Bayesian-Inference-Based Recommendation in Online Social Networks

Xiwang Yang, *Student Member*, *IEEE*, Yang Guo, *Member*, *IEEE*, and Yong Liu, *Member*, *IEEE*

Abstract—In this paper, we propose a Bayesian-inference-based recommendation system for online social networks. In our system, users share their content ratings with friends. The rating similarity between a pair of friends is measured by a set of conditional probabilities derived from their mutual rating history. A user propagates a content rating query along the social network to his direct and indirect friends. Based on the query responses, a Bayesian network is constructed to infer the rating of the querying user. We develop distributed protocols that can be easily implemented in online social networks. We further propose to use Prior distribution to cope with cold start and rating sparseness. The proposed algorithm is evaluated using two different online rating data sets of real users. We show that the proposed Bayesian-inference-based recommendation is better than the existing trust-based recommendations and is comparable to Collaborative Filtering (CF) recommendation. It allows the flexible tradeoffs between recommendation quality and recommendation quantity. We further show that informative Prior distribution is indeed helpful to overcome cold start and rating sparseness.

Index Terms—Recommender system, online social network, Bayesian inference, cold start

1 Introduction

Recommendation is playing an increasingly important role in our life. Accurate recommendations enable users to quickly locate desirable items without being overwhelmed by irrelevant information. It is of great interest for vendors to recommend to their potential customers products matching their interests, and hopefully turn them into committed buyers. No wonder, in the Netflix contest [10], an improvement of 10 percent recommendation accuracy is awarded with 1 million dollars. In real life, people often resort to friends in their social networks for advice before purchasing a product or consuming a service. Findings in sociology and psychology fields indicate that human beings tend to associate and bond with similar others, so called *homophily* [7]. Due to the stable and long-lasting social bindings, people are more willing to share their personal opinions with their friends, and typically trust recommendations from their friends more than those from strangers and vendors. The phenomenally popular online social networks, such as Facebook [12], twitter [11], and Youtube [8], provide novel ways for people to communicate and build virtual communities. Online social networks not only make it easier for users to share their opinions with each other, but also serve as a platform for developing quantitative online recommendation algorithms to automate the otherwise manual and anecdotal recommendations in real life social networks. This paper presents an effort to develop a Bayesian-inference-based recommendation algorithm for online social networks. Our algorithm operates on top of an underlying social network, either a general-purpose social network, such as the Facebook [12], or a domain-specific recommendation social network, such as Flixster [13] for movie recommendation and Epinions [9] for a wide range of product recommendation. It can be developed into a standard-alone application. It can also be adopted as the recommendation engine for existing applications in social networks, such as the Movies App [14] on Facebook.

Our algorithm is developed for recommendations for general products/services. In the rest of the paper, we use movie recommendation as an example application to illustrate the algorithm. At a high level, we assume users are connected in an online social network. Friends share with each other their ratings for movies. Our algorithm keeps track of the rating history between friends. Whenever a user wants a recommendation for a movie, he sends out a query to his direct and indirect friends within certain hops of the social network. Users who have watched/rated the movie in the past respond to the query with their ratings. Based on the responses and rating history between friends, our algorithm calculates a recommendation rating for the querying user. Within this framework, we make the following contributions:

- We propose to measure the rating similarity between friends in an online social network by a set of rating conditional probabilities. We propose to construct a Bayesian network based on the query propagation tree.
- We develop an iterative algorithm to calculate the most probable (MP) recommendation and the Bayes Mean Square Error (MSE) recommendation for a querying user.

- *X. Yang and Y. Liu are with the Department of Electrical and Computer Engineering, Polytechnic Institute of NYU, 5 Metrotech Center, Brooklyn, NY 11201. E-mail: xyang01@students.poly.edu, yongliu@poly.edu.*
- *Y. Guo is with Bell Labs, Alcatel-Lucent, 791 Holmdel Road, Holmdel, NJ 07733. E-mail: Yang.Guo@alcatel-lucent.com.*

Manuscript received 3 Oct. 2011; revised 30 May 2012; accepted 6 June 2012; published online 22 June 2012.
Recommended for acceptance by K. Wu.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2011-10-0747.
Digital Object Identifier no. 10.1109/TPDS.2012.192.

- We design distributed protocols to implement the proposed Bayesian-inference-based recommendation algorithm in online social networks. We propose to use maximum a posteriori (MAP) estimate to cope with cold start and rating sparseness.
- Using online rating data sets of real users, we show that the Bayesian-inference-based recommendation is better than the existing trust-based recommendations and is comparable to Collaborative Filtering (CF) recommendation. It allows the flexible tradeoffs between the recommendation quality and the recommendation quantity. We further show that the informative Prior distribution is indeed helpful to overcome cold start and generate ample accurate recommendations.

The rest of the paper is organized as following. We describe the related work in Section 2. We present our Bayesian-inference-based recommendation algorithm in Section 3. Implementation details of the proposed algorithm in online social networks is described in Section 4. We present the evaluation results in Section 5. The paper is concluded with summary and future directions in Section 6.

2 Related Work

Recommender systems (RSs) are usually classified into three categories: content-based recommendations, Collaborative Filtering-based recommendations, and hybrid approaches. Content-based recommendations rely on content descriptions to match the content with the users' preference. CF is the most popular approach to build recommender systems and has been successfully employed in many applications. CF recommendations use the opinions of a set of users to predict another user's interest [1], [2], [3]. Paper [2] proposed a probabilistic memory-based CF approach to learn new user's rating profile. Through an active learning scheme, the profile of a new user can be inferred with a minimum of required user effort. Paper [3] reviews the key decisions in evaluating collaborative filtering recommender systems. Paper [4] reviews matrix factorization models in CF domain. Through example of Netflix competition, author of [4] shows that matrix factorization techniques have become a dominant methodology within collaborative filtering recommenders. None of [2], [3], [4] deals with recommendation in online social networks and distributed recommender system design.

With the rapid growth of online social networks, trust-based CF approach has emerged [21], [24], [25], [26], [27], [28], [30], [33]. In trust-based CF approaches, the ratings of users socially related to the target users are also considered for recommendation.

The underlying assumption of CF approach is that those who agreed in the past tend to agree again in the future. The recommendation scheme described in this paper falls into the CF category.

2.1 Collaborative Filtering

The CF field has enjoyed a surge of interest since October 2006, when the Netflix Prize [10] competition commenced. Neighborhood-based CF and latent factor-based CF are two primary areas of CF techniques. Neighborhood-based CF is most effective at detecting very localized relationships, while latent factor-based CF is generally effective at utilizing all rating information of a user as shown in paper [4]. We compare our algorithm with both neighborhood based CF and latent factor based CF.

