into the issue before recommending anything. Secondly, we capture the context-sensitive component of trust, what we call domain-expertise, that arises from a perception of one’s knowledge in a particular domain. For example, friends from whom you solicit help for buying sports goods might be different from those whom you approach when in need of building materials. This has been incorporated by the notion of trail levels, that help direct queries on links from where a good response was received in the past for a similar query. This prevents a complete
focusing of the query at each node (as in the model proposed by Walter et al. [4]). This is one of the keys in tackling the enormity of social networks and preventing overflow of requests at any node. In further sections, the term trust and friendship trust will be used interchangeably.

Rest of the paper is organized as follows: In Section 2, we discuss the background and some related work and in Section 3, we describe our proposed model. We detail our simulation of the proposed model in Section 4 and analyze the results in Section 5. We discuss the applications, limitations and future work in Section 6 and finally, we conclude in Section 7.

2. BACKGROUND AND RELATED WORK

The concept of trust has been studied extensively in social sciences with one of the earliest references being made by Morton Deutsch in 1958. Gambetta [6] defines trust as the level of subjective probability with which an agent assesses another agent’s future behavior. Marsh’s formalization [7] of trust and classification of the same has been widely cited in computer science.

The cardinal issue in any trust-based system in social networks is how to infer trust between people who are not directly connected in the network. Many algorithms have been proposed for such trust computation in different contexts. Advogato’s maximum flow trust metric [8] works towards calculating global trust estimates relative to a set of good peers. The Appleseed algorithm [9] by Ziegler and Lausen normalizes the trust values for each person and thus does not perform satisfactorily in social network setting where the degrees of nodes can vary a great deal. Guha [10] proposes a propagation model for trust and distrust, but works with an overly simplistic trust scale of 1, 0 and -1, which is unable to capture the gradation of interpersonal trust.

The computational model of trust proposed by Golbeck [11] suits our purposes best. Equation 1 describes how transitive trust values are inferred in the model using a weighted average over all neighbors.

\[
t_{is} = \frac{\sum_{j \in N(i)} \left[ \frac{(t_{js} \times t_{ij})}{t_{ij}} \right] \text{ if } t_{ij} \geq t_{js}}{\sum_{j=0}^{N(i)} t_{ij}}
\]

where \(t_{is}\) is the trust \(i\) has in a non-adjacent node \(s\) and the summation is over all neighbors \(j\) of \(i\). This formulation allows a continuous rating scale to be used. It may be classified as a distributed local trust model and hence fits perfectly in a distributed setting such as ours. Although the trust estimates always remain conservative, the trust between two nodes even four levels apart in the social network may still be high enough for them to share recommendations. In a simple multiplication model, like the one used in Walter et al. [4], trust decays very fast and is rendered unusable after about two hops. Moreover, such a multiplication model does not specify how to compute trust between two nodes when they are connected by more than one path. Golbeck’s model also deals with low trust (possibly malicious) nodes quite well. The weighted averaging construct has been carefully crafted to ensure that low trust neighbors have less voting power in any trust computation that they are involved in.

In recent times, a lot of research has focussed on recommendation systems and several commercial online recommendation systems have also sprung up in different spheres. Massa categorizes these depending on their area of application in [12]. A number of recommendation systems have been proposed which incorporate the notion of trust and a related concept of reputation in their system to filter quality information. Reputation is indicative of the global trust acquired by a user on the basis of her past behaviour and has no relation with interpersonal faith or belief. Recommendation systems like O’Donovan [13], Montaner [14] and eBay.com employ such reputation mechanisms. Perich [15] proposes a distributed reputation modeling system in ad-hoc networks. The potential of social networks in diffusion of word-of-mouth has also been acknowledged in literature.

