FeedbackTrust: Using Feedback Effects in Trust-based Recommendation Systems

Samaneh Moghaddam
School of Computing Science
Simon Fraser University
Burnaby, BC, Canada
sam39@cs.sfu.ca

Mohsen Jamali
School of Computing Science
Simon Fraser University
Burnaby, BC, Canada
sja25@cs.sfu.ca

Martin Ester
School of Computing Science
Simon Fraser University
Burnaby, BC, Canada
ester@cs.sfu.ca

Jafar Habibi
Computer Engineering Dep.
Sharif University of Tech.
Tehran, Iran
jhabibi@ce.sharif.edu

ABSTRACT

With the advent of online social networks, the trust-based approach to recommendation has emerged which exploits the trust network among users and makes recommendations based on the ratings of trusted users in the network. In this paper, we introduce a two dimensional trust model which dynamically gets updated based on users's feedbacks, in contrast to static trust values in current trust models. Explorability measures the extent to which a user can rely on recommendations returned by the social network of a trusted user. Dependability represents the extent to which a user's own ratings can be trusted by users trusting him directly and indirectly. We propose a method to learn the values of explorability and dependability from raw trust data and feedback expressed by users on the recommendations they receive. Positive feedback will increase the trust and negative feedback will decrease the trust among users. We performed an evaluation on the Epinions dataset, demonstrating that exploiting user feedback results in lower prediction error compared to existing trust-based and collaborative filtering approaches.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Relevance feedback

General Terms

Algorithms, Experimentation

Keywords

Trust, Recommendation, Feedback

1. INTRODUCTION

Current collaborative filtering recommender systems predict ratings based on the ratings expressed by the target user and other users. The only information available for these types of recommenders are the ratings already expressed by different users.

Recently, social networking services such as Facebook, MySpace, Orkut, Flickr, ... have become very popular. Actually, they play an important role in people's daily interactions with their friends. With the advent of online social networks, the trust-based approach to recommendation has emerged. This approach exploits the trust network among users and makes recommendations based on the ratings of the users that are directly or indirectly trusted by the user seeking a recommendation. The trust network among users is very sparse. So we have to use a trust model which is able to compute trust values among indirectly connected users in the trust network.

In reality, people meet new people and give them some initial trust. Then they update their trust based on the interactions and experiences they make with these trusted friends. However, online social networks such as Facebook, Flickr, Orkut and Epinions[1] do not adequately support this real life phenomenon: They neither allow users to provide their feedback nor use their feedback to update the trust values in the network.

We envision a recommender system where users provide feedback on the recommendations they receive by telling the system their actual rating. In return, the recommender is able to provide recommendations with higher quality. The system achieves this higher quality by automatically updating certain trust values based on the user feedback. In this paper, in the absence of a system that actually allows users to provide feedback online, we simulate users feedback in an offline manner by separating the data set into training set and test set.

In order to incorporate the effect of feedback, we propose a recommendation specific trust model. In this model, we introduce two dimensions for trust: explorability and dependability. Explorability of an edge denotes the reliability of using that edge to explore the trust network and find recommendations. On the other hand, dependability of a trust edge denotes the quality of the trusted user's own knowledge for the truster. Note that both explorability and dependability are defined specifically for the purpose of recommendation. This is one of the features that distinguishes our approach from existing trust-based recommendation systems, which use a generic concept of trust with trust edges, e.g., explicitly constructed by the user or implicitly derived from observed actions such as online chats. We introduce a recommendation method which exploits explorability and dependability to predict ratings. We also propose a method to learn the values of explorability and dependability from raw trust data and user feedbacks. In other words we update the values of explorability and dependability based on feedbacks expressed by users. Positive feedback will lead to increasing the trust among users and negative feedback will decrease the trust.

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

RecSys'09, October 23–25, 2009, New York, New York, USA.