Denote the number of users as u_0 and number of items as i_0. A rating $r_{u,i}$ indicates the preference by user u of item i, where high values mean stronger preference. We distinguish predicted ratings from known ones, by using notation $\hat{r}_{u,i}$ for the predicted value of $r_{u,i}$.

For user-based nearest neighbor method,

$$\hat{r}_{u,i} = \bar{r}_u + \frac{\sum_{v \in N_u} sim(u,v)(r_{v,i} - \bar{r}_v)}{\sum_{v \in N_u} sim(u,v)},$$

where $\bar{r}_u$ is the average rating of user u and N_u are the selected neighbors of user u. We use Pearson Correlation coefficient to measure two users' similarity. Pearson Correlation between user u and user v is defined as:

$$sim(u,v) = \frac{\sum_{i \in I_C} (r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum_{i \in I_C} (r_{ui} - \bar{r}_u)^2 \sum_{i \in I_C} (r_{vi} - \bar{r}_v)^2}}, \quad (1)$$

where I_C is set of common items rated by u and v, r_{ui} and r_{vi} are their ratings for movie $i \in I_C$. We denote this method as K-Nearest Neighbor (**KNN**).

Some of the most successful realizations of latent factor models are based on matrix factorization. Matrix factorization models map both users and items to a joint latent factor space of dimensionality f, such that user-item interactions are modeled as inner products in latent factor space. Accordingly, each item i is associated with a vector $p_i \in \mathbb{R}^f$, and each user u is associated with a vector $q_u \in \mathbb{R}^f$. The dot product ${p_i}^T q_u$ captures the interaction between user u and item ith user's overall interest in the item's characteristics. The predicted rating is modeled as

$$\hat{r}_{u,i} = r_m + {p_i}^T q_u, \quad (2)$$

where p_i, $q_u \in \mathbb{R}^f$, and $r_m \in \mathbb{R}$ is a tunable parameter in the model. To learn latent factors p_i and q_u, we resort to optimizing the square error:

$$\frac{1}{2} \sum_{(u,i)\ obs.} (r_{u,i} - \hat{r}_{u,i})^2 + \frac{\lambda}{2}(\|p_i\|_F^2 + \|q_u\|_F^2). \quad (3)$$

Here, (u,i) $obs.$ is the set of the (u,i) pairs for which $r_{u,i}$ is known(the training set). To prevent overfitting, we append a regularization term. $\lambda > 0$ is a regularization parameter. We use the Frobenius norm, denoted by $\|.\|_F$, to regularize the learned item and user latent factors. Note that normally the number of features $f \ll min(u_0, i_0)$. We set $f = 10$ in our experiments. We denote this model as Singular Value Decomposition(**SVD**).

Basically, Bayesian inference-based Recommender System and SVD RS are two different kinds of systems. Bayesian inference-based RS is a distributed system and users do not need to worry about their rating privacy because they only share history ratings with his/her direct friends. While in centralized RS, user's privacy is a big concern. Paper [20] demonstrates an algorithm to link IMDB [15] user to Netflix record. While average Netflix subscriber may not care about his movie rating history, but someone does. That paper shows that it is possible to learn sensitive nonpublic information about a person from his or her movie viewing history. Generally speaking, Bayesian inference-based RS only uses localized information, it provides user with good protection of privacy and it is more scalable. In contrast, SVD RS is good at prediction accuracy and recommendation coverage. Furthermore, in commercial RS, users update their rating profiles all the time. Our Bayesian inference-based RS can integrate newly created and updated user rating profile effectively. However, SVD needs to recalculate all user and item latent factors whenever it wants to integrate newly created user rating profiles.

2.2 Trust-Based Recommendation

Trust has recently been identified as an effective means to utilize social network to improve the recommendation quality. Empirical studies in [22], [23] showed the correlation between trust and user similarity. Various techniques have been proposed to incorporate trust into CF approaches [21], [24], [25], [26], [27], [28], [30], [33]. Typically, the rating similarity between friends is quantified by a numerical value, with larger value indicating higher level of trust. Then, a recommendation is calculated for a user as a function of the ratings and the associated trust values of his friends. Different from trust-based recommendation, we use conditional probability distributions to capture the similarity between users. Probability distributions carry richer information than trust values.

Authors of [32] proposed a belief propagation-based probabilistic algorithm for the design and evaluation of recommender systems. Users' rating information is represented as a bipartite graph over which the belief is propagated. Jamali and Ester [33] explore a model-based approach for recommendation in social networks using matrix factorization techniques. Different from [32] and [33], our approach is a fully distributed algorithm with friends' recommendations propagating on the social network directly.

Finally, Bayesian approaches have been employed in the past in recommendation systems [6], and [16], [17], [18], [19]. A preliminary work on Bayesian inference based recommendation in online social networks is studied in paper [6]. In particular, [16] forms a Bayesian network with a node corresponding to an item. The states of nodes correspond to the possible rating values. Learning algorithm searches over various model structures and converges to the best one. In our design, a node in Bayesian network is a user. The structure of Bayesian network is governed by the underlying social network within which similar users are connected to each other. Furthermore, our recommendation is carried out in a distributed fashion, thus more scalable than centralized recommendations.

3 Bayesian Inference Based Recommendation

3.1 Motivating Example

We develop distributed recommendation algorithms for social networks. As shown in Fig. 1, users are connected in a social network by running social network agents on their own devices, e.g., PCs, mobile phones, set-top boxes, etc.

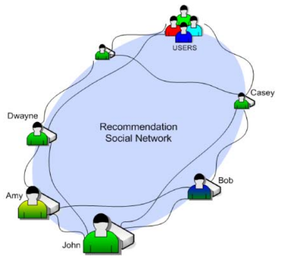

Fig. 1. Recommendation social network.

After watching a movie, a user may choose to publish his recommendation/opinion to his friends. A user can also query his friends regarding the ratings of a specific movie. For instance, in Fig. 1, John is a friend of Amy, Bob, Casey, and Dwayne. Suppose John is thinking about whether he should watch a new release. He issues a query to his friends. It turns out that Amy, Bob, and Casey have watched the movie, and return their ratings of five stars, three stars, and five stars, to John. Based on the returned ratings, John's local recommendation agent computes a recommendation rating that most likely reflects John's interest, and recommends the movie to John if the recommendation score is high.