Walter et al. [4] propose the use of social network information in recommendation systems and analyze the impact of trust dynamics on the performance of such a system. They study the effect of preference heterogeneity of agents and network density on usefulness of trust in the system. The authors take a random directed graph for the underlying social network structure and have considered only 100 agents with a limited number of items in which agents seek recommendations, taking it far from a real-world scenario. We conquer these limitations by taking actual online social network data, and a large database of items. Moreover, trust according to them is based on past experience of recommendations, which is the domain expertise in our context, but they do not talk about friendship trust.

3. PROPOSED MODEL

We present a distributed trust-based recommendation system on a social network. There are three major components on which our system is built. Firstly, there should be a social network with friendship-trust values associated with each edge. Secondly, we require a model for computation of trust values between non-adjacent nodes. Finally, each node should have a knowledge base (KB), which is a listing of the vendor preferences (tagged with a rating) that the node has for various products and services. Given these components, we build an automated model to compute recommendations. (Henceforth, the term product will refer to both products and services)

3.1 Trust Modeling

We propose a distributed local trust model for our social network where every node stores its trust values for all other nodes in the network in a trust database, after all transitive trusts have been computed. We prefer to store trust values rather than compute them at run-time for two reasons. Firstly, during run-time, the nodes that are down will not participate in the trust computation and so the values obtained will not be ‘true’ trust values. Secondly, with trust estimation being a very frequent task in our model, a dynamic computation scheme would not allow queries to be answered in real-time. Now, it might seem that this means a
lot of storage space, but even in a network with over a million nodes, the trust database can only grow to a few MBs of storage space for each node. We use a continuous scale of $[0,1]$ for friendship-trust. In this work, we assume both the network and the trust values to remain static. A pseudo-static assumption maybe considered in future i.e. the change in trust levels and network is over weeks and months while you may see hundreds of queries a day. So incremental re-computation for the trust weights, say, overnight or every week, may be a viable approach. The exact implications of such a strategy in terms of bandwidth, and processing time comprise future work.

3.2 Trail Levels

The basic premise behind trail levels is that friends who have given good responses for a certain product are likely to do the same for ‘similar’ products. For this, a node should be able to classify all products into product-classes. Every node tracks the quality of responses received from each of its links under these product classes through these trail levels. High trail level for a particular product-class on a link signifies that high quality responses on products of that class have been received through that link. Updation of trail levels is discussed in Section 3.4.

3.3 Query Propagation

A node on the social network can generate a query seeking recommendations for any product. This node, called the originator, creates a query with the following structure: $(query_id, product, trust_threshold, rating_threshold, max_hops, visited_nodes)$, where query_id uniquely identifies the query; product is the product for which the recommendation is sought; trust_threshold defines the minimum value of trust in an immediate neighbor so that the query can be propagated on that link; rating_threshold is the minimum satisfactory rating of a vendor as desired by the originator; max_hops defines the maximum number of hops the query can make; and visited_nodes is the list of all nodes visited by the query in its propagation from the originator. Each node adds its node-id to visited_nodes before propagating the query further.

When a node C receives a query from its neighbor B, it undertakes steps as outlined in Algorithm 1.

Algorithm 1 Query Propagation

1: if product $\in$ KB of V and rating r ($> \text{rating\_thresh}$) then
2: Send response as (C, V, r) to B
3: else if length(visited_nodes) $< \text{max_hops}$ then
4: for all friends X of C s.t. trust$_{CX} > \text{trust\_thresh}$ do
5: Compute $\eta_X = \beta_{*} \text{trust}_{CX} + (1 - \beta) * \text{trail}_{CX}$
6: end for
7: Add C to visited_nodes
8: Send query to few random links from top N in order of $\eta$
9: end if

If C has an entry in its knowledge base for that product with a satisfactory rating, it sends back a response and does not propagate the query further. Otherwise, it propagates the query further to its neighbors if the number of hops done till now is less than max_hops. For propagation, the node considers all its neighbors in whom its trust is greater than the trust_threshold. Using the trust for a neighbor and the trail value for the product’s product-class for that link, the node calculates the attractiveness, $\eta$, of that link as in Step 5 of the algorithm. Here, $\beta$ defines the relative weights of trust and trail in deciding the attractiveness of the link. We have also defined a Maximum Propagation Number, N, as the maximum number of links on which a node can propagate the query further. This prevents flooding of the query and is closer to real life where a person asks only a limited number (about 10-15) of her friends for recommendations. The node C then selects a few random links from the top N links in order of attractiveness, $\eta$, and propagates the query to these neighbors. The query propagation is happening automatically with help of existing knowledge and trust databases, without needing any user input.