2. PRELIMINARIES

Typically in a recommendation system there is a set of N users $U = \{u_1, u_2, ..., u_N\}$ and a set of M items $I = \{i_1, i_2, ..., i_M\}$. Each user u rates a set of items $I_u = \{i_{u_1}, ... i_{u_k}\}$. The rating of user u on item i is denoted by $r_{u,i}$. $r_{u,i}$ can be any real number, but often ratings are integers in the range $[1, 5]$. In a trust-based system, we also have a trust network among users. If u trusts v, then $t_{u,v}$ denotes the strength of this trust as a real number in $[0, 1]$. In the following we define basic concepts used in our work.

Definition 1. ***Requester***: Let $u \in U$ and item $i \notin I_u$, when u requests a prediction for the rating on i, we call user u the requester and i the target item.

Definition 2. ***Rater***: Let $u \in U$ and $i \in I$, we call u a rater for i if $i \in I_u$.

When a requester u asks for a prediction on target item i, the first task of a recommendation system is to find raters. For this purpose we start searching the trust network from u, adopting a breadth-first search method to search for the raters. We put a $maxDepth$ threshold on the breadth first search to stop searching at some points. The method also labels all users visited to prevent cycles. After finding all the raters in the neighborhood, we aggregate their ratings to compute a prediction for requester u on target item i ($\widehat{r}_{u,i}$). We use a simple weighted average as the aggregation function:

$$\widehat{r}_{u,i} = \overline{r_u} + \frac{\sum_{v \in N_u} w_{u,v}(r_{v,i} - \overline{r_v})}{\sum_{v \in N_u} w_{u,v}} \quad (1)$$

In this equation, $\overline{r_u}$ is the average of ratings expressed by u. $w_{u,v}$ is the weight denoting the effect of rater v on the prediction. This weight should be related to the trust between u and v. We call this weight the *trust value* between u and v (shown by $t_{u,v}$). Note that trust relationships are not necessarily symmetric, i.e. in general $t_{u,v} \neq t_{v,u}$. So, our trust network is directed graph.

Definition 3. ***Trust Network***: represented as a directed graph $G =< U, E >$ which U are users, and E is the set of trust statement (edges) among users and is of the form <u, v>.

Definition 4. ***Predecessor***: $v \in U$ is a predecessor of u if there are $u_1, ..., u_k \in U$ and <v, u_1>, <u_1, u_2>, ..., <u_k, u> $\in E$.

Definition 5. ***Successor***: $v \in U$ is a successor of u if there are $u_1, ..., u_k \in U$ and <u, u_1>, <u_1, u_2>, ..., <u_k, v> $\in E$.

3. TRUST MODEL AND USER FEEDBACK

Social scientists have identified three types of trust [1]: dispositional, interpersonal, impersonal. Dispositional trust describes the general trusting attitude of the truster which is independent of any trustee or context. Interpersonal trust is the trust one user has in another user directly. And finally, impersonal trust refers to trust that is not based on any property or state of the trustee but rather on the perceived properties or reliance on the system or institution within which that trustee exists. The trust to a news agency is one such example. Inspired by this categorization, we define a two-dimensional trust model for using in recommender systems. In our trust network, each edge has two trust components: explorability and dependability.

Definition 6. ***Explorability***: If $u, v \in U$ and <u, v>$\in E$, then the explorability $e_{u,v}$ denotes the extent to which u and its predecessors can rely on recommendations returned by v's successors. In other words, how useful are v's successors' rating for u and its predecessors.

Definition 7. ***Dependability***: If $u, v \in U$ and <u, v>$\in E$, then the dependability $d_{u,v}$ denotes the extent to which u and its predecessors can depend on v's own ratings. In other words how useful are v's own ratings for u and its predecessors.

Definition 8. ***Trust Statement***: A trust statement t (which is a property of an edge) is an ordered pair $t(e, d)$ where the first entry e denotes explorability and the second entry d indicates dependability of the edge.

The ranges of explorability and dependability are [0,1]. The dependability corresponds to the quality of trustee's own knowledge, while the explorability represents the quality of recommendations returned by trustee's social network.