The key problem in this example is how to estimate John's rating with low estimation error. One naive approach is to recommend to John the average of his friends' ratings. The underlying assumption of this approach is that John's friends' ratings are equally important to John. In reality, John's movie taste maybe "closer" to that of Bob than to Amy and Casey. Intuitively, Bob's rating should carry more weight when estimating John's rating. One can introduce "trust value" between friends and use the trust-weighted sum as recommendation [21], [24]. However, it is challenging to represent the rating closeness between friends using one numerical value. A more refined approach is to look into the pairwise movie rating correlation between John and his friends in the past, and infer the "most probable" rating of John for the current movie. More specifically, if John and Amy have watched and rated a common set of movies before, the rating correlation between them can be measured by a set of conditional probabilities $P(R_J|R_A)$ and $P(R_A|R_J)$, where R_J and R_A are the random variables representing the ratings of John and Amy, respectively. If Amy gives a score r_A for the new release, based on the rating history, the most probable rating of John for the movie can be estimated as

$$\hat{R}_J|_{R_A=r_A} \triangleq \mathop{\mathbf{argmax}}_r P(R_J = r|R_A = r_A).$$

Similarly, based on Bob and Casey's ratings, we can calculate two more estimates of John's rating $\hat{R}_J|_{R_B=r_B}$ and $\hat{R}_J|_{R_C=r_C}$. At the end, we need to generate one recommendation for John by reconciling three estimates based on the marginal conditional distributions on individual friend links. Ideally, we want to estimate the most probable rating of John conditional on the joint ratings of his friends:

$$\hat{R}_J|_{\{r_A,r_B,r_C\}} \triangleq \mathop{\mathbf{argmax}}_r P(R_J = r|r_A, r_B, r_C),$$

where we introduce the abbreviation $r_{\{x\}}$ for the event $R_{\{x\}} = r_{\{x\}}, x = A, B, C$. Unfortunately, in practice, it is hard to estimate the joint conditional distribution between John and all his friends. We instead resort to Bayesian network to reconstruct the joint conditional distribution based on the marginal conditional distributions between John and each of his friends. More specifically, we construct a one-level Bayesian tree between John and his friends as following: 1) John is the root of the tree; 2) each of John's friend is a direct child of John in the tree. Following the definition of Bayesian network, we essentially assume John's friends' ratings are independent with each other conditional on John's rating: $\{R_A \perp R_B \perp R_C | R_J\}$. Consequently, we assume the rating discrepancies between John and his friends are independent. Under this assumption, we have

$$P(r_A, r_B, r_C | r_J) = P(r_A|r_J)P(r_B|r_J)P(r_C|r_J).$$

Following the Bayesian rule, we can calculate the conditional probability of John's rating based on his friends' ratings as

$$P(r_J | r_A, r_B, r_C) = \frac{P(r_A, r_B, r_C | r_J)P(r_J)}{\sum_r P(r_A, r_B, r_C | R_J = r)P(R_J = r)}.$$

3.2 Recommendation System Design

In more general settings, when a user wants a recommendation rating for a movie, it may happen that none of his direct friends has watched/rated the movie. To increase the chance of recommendation, we allow a user to propagate his query through the social network and collect ratings from indirect friends who are several social hops away. Based on the collected ratings, we construct a multilevel Bayesian network to estimate the most probable rating for the querying user.

3.2.1 Framework

More specifically, our distributed recommendation system works in the following framework.

- Users are connected in a social network $G = (V, E)$, where V is the set of users, E is the set of friendship links. Each user shares his ratings with his direct friends;
- Each pair of friends $(u, v) \in E$ measure their *rating similarity* by a set of conditional distributions $P(u|v)$ and $P(v|u)$, each of which is one user's rating distribution given the other user's rating;
- A user sends out a rating query for a movie to his direct friends in the social network G. Upon receiving a query for a movie, a user returns its rating if he has rated the movie before; otherwise it relays the query to its friends. To limit the range of query flooding, a query will be dropped after a predefined number of hops.
- Recommendation agent calculates a score for the querying user using a Bayesian network.

3.2.2 Recommendation Propagation Tree

Let $S \in V$ be the user originating the query for a movie. Let $\mathcal{L} = \{L_i \in V, i = 1, 2, \ldots, k\}$ be the indexed set of direct and indirect friends of S who respond to the query. Social network normally has rich connectivity. To avoid redundant query responses, we generate an unique id for each query. Each user in $\mathcal{L}$ only responds to a query when he receives the query for the first time. We further assume that a query response will be returned to S along the reverse path of the query propagation path. As a result, the union of all query response paths is the shortest path tree from all recommenders in $\mathcal{L}$ to the common root S.

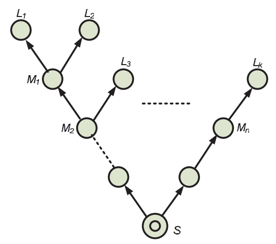

Fig. 2. Recommendation propagation tree as a Bayesian network.

Fig. 2 illustrates an example of a recommendation propagation tree with recommenders in $\mathcal{L}$ as leaves and S as the root. The height of the tree is bounded by the scope of the query flooding. There are three types of users in a recommendation tree: query initiator S, recommenders $\{L_i\}$, and intermediate users $\{M_i\}$. A recommender, L_i, has a recommendation rating of r_i for the queried movie, and transmits his rating to his parent user. An intermediate user sits on the path from responding users to the querying initiator. He collects the recommendation information from his children, aggregates the information, and relays the aggregated information to his parent. Finally, given structure of the recommendation propagation tree and ratings on all leaf users, the querying initiator computes the final recommendation rating.

3.2.3 Bayesian Network

To work with ratings from direct and indirect friends, we resort to Bayesian network to calculate the most probable rating at the root. We construct a Bayesian network out of the recommendation propagation tree. Each user takes its next-hop node on its shortest path to the root as its only parent in the Bayesian network. In other words, we assume that a user's rating is independent of the ratings of nondescendant users in the propagation tree conditional on its parent's rating. Intuitively speaking, any user in the recommendation propagation tree has the strongest rating correlation with his parent among all his nondescendant users. His rating dependence with nondescendant users is "blanketed" by his parent. Using the Bayesian network, one can calculate the conditional probability of $Pr(R_S = s | R_{L_i} = r_i, 1 \leq i \leq K)$ based on the recommendation rating probability distribution and the marginal conditional probability distributions on all friend links in the propagation tree.

For any node m in the propagation tree, we define C_m as the set of its direct children. We further define D_m as the set of recommenders in the subtree rooted at m:

$$D_m \triangleq \{L_i \in \mathcal{L} : L_i \text{ is in the subtree rooted at } m\}.$$

Recommendations generated by users in D_m are relayed by node m to the root. Obviously, we have $D_S = \mathcal{L}$. Due to the query forwarding protocol, if m itself is a recommender, it will not forward a query further to its friends, and will be a leaf node in the propagation tree: $D_m = \{m\}$, if $m \in \mathcal{L}$.