3.4 Query Response Accumulation

A response is a tuple of the form: (recommendation, vendor, rating). Algorithm 2 details the steps taken when a node B receives a response from its neighbor C, with Y being the recommender. B combines its transitive trust in Y with the rating linearly to compute the score that is indicative of the quality of the response (Step 1). B then updates the trail level for this product-class on the link B $\rightarrow$ C using the Exponentially Weighted Moving Average (EWMA) for trails (Step 2). B finally sends the response to A, from whom it had received the query. Trust and rating maybe combined using other non-linear functions too.

Algorithm 2 Response Retrace

1: Compute score = trust$_{BY} \gamma \ast$ rating$\delta$
2: Compute new trail = $\rho \ast$ score + $(1 - \rho) \ast$ old trail
3: Update trail level for link B $\rightarrow$ C to new trail
4: Send response to A, who had sent the query to B

Here $\rho$ is the smoothing factor in calculating the EWMA for trails; $\gamma$ and $\delta$ give relative weights to trust and rating in the evaluation of a response. As the system approaches a steady state (after accumulating sufficient trail levels), these parameters can be varied to determine the values that gives the best recommendations. Responses trace back the path of the query and at each node trail levels are updated to assist future queries. After a sufficient amount of time T, the originator selects the response which has the best score and chooses the respective vendor for the product it desired.

4. SIMULATION

The model proposed in the above section was implemented as an independent application and experiments were performed to evaluate and validate the model. A useful of application of this model can be as a plug-in over existing social
networks such as Facebook to recommend any item. We can use social network databases and links to forward and answer queries. Such an implementation needs real data - actual trust values and user preferences to bootstrap the system. However, obtaining this real data still remains a big challenge. The trust survey we set up is a step in that direction. None of the present day online social networks have trust values in-built, although popular websites like Orkut and Facebook have recently introduced similar features. These trust values can either be taken as input from the user or there can be ways to infer them. Moreover, user feedback mechanism can be devised by which we can test the usefulness of the recommendations and also update the trust values accordingly. Comparing the results with global reputation systems might also be a considered a viable approach in the future. In the following subsections, we describe the simulation of the different components.

4.1 Social Network Structure

The underlying social network was created by crawling a popular social networking site called Orkut. Crawling was done by the snowball sampling method where one seed node is randomly selected and a BFS is performed till the desired number of nodes have been crawled. Snowball sampling method with early termination tends to overestimate the node degree, but is known to preserve most of the structural properties of the network [16]. Also, starting from a random seed node can be justified since a node with a very high degree is reached within 3-4 hops. We crawled about 200,000 users, which corresponds to 0.3% of the total population of 67 million users at the time of crawling. Further, we removed the nodes on the boundary whose friend-list had not yet been crawled, thus creating a sample network of 10,000 nodes. The degree distribution in our network exhibited power-law with degree exponent $\alpha \sim 3.5$ in higher degree regions and $\alpha \sim 1.81$ in the lower degree regions. The clustering coefficient was calculated as 0.4141. Both these results are of the same order as observed by Ahn [17] and Mislove [18]. For lack of space, details of this network analysis have been omitted.

4.2 Initial Trust Distribution

For our simulation, we set up a trust survey to obtain a general distribution of trust in Orkut. The survey crawled the participant’s friend list from Orkut and the participant then rated each of her friends on a scale of 1-5, depending on how much faith she has in that person’s recommendations. We publicized the survey among the people we know, mostly from our institute, and around 300 people participated in the survey. The data obtained is shown in Figure 1. This distribution of trust values was then imposed on the entire network by choosing an edge at random and probabilistically assigned it a trust value. Then, we computed the trust between pairs of non-adjacent nodes using the formula in Equation 1, as discussed in Section 2.