We need to clarify two issues. The first issue is, "why do we separate the concept of trust into two new concepts?". When we talk about trust in a person, there are two dimensions: trust in his own opinions, and trust in the opinions returned by him coming from his trusted neighbors. The first kind of trust, dependability, is an interpersonal trust which shows the trust in other user's knowledge, and the second one, explorability, is an impersonal trust which represents the trust in the social network of the trustee.

The second issue is, "why are explorability and dependability are properties of edges and not of nodes?". Explorability denotes the reliability of using the edge to explore the trust network, while dependability denotes the quality of the target user's own knowledge. For a specific user u, different in-links connecting to u may have different values of dependability and explorability, since links may come from different communities of users. For example, in the context of book recommendation, a university professor will have high level of dependability in the academic community, since he knows lots of specialized books. However, he will not be a good recommender for his daughter about story books, or for his friend about psychology books. So, in the later communities he will have a low level of dependability. This example shows that dependability is community-dependent and so is a property of an edge connecting a user to a community. In the same way, explorability is community-dependent and should be modeled as a property of an edge.

3.1 Computation of Trust Values

In this subsection, we assume that all the values of explorability and dependability have already been learnt. The learning mechanism is discussed in the next subsection.

If the actual trust value between u and v be $t_{u,v}$, $\hat{t}_{u,v}$ denotes an estimate of $t_{u,v}$. There are two situations when we want to estimate the trust value between two users: they are direct friends in the trust network, or they are indirectly connected to each other using a trust path. Considering the first situation, let $< u, v > \in E$, then we have explicit values for $e_{u,v}$ and $d_{u,v}$. $\hat{t}_{u,v}$ should represent the effect of v's own rating on the prediction for u. So we assign $\hat{t}_{u,v} = d_{u,v}$.

In the second situation, let <u, v>$\notin E$, v is a rater found in the neighborhood, there should be some trust paths from u to v in the trust network. First we compute the trust for a single path. Consider a path $p =< u, w, v >$ from u to v. We assign the trust of a path as the product of the trust values of the edges on the path. The factors effecting the trust of the path are the explorability of the edge $< u, w >$ and the dependability of the edge $< w, v >$. So we have $\hat{t}^p_{u,v} = e_{u,w} \times d_{w,v}$. In general, for a path $p =< u, u_1, ..., u_k, v >$ we have:

$$\hat{t}^p_{u,v} = e_{u,u_1} \times \prod_{l=1}^{k-1} e_{u_l,u_{l+1}} \times d_{u_k,v} \quad (2)$$

The intuition behind equation (2) is that your trust to a neighbor's opinion depends on his own knowledge plus the reliability of the trust path connecting him to you. Notice that the trust value for a direct friend is equal to his dependability.

To compute $\hat{t}_{u,v}$ we need to consider all different paths from u to v. In FeedbackTrust, when the BFS method finds a rater in depth k, no more pathes with length of more than k will be considered. In other words, the search method only finds the shortest paths between the requester and the rater (similar to [2]). So all paths between the requester and the rater have the same length. To obtain

a single trust value $\hat{t}_{u,v}$, we have to aggregate the trust values of different paths. We use a simple averaging method for aggregation: $\hat{t}_{u,v} = (\sum_{p \in P} \hat{t}^p_{u,v})/|P|$, where P is the set of all paths found by the BFS method which connect requester u to rater v. Notice that if we used max or min instead of taking the average, then paths with outlier trust values could dominate the aggregated trust value. Taking the mean balances the effect of different paths.

3.2 Learning From User Feedback

So far, we have discussed the recommendation method given the values of explorability and dependability. But how does the system learn the values of $e_{u,v}$ and $d_{u,v}$ for all $<u, v> \in E$? To learn these values we need a training set with known ratings (for a subset of item-user pairs) and with known trust edges (but without dependability and explorability values). Initially, we set all explorability and dependability values to a default of 0.5. Then we withhold the known ratings (one at a time), predict them and compare the prediction to the actual rating.