We define the probabilistic event that recommender L_i gives rating r_i as $\Phi(L_i) \triangleq \{R_{L_i} = r_i\}$. Then, the event of the joint ratings of all recommenders under m can be composed as

$$\Psi_m \triangleq \{R_{L_i} = r_i, L_i \in D_m\} = \bigcap_{L_i \in D_m} \Phi(L_i).$$

Our goal is to estimate $P(R_S = s|\Psi_S)$. Given the rating range of $[1, N]$, following the Bayesian rule, we have

$$P(R_S = s|\Psi_S) = \frac{P(\Psi_S|R_S = s)P(R_S = s)}{\sum_{r=1}^{N} P(\Psi_S|R_S = r)P(R_S = r)}. \tag{4}$$

Since $D_S = \bigcup_{c \in C_S} D_c$ and $\Psi_S = \bigcap_{c \in C_S} \Psi_c$, we have

$$P(\Psi_S|R_S = s) = P\left(\bigcap_{c \in C_S} \Psi_c|R_S = s\right) = \prod_{c \in C_S} P(\Psi_c|R_S = s), \tag{5}$$

where the last equivalence is established by the conditional independence in the Bayesian network. If a child c is a recommender, i.e., $c \in \mathcal{L}$, then $D_c = \{c\}$, and $P(\Psi_c|R_S = s) = P(R_c = r_c|R_S = s)$, which can be directly obtained from the conditional probability between c and its parent S. If c is a relay node, i.e., $c \notin \mathcal{L}$, then we have

$$\begin{aligned} P(\Psi_c|R_S = s) &= \sum_{i=1}^{N} P(\Psi_c \cap \{R_c = i\}|R_S = s) \\ &= \sum_{i=1}^{N} P(\Psi_c|R_c = i)P(R_c = i|R_S = s), \end{aligned} \tag{6}$$

where the last equivalence is established by the independence of the ratings of c's descendants in the Bayesian network with S conditional on c's rating. The last term $P(R_c = i|R_S = s)$ in the summation is readily available from the marginal distributions between c and S. The first term $P(\Psi_c|R_c = i)$ can be recursively calculated following the similar process by going up one level in the Bayesian network.

3.2.4 Recommendation Calculation

Based on the Bayesian network, we are ready to calculate one recommendation rating for the query initiator S. We propose two recommendation calculation schemes. The first one is the **most probable recommendation**, $\hat{S}^{MP}$, that maximizes the joint conditional probability

$$\hat{S}^{MP} \triangleq \mathop{\mathbf{argmax}}_{1 \le s \le N} P(R_S = s|\Psi_S). \tag{7}$$

The second one is the **Bayes Mean Square Error estimator**, $\hat{S}^{MSE}$, that minimizes mean square error

$$\hat{S}^{MSE} \triangleq E[S|\Psi_S] = \sum_{s=1}^{N} sP(R_S = s|\Psi_S). \tag{8}$$

Note that, $\hat{S}^{MP}$ calculated by (7) is always an integer value, but $\hat{S}^{MSE}$ calculated by (8) can be fractional.

4 Distributed Protocol Design

In this section, we present a distributed protocol to implement the proposed Bayesian-inference recommendation algorithm and automatically calculate recommendations for users in online social networks. The design consists of three key components:

- *User-recommendation interface:* It allows users to input their recommendations after watching movies, and query the recommendation engine for unseen movies.
- *Learning module:* It communicates with direct friends, exchanges recommendations, and constructs probability distributions required by the Bayesian network. The learning module estimates the local user's recommendation rating distribution, and the conditional distributions between the local user and each of his friends.
- *Recommendation engine:* It computes recommendations for querying users based on the available ratings inside the social network. A query is propagated to friends, who may relay the query to their own friends. Each available recommendation propagates back to the query initiator in the reverse direction of the query propagation path. Bayesian network-based inference is conducted in a distributed fashion to compute a recommendation score.

Below we describe the details of the learning module and the recommendation engine.

4.1 Learning Friends

Individual users are required to estimate their own rating probability distribution, and their direct friends' recommendation rating probabilities conditioned on their own ratings. Accordingly, learning module maintains a database that stores the following counters: $\{n_i\}$ and $\{n^T_{j,i}\}$, where n_i is the number of occurrences of rating i made by the local user, and $n^T_{j,i}$ is the number of occurrences that a friend T gives a rating j to a movie while the local user gives a rating i to the same movie, where $i, j = 1, 2, \ldots, N$. Empirical distributions can be estimated as $P(i) = n_i / \sum_{i=1}^{N} n_i$, and $P^T(j|i) = n^T_{j,i} / \sum_{j=1}^{N} n^T_{j,i}$. Note that the frequency-based estimation is maximum likelihood estimator (MLE).

All counters are initialized to be zero. When user S makes a recommendation of $\{movie\_id, s\}$ through the recommendation interface, the learning module stores the recommendation pair into a recommendation table. The counter n_s is increased by one. The learning module sends $\{S, movie\_id, s\}$ to all direct friends. When friend T receives the message, its learning module conducts a lookup service at its recommendation table, and determines if T has already rated the same movie. If T never rated the movie, the message is discarded without further action. If T rated the movie with rating t, the learning module of T increases the counter $n^S_{s,t}$ by one. In addition, T sends back a message of $\{T, movie\_id, t\}$ to S. This message allows user S to update its conditional counter accordingly.

4.1.1 Coping with Cold Start and Rating Sparseness

Frequency-based estimation is not accurate when the number of common movies watched by a pair of friends is small. Social network offers a new venue that employs users inputs to tackle the above issues. Given a rating scale of $[1, N]$, for each friend, a user chooses N confidence values $\{c_i, 0 \leq i \leq N-1\}$, where c_i represents his confidence that the absolute rating difference between him and the friend is i. The confidence level is quantified into one to 10: the higher the number, the higher the confidence. Suppose user X has a neighbor user Y. Denote by x and y user X and user Y's ratings. To alleviate cold start problem, we estimate $p(y = h|x = l)$, $l, h = 1, 2, \ldots, N$, as follows:

$$p(y = h|x = l) = \begin{cases} \dfrac{c_0}{\sum_{i=1}^{N} c_{|l-i|}/2 + c_0}, & \text{if } h = l \\ \dfrac{c_{|l-h|/2}}{\sum_{i=1}^{N} c_{|l-i|}/2 + c_0}, & \text{if } h \neq l. \end{cases} \tag{9}$$

Except for c_0, confidence levels are divided by two in computing probability because offset by i stars can either be greater than the conditional value l by i stars, or smaller than the conditional value l by i stars.