4.3 Knowledge Base

We simulate the Knowledge Base at every node as a table of the form $\langle product, vendor, rating \rangle$. We used a comprehensive database of products and services provided by [20], pruned it for our purposes to obtain an ontology tree with 3-level classification having 10 product categories at level 1, 60 product-classes at level 2, and about 240 products at the 3rd level - the leaves. We allocated 100 vendors for each of the 10 product categories. Next, for each product, we assigned 5 to 15 vendors out of the 100 vendors of its category. This completed the database of possible products and their vendors. Now, we chose 20-30 products for each node and randomly selected a vendor out of those providing that product. Ratings were assigned on a scale of [0,1].

Thus each user has the following information stored as a table $\langle product, vendor, rating \rangle$ - a tuple describing the product he has used, the vendor he prefers and the rating that he has assigned to that vendor for that product. In the current model, this information is static. In a further work, we will investigate a dynamic knowledge base.

4.4 Query and Response Mechanism

We simulated the query generation and propagation mechanisms as outlined in Section 3.3. A random node is chosen as the originator who seeks recommendation for a product chosen at random. Query is propagated and responses are received in accordance with the Algorithm 1. After sufficient amount of time $T$, all propagations are stopped and responses that have reached the originator of the query are evaluated by the originator who selects the one with the maximum score. We note the branching factor of a query, defined as the average number of neighbors to which the query is propagated at each step. In the next section, we discuss the results obtained from these simulations and analyze the advantages achieved through our model.

5. RESULTS

We simulated ~10000 query propagations in the recommendation system, and recorded the results. Table 1 summarizes various parameters of the model and the values assigned to them during the simulations.

We plotted the Cumulative Distribution Function (CDF) of scores obtained against number of queries with that score. From the CDF, it was observed that about 75% of the scores were between 0.2 and 0.6. We classify this range as average.
score, scores below this range as bad, and above this range as good. Mean score was 0.355 with a standard deviation of 0.175. The Average Branching Factor, $B$, over all queries was found to be 12. In order to verify that the system had stabilized, we plotted CDFs for subsets of $\sim 200$ queries. We observed similar characteristics for all subsets, thus verifying that the results obtained on 10000 queries were representative of the system. Now we discuss the various configurations in which the system was run, and results obtained therein. Further we analyze the results, evaluate the effectiveness of the model and optimize various design parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trust Threshold</td>
<td>0.27</td>
</tr>
<tr>
<td>Rating Threshold</td>
<td>0.3</td>
</tr>
<tr>
<td>Maximum Hops</td>
<td>4</td>
</tr>
<tr>
<td>Maximum Propagation</td>
<td>20</td>
</tr>
<tr>
<td>$\beta$</td>
<td>0.5</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>1.0</td>
</tr>
<tr>
<td>$\delta$</td>
<td>1.0</td>
</tr>
<tr>
<td>$\rho$</td>
<td>0.7</td>
</tr>
</tbody>
</table>

Table 1: Parameter Values

5.1 Comparison with Generic Recommendation System

We consider a generic recommendation system in which queries are propagated without any notion of trust. Each node that does not have a satisfactory recommendation for a query, propagates it along randomly chosen $B$ links (value of $B$ obtained from results of the original system). No trail levels are maintained during response retrace. After time $T$, all propagations are stopped and the originator selects the response with the maximum rating. Now, for the purpose of comparison, we calculate score for this response using the same formula as in Algorithm 2.