More specially, when a requester u asks for a prediction on target item i, the neighborhood around u will be explored to find raters of i. Then, the ratings of raters will be returned along the path between u and the rater. After receiving a recommendation from raters in the neighborhood, the requester provides feedback for each rater by actually rating the target item (the actual rating in the training data being withheld). Based on the similarity of this actual rating and the returned ratings from each rater in the neighborhood, some of the explorability and dependability values will get updated.

Explorability of a trust edge $e_{u,v}$ will get updated when u receives a recommendation from successors of v. Then, u provides a feedback for the recommendation and this feedback affects $e_{u,v}$. The dependability of the edge, $d_{u,v}$, will get updated when u or its predecessors receives recommendation from v's own ratings. In this case, the user who receives this recommendation provides a feedback which affects $d_{u,v}$. To determine the feedback user v receives from u, we first compute the error of the prediction u received from v about item i: $Err_{u,i,v} = (|r_{u,i} - r_{v,i}|)/Range$

In the this equation, $Err_{u,i,v}$ is the normalized error of the prediction provided by rater v for requester u about item i, $r_{u,i}$ is the actual rating of i expressed by u, $r_{v,i}$ is the estimate for the rating of i provided by v, and $Range$ is the range of possible ratings on items. The value of Err will be in the $[0, 1]$ range. A low value of error indicates that the recommendation was good, and therefore the requester will give positive feedback. Note that this error is computed for each rater separately, based on his individual rating (not the aggregate rating over all raters). Finally, after computing the error of each trust path $< u, u_1, ..., u_k, v >$, FeedbackTrust updates the explorability of the first edge, e_{u,u_1}, and dependability of the last edge, $d_{u_k,v}$, in that path.

What is the intuition behind updating the explorability and dependability values of these particular edges? Assume that in the real world you request a recommendation about some item from your friend. He may not have enough information to help you in decision making. So, he asks some friends about it. Finally, he gives you a recommendation about that item. If he gives you a good recommendation, your trust in his social network will be increased, but your trust in other people's networks will not be changed even if you know them. Therefore, we update the explorability of the edge between the requester and a direct friend. Now, what happens to the actual source of your recommendation? You may not know him, however he provides you with a good recommendation. This shows that he has some good information about that item. Our method tries to highlight him, so that other users can use his information as recommendation. That is the reason for updating the dependability of the edge end to the rater.

For applying user's feedback on the specified explorability and dependability values, we define a modification rate ($MRate_{u,i,v}$) for each rater as follows: $MRate_{u,i,v} = 0.5 - Err_{u,i,v}$, with $MRate_{u,i,v}$ in the range [-0.5,0.5].

As explained before, when the rater's rating is close to the requester's rating, the prediction error will be low, otherwise it will be high. Given that the error is in $[0, 1]$, the $MRate$ formula gives us a symmetric distribution of the modification rate centered at zero. We assume that when the error of the prediction is lower that 0.5, it is a good prediction, and so the rater should receive positive feedback. On the other side, when the prediction error is higher than 0.5, it cannot be a good prediction, and so the requester will send negative feedback.

We define and use a trust updating function for explorability and dependability based on the trust evolution function proposed by Jonker and Treur in [3]. We use the modification rate as an "inflation factor" for the trust evolution function.

$$e^{t+1}_{u,u_1} = (1 - MRate_{u,i,v}) \times e^t_{u,u_1} + MRate_{u,i,v} \quad (3)$$

$$d^{t+1}_{u_k,v} = (1 - MRate_{u,i,v}) \times d^t_{u_k,v} + MRate_{u,i,v} \quad (4)$$

In equations (3) and (4), the variable t denotes the time. So, e^t_{u,u_1} denotes the value of explorability of the edge $< u, u_1 >$ before updating, while e^{t+1}_{u,u_1} indicates the updated value. $d^t_{u_k,v}$ and $d^{t+1}_{u_k,v}$ show the dependability of the edge $< u_k, v >$ before and after updating, respectively. Note that if there are different paths from u to v, then the modification will be applied to all these paths.