Due to space limit, we cannot present the reason why we estimate $p(y = h|x = l)$ using (9). The detailed way of coping with cold start and rating sparseness and evaluation result are available in our technical report.[1]

4.2 Distributed Recommendation Engine

Query generation: Assume user S queries the recommendation engine. The recommendation engine sends the query message to neighbors over the social network. A query message is a 3-tuple of $[sequence\_id, movie\_id, TTL]$. $sequence\_id$ is a unique id identifying this query message. For instance, the id can be created by hashing the $movie\_id$ together with the querying user's own social network id. $movie\_id$ identifies the movie for which the recommendation is sought after. TTL, or time-to-live, defines the search scope of the query. A query is dropped when its TTL goes to zero.

Query propagation: Upon receiving a query, a user first checks if he has already rated the movie. If yes, his movie rating is returned back to the user from which the query message is received. The query message is dropped without further forwarding. If the receiving user has not rated the movie, and the query's TTL value is positive, the user decrement the query's TTL and forwards the query to his neighbors, except the one from which the query is received. Finally, if a query's TTL is zero, it will be dropped. The receiving user sends a *DROP* message back to the user from which the query is received, indicating that the query has been dropped without recommendation. A social network may have loops, and a user may receive the same query from multiple neighbors. To simplify the inference, a user only responds to the neighbor from which a query is received for the first time, and sends *DROP* messages to all other neighbors.

Response generation: After the query process, a recommendation tree is constructed. Query responses will be propagated back to the querying user along the tree. The goal is to allow the root (querying user) to calculate the conditional probabilities defined in (4), (5), and (6). From the root point of view, the conditional probabilities in (5) and (6) are calculated in a recursive way. It is implemented as an iterative algorithm in the Bayesian network, starting from the leaf nodes. Specifically, all leaf nodes respond to the query by sending their ratings to their parents. For an intermediate user m, following the definitions in Section 3.2.3, he is responsible for sending a set of conditional rating probabilities, $P(\Psi_m|R_m = r), 1 \leq r \leq N$, to his parent. To do this, m waits for the responses after forwarding the query to his neighbors. If a neighbor returns a *DROP* message, no action is needed. All neighbors returning a rating or conditional rating probabilities are m's children in the propagation tree. Similar to (5) and (6), we have $P(\Psi_m|R_m = r) = \Pi_{c \in C_m} P(\Psi_c|R_m = r)$. If m's children $c \in C_m$ is a recommender, i.e., $c \in \mathcal{L}$, then $P(\Psi_c|R_m = r) = P(R_c = r_c|R_m = r)$; otherwise, $P(\Psi_c|R_m = r) = \sum_{i=1}^{N} P(\Psi_c \mid R_c = i)P(R_c = i|R_m = r)$. After m collects ratings $R_c = r_c$ from his recommending children and conditional probabilities $P(\Psi_c|R_c = i)$ from his relaying children, he can compute his joint conditional probabilities $P(\Psi_m|R_m = r)$ based on the marginal conditional probabilities $\{P(R_c = r_c|R_m = r), \forall c \in C_m\}$ obtained by the learning module. The computed conditional probabilities will be sent to m's parent as the query response.

Recommendation calculation: Once the querying user S receives responses from all his children, he will calculate $P(R_S = s|\Psi_S)$ following (4), (5) and (6). Finally, the recommendation engine on S recommends to the user a most probable rating $\hat{S}^{MP}$, or a Bayes Mean Square Error rating $\hat{S}^{MSE}$, according to (7) and (8), respectively.

5 Evaluation

In this section, the performance of the Bayesian inference-based recommendation is evaluated using two data sets: Epinions data set [9] and MovieLens data set [31]. Epinions data set contains both item rating information and trust network topology information, which allows us to compare the performance of Bayesian inference-based recommendation with that of trust-based recommendation. MovieLens data set is denser than Epinions data set, and the ratings are time stamped. Denser ratings enable better learning of conditional probability between neighboring users, while the time stamped ratings allow us to simulate the temporal evolution of Bayesian inference based recommendation system, and evaluate the impact of MAP Prior distribution to both the recommendation quality and quantity.

5.1 Evaluation Using Epinions Data Set

Epinions [9] is a consumer opinion site where users can review and rate items, including movies, cars, books, software, etc. The data set consists of 49,290 users, 664,824 reviews/ratings, and 139,738 different rated items. The data set was collected by Paolo Massa in a 5-week crawl of the Epinions website during November and December in 2003. The ratings are on the numerical five-star scale, where one and two stars represent negative ratings, three stars represent ambivalence ratings, and four, five stars represent

1. http://wanlab.poly.edu/xiwang/doc/SN-recomm-tpds-TR.pdf.

positive ratings. Users also express their *trust* to a set of users whose reviews/ratings are found to be valuable. The total number of issued trust is 487,181. In a trust-based recommender, a user receives recommendation only if it is connected with other users via the trust relationship. After filtering out isolated users, the data set has 33,960 users and 589,714 ratings. We use the social network topology as the trust network topology in the study.

We evaluate the recommendation systems using the *leave-one-out method*, where one rating is taken out of the data set and then compared with the recommendation rating obtained using the rest of the data. Such procedure is iterated through the entire data set. Leave-one-out method is commonly used in comparing two recommendation systems.

We compare the performance of Bayesian inference based recommendation with TidalTrust [21] and MoleTrust [29], two representative trust based recommendation systems. TidalTrust and MoleTrust differ in their strategies in computing trust between two nonadjacent users. Six *views* are formed for the purpose of detailed comparison: *cold start users*, users who provided less than five ratings; *heavy raters*, users who provided more than 10 ratings; *opinionated users*, users who provided more than four ratings and whose standard deviation is greater than 1.5; *black sheep users*, users who provided more than four ratings, and the difference between their ratings on a specific item and the average rating of that item is greater than one; *niche items*, items receiving less than five ratings; and *controversial items*, items receiving ratings with standard deviation greater than 1.5. Since the neighbors in social network have shown trust to each other, we use the confidence levels of $\{3, 1.5, 1, 1, 1\}$ for the Prior distribution, i.e., neighbors are more likely to have similar ratings.

We use the most probable recommendation rating and the *Mean Absolute Error* (MAE) to measure the recommendation quality. Recommendation coverage, defined to be the percentage of queries that actually obtain the recommendation, measures the recommendation quantity. The results of MAE and recommendation coverage for the KNN, Bayesian inference based recommendation, MoleTrust, and TidalTrust [21] are presented in Table 1. For Bayesian inference-based recommendation and two trust-based recommendations, the following three scenarios with different query scopes are considered: 1-hop (TTL = 1), 2-hop (TTL = 2), and 3-hop (TTL = 3). The results of Moletrust and KNN are cited from [29]. Since the social network topology is the same as the trust network topology, TidalTrust, MoleTrust, and Bayesian inference-based recommendation have the same recommendation coverage for the same query range.