We simulated the same queries under this system and plotted the CDF of scores vs number of queries. From the CDF, it was observed that about 50 % of the scores are 0 and 45% of the scores were between 0.2 and 0.6. Mean score was 0.162 with a standard deviation of 0.198. Figure 2(a) contrasts the CDFs of the original system with that of the generic one. Further, to obtain a finer view of variations in score, we calculate difference in score for every query between the two systems. Figure 2(b) shows the CDF of this difference in scores. We observe that for over 70% queries, the system with trust gives better scores. About 30% of the queries receive same or better scores in the generic system, which is expected due to the inherent randomness in the system.

From these results, we conclude that without incorporating trust in the system, one tends to follow general recommendations that might not be suited to one’s specific tastes; whereas a trust-based recommendation system does a very good job in providing personalized recommendations. In this sense, a trust based recommendation system is a step towards personalized search.
5.2 Variation of scores with \( \beta \)

Figure 2(c) compares the score distribution obtained with different values of \( \beta \) defines the relative weights of trust and trail in deciding the attractiveness of a link. We compared the score distribution obtained with three values of \( \beta : 0.0 \) (only trail values), 0.5 (both friendship trust and trail), 1.0 (only friendship trust). We simulated 500 queries with each of the above values and recorded the scores (Figure 2(c)). From the graph we can observe that results obtained are slightly better when trust and trail values both are used \((\beta = 0.5)\) than when only trust or only trail is used. This suggests the fact that both notions of trust should be combined for efficient propagation of queries. We expect to achieve better sensitivity to \( \beta \) as the system gains more history.

5.3 Variation of scores with \( N \)

Next we varied the Propagation Number \( N \), i.e., the maximum number of links a query can be propagated to. Figure 2(d) compares the score distribution obtained. First we set \( N \) as 20, and propagated the queries to 10-15 random links from the top 20 attractive links. Next we propagated the queries to only top 5 links, and then to top 2 links. We simulated 500 queries with each of these values, and plotted the scores. From the graph we observe that the scores are best when the value is at 5, which can be explained as there is a high chance that one of these high trusted links will lead to a favourable recommendation and hence a high score. Scores drop down somewhat with the value 2 because some nodes, which might have good recommendations but may not be as highly trusted, will be missed. But both of these perform better than the value 20 in the same time limit since they are deterministic, and never lose out on the trusted links. But we assert that randomness in the system is required for the system to learn, otherwise initially less trusted paths will never be traversed and their trail values will never be updated.

6. DISCUSSION

With rapid growth in Semantic Web, peer-to-peer networks and Mobile Ad-hoc networks, it may be expected that in near future there would be formation of social networks on such infrastructure. In this context, a distributed recommendation system such as ours will prove to be more relevant than a centralized system.

The crucial problem of how to intelligently route queries, given the scale of online social networks, has lacked attention in previous proposals of distributed recommendation systems ([4], [9], [19]). The simple abstraction of trail levels that we have introduced provides an elegant way of directing queries to nodes who are capable of answering them, along the shortest possible paths.

Another feature quite common in online social networks is the formation of communities of users with similar interests. It would be interesting to analyze the performance of our system with regard to existence of such communities and observe the accumulation of trail levels on the links leading to such clusters. For example, it is expected that we would find a high concentration of the trail for sports goods on the edges leading to members of an interest group on sports.

The notion of distrust and its impact has been widely studied in literature (Guha [10], Ziegler [9]). In the context of a recommendation system, distrust will not make an impact on the responses received since queries will not be routed on ‘dis-trusted’ paths. However, the transitive trust in the recommender may change considerably, and hence adding distrust to the system may provide interesting insight.

7. CONCLUSION

We have proposed a model of a recommendation system based on a social network, which uses interpersonal trust among users of the network. By comparison with a generic recommendation system on social networks, we have empirically validated that trust can significantly enhance the utility of a recommendation system, and pave way for personalized search as well. We have used a dual notion of trust, incorporating both context-independent friendship trust, and context-specific domain-expertise based on history of queries. Query propagation occurs only along the most attractive links in accordance with both these notions. Simulation of a large social network and knowledge base helps in bringing the model closer to real life.

8. REFERENCES