4. EXPERIMENTS

In this section, we present the experiments we have conducted for evaluating the performance of our method. We compared our FeedbackTrust method with MoleTrust presented by Massa, TidalTrust proposed by Golbeck, and also the standard user-based collaborative filtering method. We also evaluate the results of our model when we ignore users' feedback. We call it *BaselineTrust* method. In the BaselineTrust method all the explorability and dependability values are equal to 0.5 and never get updated. Basically, we analyze the effect of users' feedback by comparing BaselineTrust and FeedbackTrust. In the following, we first describe the dataset used and introduce the evaluation strategy, then we present the experimental results.

4.1 Dataset

The data set used in our experiments is derived from the *Epinions.com* Web site. In *Epinions.com* users can express their opinions about different products and also assign a rate (ranging from 1 to 5) to each item to show the overall quality of a product. They can also express their Web of Trust, i.e. reviewers whose reviews and ratings they have found to be valuable. When a user adds a new reviewer to his Web of Trust, it is equals to issuing a trust statement of value 1 in this reviewer.

This data set[2] contains 49k users, 139k items and 664k ratings expressed by users on different items. Also the trust network includes 487k edges. The sparsity of the rating matrix (user×item), which is equal to the percentage of empty cells, is 99.99%.

4.2 Experimental Design

To evaluate our method, we performed 5-fold cross-validation. Since our method needs to learn trust values using feedbacks, we took one fold of the ratings ("user-item-rate" entries) as test set and used the remaining folds for training the trust network. Typically, the *leave-one-out* method is used to evaluate recommendation systems [2][4]. This technique involves withholding one rating and

[2]http://www.trustlet.org/wiki/Downloaded_Epinions_dataset

trying to predict it using the trust network and the remaining ratings. Then the predicted rating can be compared with the actual rating and the difference will be considered as the prediction error. We use two evaluation measures:

Mean Absolute Error(MAE), measures the deviation of predictions generated by the Recommender from the true rating values, as they were specified by the user. In recommender system area if the size of the test set be M, and prediction and actual rating represent by p_j and r_j respectively, MAE will be equal to: $MAE = (\sum_{j=1}^{M} |p_j - r_j|)/M$

Coverage, is a measure of the percentage of ratings for which, after being hidden, the algorithm can provide predictions. The coverage is defined as $Coverage = (|T| - m)/|T|$, which m is the number of ratings the method could not predict, and $|T|$ is the size of the test set.

4.3 Experimental Results

Cold start users are those who in total provided less than five ratings[4]. In terms of the maximum searching depth in trust network, we report the results for three depths. Finally, to analyze the effect of users' feedback, we report the results of BaselineTrust and FeedbackTrust separately.

Table 1 and 2 show the MAE and coverage of FeedbackTrust and other methods, for cold start and all users. In each table we show the results of each method for different predefined maximum depth (d = 1, 3, 5). Of course, standard collaborative filtering (CF) and TidalTrust methods are independent form the depth of search.

Models	Cold Start Users			All Users		
	d=1	d=3	d=5	d=1	d=3	d=5
BaselineTrust	0.839	0.912	0.846	0.832	0.814	0.801
FeedbackTrust	0.817	0.871	0.757	0.794	0.759	0.706
MoleTrust	1.04	1.114	1.069	0.859	0.839	0.825
TidalTrust	0.856	0.856	0.856	0.825	0.825	0.825
Standard CF	1.077	1.077	1.077	0.961	0.961	0.961

Table 1: Mean Absolute Error (MAE) for different methods.

Comparing the MAE results of cold start users and all users shows that in all models the error of prediction for cold start users is higher than all users' error. This is due to the fact that less information is available for cold start users, so the methods are unable to find enough raters to precisely predict the ratings.