Table 1 presents the results for six views, one row for each view. Within each view, in the KNN column, there are two subrows, with the top row be MAE and the bottom row be coverage. Results for Bayesian inference-based recommendation and two trust-based recommendations with the same query range are listed in the same column, in which the bottom subrow is their common coverage, the first subrow (in Italic font) is the MAE for Bayesian inference, the second and third subrows are the MAEs for MoleTrust and TidalTrust, respectively.

TABLE 1
MAE and Recommendation Coverage Comparison of KNN, Bayesian Inference-Based Recommendation, Moletrust, and TidalTrust

Mean Absolute Error				
views	KNN	1-hop	2-hop	3-hop
		0.747	*0.786*	*0.791*
All	0.843	0.832	0.846	0.829
		0.842	0.826	0.833
	51.28%	28.33%	60.47%	74.37%
		0.761	*0.893*	*0.878*
Cold users	1.094	0.674	0.833	0.854
		0.761	0.875	0.878
	3.22%	11.05%	25.02%	41.74%
		0.745	*0.780*	*0.785*
Heavy raters	0.850	0.873	0.869	0.846
		0.843	0.823	0.828
	57.45%	30.85%	64.82%	77.81%
		1.339	*1.484*	*1.518*
Contr. items	1.515	1.425	1.618	1.687
		1.420	1.610	1.701
	45.42%	25.09%	60.64%	81.80%
		0.460	*0.633*	*0.699*
Niche items	0.822	0.734	0.806	0.828
		0.527	0.710	0.771
	12.18%	8.32%	24.32%	30.43%
		1.114	*1.205*	*1.202*
Opin. users	1.200	1.020	1.102	1.096
		1.044	1.091	1.092
	50%	23.32%	57.31%	74.24%
		1.137	*1.224*	*1.237*
Black sheep	1.235	1.152	1.238	1.242
		1.185	1.228	1.237
	55.74%	23.66%	59.21%	76.32%

KNN is not effective in providing accurate recommendations due to the sparsity of the rating matrix, which is defined to be the percentage of unrated items in the rating matrix of users versus items. Bayesian inference-based recommendation and two trust-based recommendations outperform the KNN in terms of MAE for most of the views. KNN offers better recommendation coverage than trust-based recommendations and social-network-based recommendation when the query range is one. However, as the query range increases to two or three, the recommendation coverages of the trust-based recommendation and the social network-based recommendation become larger than that of KNN.

Next, the recommendation accuracy of Bayesian inference based recommendation, MoleTrust, and TidalTrust are compared. These three approaches have the same recommendation coverage. In terms of overall MAE, Bayesian inference-based recommendation outperforms MoleTrust and TidalTrust with a margin as large as 10 percent. Except for *cold users* and *opinioned users*, our Bayesian approach consistently provides more accurate recommendations than MoleTrust and TidalTrust. For instance, the MAE of 1-hop query of *heavy raters* for Bayesian inference is 15 percent smaller than that of MoleTrust, and 12 percent smaller than that of TidalTrust. Trust-based recommendations perform better for *cold users* and *opinionated users*. In the case of *cold users*, the estimation of user rating probability distribution are not accurate due to small number of samples. *Opinionated users* exhibit large deviation in ratings. Our Gaussian-shape prior does not correctly reflect the *opinionated users'* rating habits, thus leads to larger MAE.

TABLE 2
MAE and Recommendation Coverage Comparison of SVD and Bayesian Inference-Based Recommendation

Mean Absolute Error					
views	SVD	1-hop	2-hop	3-hop	4-hop
All	0.924	0.782	0.893	0.894	0.888
	100%	9.71%	37.13%	61.56%	72.40%
Cold users	1.038	0.636	1.061	1.028	0.97
	100%	2.05%	13.34%	39.55%	59.98%
Heavy raters	0.887	0.786	0.88	0.875	0.872
	100%	11.72%	42.93%	65.82%	74.15%
Contr. items	1.527	1.532	1.746	1.644	1.647
	100%	7.73%	27.01%	48.14%	59.28%
Niche items	0.9	0.482	0.573	0.639	0.686
	100%	8.43%	12.46%	18.27%	22.76%
Opin. users	1.274	1.029	1.398	1.396	1.414
	100%	7.73%	35.21%	63.43%	75.44%
Black sheep	1.271	1.026	1.257	1.287	1.304
	100%	7.89%	36.08%	65.82%	78.22%

Furthermore, we compare Bayesian inference-based algorithm with SVD method. The result is shown in Table 2. In SVD, we tune parameters λ in (3), and r_m in (2) for the best MAE result on the test set. We get the best MAE result when setting $r_m = 4.0$ and $\lambda = 0.2$. Note that the optimal value of r_m is the same as the mean of all training ratings. For Bayesian inference based recommendation, we consider four scenarios: 1-hop, 2-hop, 3-hop, and 4-hop. We can see from Table 2 that, SVD is good at providing recommendation to all possible ratings, which has coverage of 100 percent. However, Bayesian-based recommendation can provide more accurate recommendation when sacrificing coverage. Note that Bayesian inference-based RS and SVD RS are two different kinds of systems. Bayesian inference-based RS is a distributed system and can protect user's privacy naturally which SVD cannot. In contrast, SVD RS is good at prediction accuracy and recommendation coverage.

5.2 Evaluation Using MovieLens Data Set

MovieLens data set [31] contains about one million ratings for 3,952 movies and 6,040 users. The same data set has been widely used in other studies. The number of ratings, movies, and users are manageable yet rich enough to evaluate the algorithm. The ratings are on the numerical five-star scale. Below we construct a synthesized social network topology which observes the *homophily* principle [7].

The data set is divided into two parts: the first 70 percent ratings are used for the training purpose—to construct the social network and to estimate the rating probability and conditional probability distributions at individual users. The remaining 30 percent ratings are used for testing/evaluation.

We use the Pearson correlation coefficient as in (1) to measure two users' similarity. Each user randomly picks 500 users out of the entire user population and calculates the Pearson coefficients with each of them. The 10 users with the highest coefficients are selected as his direct friends. Since a user is also selected by other users as friend, the average number of friends, or the degree of a user in the social network, is 20. We also tried to construct synthesized social networks with higher degree. In general, the more friends a user has, the better the recommendation accuracy and recommendation coverage are. Below we only present the results with the average number of friends equals to 20.