The comparison of MAE of BaselineTrust and FeedbackTrust supports our claim that users' feedback can reduce the prediction error and improve the precision of recommendations. As shown in table 1, using users' feedback decreases the MAE for both cold start users and all users. The MAE of BaselineTrust is similar to TidalTrust's, but the FeedbackTrust model performed much better especially when the maximum depth of search increases. Also we can see that trust-based approach generally outperformed the standard collaborative filtering approach. Among trust-based approaches, the results of BaselineTrust method are not as good as TidalTrust, but the FeedbackTrust performs consistently better than TidalTrust, in particular for larger maximum depth values.

Models	Cold Start Users			All Users		
	d=1	d=3	d=5	d=1	d=3	d=5
FeedbackTrust	6.76	41.7	52	26.33	71.57	76.25
MoleTrust	6.63	43.73	53.52	26.24	71.16	75.57
TidalTrust	51.53	51.53	51.53	76.08	76.08	76.08
Standard CF	19.75	19.75	19.75	72.37	72.37	72.37

Table 2: Coverage(%) of different methods.

Finally, we compare the coverage of all methods. In general, the coverage of cold start users is less than the coverage of all users. Cold start users, who rated few number of items, usually have only small trust networks. So, it is hard for both trust-based methods and collaborative filtering methods to find a rater. However, trust-based methods perform better than CF for cold start users. As soon as a user adds one friend to his trust network, he gets connected to some raters in farther distances.

The coverage of depth-dependent trust-based methods, i.e. FeedbackTrust and MoleTrust, increases with increasing maximum depth (for both cold start users and all users). Since these two methods use the same search method, they show similar coverage. Finally, the coverage of TidalTrust and CF is fixed and independent from the search depth. TidalTrust has high coverage for both cold start users (51%) and all users (76%). While the coverage of FeedbackTrust for maximum depth 5 is the same as TidalTrust's coverage, for the other two maximum depths it has lower coverage.

5. CONCLUSION

In this paper, we proposed a novel trust model which considers two dimensions for trust: explorability and dependability. The explorability of the edge between users u and v denotes how useful the ratings of v's successors are for u and its predecessors. The dependability of that edge denotes how useful v's own ratings are for u and its predecessors. Both dimensions of trust introduced in this paper are specific to recommendation systems which makes them work better for prediction of ratings.

We introduced a feedback-based method to learn both explorability and dependability values. Good recommendations, where the recommendation agrees with the user's actual ratings, improve the dependability of the trust edge connecting to the rater and also improve the explorability of the edge from which we started to explore the network, while bad recommendations decreases these values. We also presented a method to compute the trust between a requester and a rater based on the explorability and dependability values along a connecting path.

Exploiting users' feedback on recommendations they receive enables us to highlight those friends who are actually reliable for users. This leads to an improvement of trust-based recommendation. This claim was confirmed in our experiments on the Epinions data set. In the absence of a system that actually allows users to provide feedback online, we simulated user feedback in an offline manner by separating the data set into training set and test set. However, curresnt recommender system can use this model by constructing a virtual network on top of the user network, and automatically generate feedbacks using requesters preferences. Experiments showed that FeedbackTrust outperforms existing trust-based recommenders MoleTrust and TidalTrust in terms of MAE. Our experimental results also demonstrated that FeedbackTrust outperforms the standard user-based CF, due to the fact that FeedbackTrust uses the trust network besides the rating data.

6. REFERENCES

[1] A. Abdul-Rahman and S. Hailes. Supporting trust in virtual communities. In *Proceedings of the 33rd Hawaii International Conference on System Sciences*, USA, 2000.

[2] J. Golbeck. Generating predictive movie recommendations from trust in social networks. In *Proceedings of The Fourth International Conference on Trust Management*, 2006.

[3] C. M. Jonker and J. Treur. Formal analysis of models for the dynamics of trust based on experiences. In *Proceedings of the 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, London, UK, 1999.

[4] P. Massa and P. Avesani. Trust-aware recommender systems. In *RecSys'07*, Minneapolis, Minnesota, USA, 2007.