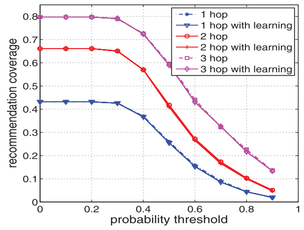

Fig. 3. Recommendation coverage versus probability threshold.

5.2.1 *Impact of Probability Threshold, Dynamic Learning, and Query Range*

Each recommendation calculated by the recommendation engine as in (4) is associated a probability value. To filter out low probability recommendations, we set a probability threshold and only the recommendations with associated probability greater than the threshold are presented to the querying user.

Figs. 3 and 4 depict the recommendation coverage and MAE versus the probability threshold with different query range (TTL), and with and without *dynamic learning*, to be defined below. We first examine the impact of probability threshold. Both recommendation coverage and MAE decreases as the probability threshold increases. Intuitively, as the probability threshold increases, more and more Most Probable recommendations with low associated probability are filtered out. In addition, recommendations with low probability are more likely to be inaccurate, thus larger MAE. The probability threshold can serve as a knob to trade off the recommendation accuracy against the amount of recommendations.

Next, we examine the impact of dynamic learning. With dynamic learning, the learning module is on. The users continuously update the probability distribution throughout the simulation. Without dynamic learning, the learning module is off. The same probability distribution, estimated using the training data set at the beginning, is used throughout the simulation. With the same TTL, recommendation coverage is roughly the same with and without the dynamic learning. The MAE is lower with dynamic learning. Dynamic learning refines the conditional probability distribution over time, thus leads to better recommendation accuracy.

Finally, we examine the impact of query range. The recommendation coverage increases as the query range increases. A larger query range gives a querier better chance to encounter the users who have already rated the movie. Nevertheless, the recommendation quality decreases as the query range increases. This is because socially closer friends is more similar with each other.

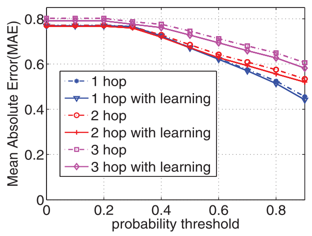

Fig. 4. MAE versus probability threshold.

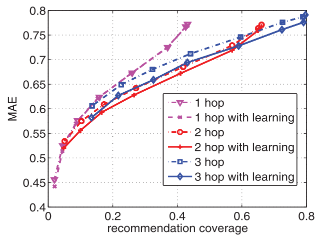

Fig. 5. Comparison of Bayesian inference-based recommender versus CF recommender.

5.2.2 Comparison with Collaborative Filtering

First, we compare the performance of Bayesian inference-based recommendation with KNN model. We vary the value of K and plot the curve of recommendation coverage versus MAE for KNN. The result is shown in Fig. 6. According to Fig. 5, three hop with learning can achieve MAE of 0.791 with coverage of 79.7 percent when setting probability threshold to 0. From Fig. 6, we can see that KNN can achieve best MAE of 0.765 with coverage of 81.9 percent when neighborhood size $K = 1{,}700$. From our three hop with learning result, we can achieve same MAE as best KNN MAE with coverage of 73 percent when setting probability threshold to 0.37. If the probability threshold is set higher, our system can provide even more accurate recommendations.

Second, we compare Bayesain inference-based recommendation with SVD method. For SVD method, the best MAE comes when setting $\lambda = 0.05$, $r_m = 3.6$. The best MAE is 0.755. Note that $r_m = 3.6$ is equal to the mean of all training ratings. Also note that recommendation coverage for SVD method is 100 percent. In Bayesian-based recommendation, three hop with learning case can achieve the same MAE as SVD with coverage of 68 percent when set probability threshold to 0.42. Our system can achieve better result if we set probability threshold larger.

6 Conclusions and Future Work

In this paper, we propose a Bayesian-inference-based recommendation system that leverages the embedded social structure inside a social network to provide accurate and personalized recommendations. Using conditional probability distributions to measure the rating similarity between friends, a Bayesian network inference-based framework is designed to calculate the most probable recommendation. The experiments show that the accuracy of Bayesian inference-based recommendation is better or comparable to that of centralized CF-based approaches and trust-based approaches, and can flexibly tradeoff the amount of recommendations against the recommendation accuracy. Furthermore, social network affords us a unique opportunity to tackle the cold start issue and the rating sparseness issue that have baffled the neighborhood-based CF approaches. We have implemented the Bayesian inference-based recommendation as a Facebook application and will release it to the public soon.

Future work can proceed in several directions. Due to the lack of data set, the social network topology is synthesized in our evaluation. We would like to test the algorithm using traces containing both social structure and

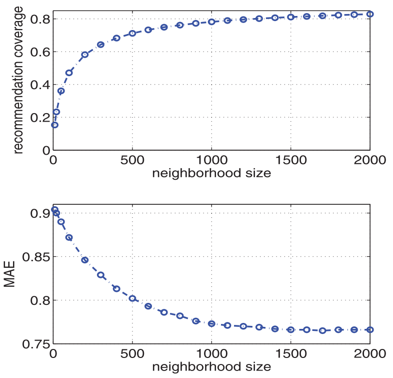

Fig. 6. MAE and coverage result of KNN method. The upper figure is recommendation coverage versus CF neighborhood size. The bottom figure is MAE versus CF neighborhood size.

recommendation information. Second, while the initial results show that the MAP-based Prior is effective for new users to obtain accurate and sufficient recommendations, how a real human being would utilize such interface is not known and worth further study.

References

[1] G. Adomavicius and A. Tuzhilin, "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions," *IEEE Trans. Knowledge and Data Eng.*, vol. 17, no. 6, pp. 734-749, June 2005.

[2] K. Yu, A. Schwaighofer, V. Tresp, X. Xu, and H. Kriegel, "Probabilistic Memory-Based Collaborative Filtering," *IEEE Trans. Knowledge and Data Eng.*, vol. 16, no. 1, pp. 56-69, Jan. 2004.

[3] J. Herlocker, J. Konstan, L. Terveen, and J. Ridel, "Evaluating Collaborative Filtering Recommender Systems," *ACM Trans. Information Systems*, vol. 22, no. 1, pp. 5-53, Jan. 2004.

[4] Y. Koren, R. Bell, and C. Volinsky, "Matrix Factorization Techniques for Recommendation Systems," *Computer*, vol. 42, no. 8, pp. 30-37, Aug. 2009.

[5] J. Bennet and S. Lanning, "The Netflix Prize," *Proc. KDD Cup and Workshop*, www.netflixprize.com, 2007.

[6] X. Yang, Y. Guo, and Y. Liu, "Bayesian-Inference Based Recommendation in Online Social Networks," *Proc. IEEE INFOCOM '11*, 2011.

[7] M. McPherson, L. Smith-Lovin, and J.M. Cook, "Birds of a Feather: Homophily in Social Networks," *Ann. Rev. Sociology*, vol. 27, no. 1, pp. 415-444, 2001, doi:10.1146/annurev.soc.27.1.415.

[8] YouTube, http://www.youtube.com, 2012.

[9] Epinions, http://www.epinions.com/, 2012.

[10] NetflixPrize, http://www.netflixprize.com/, 2012.

[11] Twitter, http://twitter.com, 2012.

[12] Facebook, http://www.facebook.com, 2012.

[13] Flixster, http://www.flixster.com, 2012.

[14] Flixster, http://www.facebook.com/apps/application.php?id=111849295510739, 2012.

[15] IMDB, http://www.imdb.com, 2012.

[16] J.S. Breese, D. Heckerman, and C. Kadie, "Empirical Analysis of Predictive Algorithms for Collaborative Filtering," technical report, Morgan Kaufmann, pp. 43-52, 1998.

[17] Y.-H. Chien and E.I. George, "A Bayesian Model for Collaborative Filtering," *Proc. Seventh Int'l Workshop Artificial Intelligence and Statistics*, 1999.

[18] A. Ansari, S. Essegaier, and R. Kohli, "Internet Recommendations Systems," *J. Marketing Research*, vol. 37, no. 3, pp. 363-375, Aug. 2000, doi: 10.1509/jmkr.37.3.363.18779.

[19] Y. Zhang and J. Koren, "Efficient Bayesian Hierarchical User Modeling for Recommendation System," *Proc. 30th Ann. In'l ACM SIGIR Conf. Research and Development in Information Retrieval*, pp. 47-54, 2007, doi:10.1145/1277741.1277752.

[20] A. Narayanan and V. Shmatikov, "Robust de-Anonymization of Large Sparse Data Sets," *Proc. IEEE Symp. Security and Privacy*, 2008.

[21] J. Golbeck and J. Hendler, "FilmTrust: Movie Recommendations Using Trust in Web-Based Social Networks," *Proc. IEEE Consumer Comm. and Networking Conf.*, pp. 282-286, Jan. 2006.

[22] C. Ziegler and J. Golbeck, "Investigating Interactions of Trust and Interest Similarity," *Decision Support Systems*, vol. 43, no. 2, pp. 460-475, 2007, doi:http://dx.doi.org/10.1016/j.dss.2006.11.003.

[23] C. Ziegler and G. Lausen, "Analyzing Correlation between Trust and User Similarity in Online Communities," *Proc. Second Int'l Conf. Trust Management*, pp. 251-265, 2004.

[24] P. Massa and B. Bhattacharjee, "Using Trust in Recommender Systems: An Experimental Analysis," *Proc. iTrust In'l Conf.*, pp. 221-235, 2004.

[25] T. DuBois, J. Golbeck, J. Kleint, and A. Srinivasan, "Improving Recommendation Accuracy by Clustering Social Networks with Trust," *Proc. ACM RecSys Workshop Recommender Systems and the Social Web*, Oct. 2009.

[26] J. O'Donovan and B. Smyth, "Trust in Recommender Systems," *Proc. 10th In'l Conf. Intelligent User Interfaces*, 2005.

[27] F. Walter and S. Battiston, and F. Schweitzer, "A Model of a Trust-Based Recommendation System on a Social Network," *Autonomous Agents and Multi-Agent Systems*, vol. 16, no. 1, pp. 1573-7454, Oct. 2007.

[28] K. Sarda, P. Gupta, D. Mukherjee, S. Padhy, and H. Saran, "A Distributed Trust-Based Recommendation System on Social Network," *Proc. IEEE 10th Second Workshop Hot Topics in Web Systems and Technologies*, 2008.

[29] P. Massa and P. Avesani, "Trust-Aware Recommender Systems," *Proc. ACM Conf. Recommender Systems*, 2007.

[30] P. Massa and P. Avesani, "Controversial Users Demand Local Trust Metrics: An Experimental Study on epinions.com Community," *Proc. 25th Am. Assoc. for Artificial Intelligence Conf.*, 2005.

[31] GroupLens, "MovieLens Data Sets," http://www.grouplens.org/node/73#attachments, 2010.

[32] E. Ayday and F. Fekri, "A Belief Propagation Based Recommender System for Online Services," *Proc. ACM Conf. Recommender Systems*, 2010.

[33] M. Jamali and M. Ester, "A Matrix Factorization Technique with Trust Propagation for Recommendation in Social Networks," *Proc. ACM Conf. Recommender Systems*, 2010.

Xiwang Yang received the BS degree from the Department of Electronic Science and Technology, University of Science & Technology of China (USTC), in June 2008, and is currently working toward the PhD degree at the Department of Electrical and Computer Engineering, Polytechnic Institute of New York University, Brooklyn, New York. His research interests include recommender systems and online social networks, focused on developing models and algorithms for explanatory and predictive analysis of large-scale online user behavior. He is a student member of the IEEE.

Yang Guo received the BS and MS degrees from Shanghai Jiao Tong University, and the PhD degree from the University of Massachusetts at Amherst in 2000. He is with the Network Protocols and Systems Department at Bell Labs, Alcatel-Lucent. Prior to the current position, he was a principal scientist at Technicolor/Thomson Corporate Research, Princeton, NJ. His research interests include multimedia systems and networking in general, and cloud computing, peer-to-peer, content distribution, online social networks, and media streaming in particular. He is a member of the IEEE.

Yong Liu received the bachelor's and master's degrees in the field of automatic control from the University of Science and Technology of China, in 1994 and July 1997, respectively, and the PhD degree from Electrical and Computer Engineering Department at the University of Massachusetts, Amherst, in May 2002. He is an associate professor at the Electrical and Computer Engineering Department of the Polytechnic Institute of New York University (NYU-Poly). He joined NYU-Poly as an assistant professor in March, 2005. His research interests include modeling, design, and analysis of communication networks. His current research directions include Peer-to-Peer systems, overlay networks, network measurement, online social networks, and recommender systems. He is the winner of the IEEE Conference on Computer and Communications (INFOCOM) Best Paper Award in 2009, and the IEEE Communications Society Best Paper Award in Multimedia Communications in 2008. He is currently serving as an associate editor for *IEEE/ACM Transactions on Networking*, and *Elsevier Computer Networks Journal*. He is a member of the IEEE and ACM.