Bayesian-inference Based Recommendation in Online Social Networks

Xiwang Yang, Yang Guo, and Yong Liu,

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 more accurate than the traditional Collaborative Filtering (CF) recommendation and the existing trust-based recommendations. It allows the flexible trade-offs 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 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 [7], an improvement of 10% recommendation accuracy is awarded with 1 million dollars. In real life, people often resort to friends in their social networks for advices 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* [4]. 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 [9], twitter [8], and Youtube [5], 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 [9], or a domain-specific recommendation social network, such as Flixster [10] for movie recommendation and Epinions [6] 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 on social networks, such as the Movies App [11] 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 calculate a recommendation rating for the querying user. Within this framework, our paper addresses the following issues:

– We propose to measure the rating similarity between friends in an online social network

- *Xiwang Yang is with the Department of Electrical and Computer Engineering, Polytechnic Institute of NYU, Brooklyn, New York, 11201.*
  E-mail: xyang01@students.poly.edu
- *Yang Guo is with Bell Labs, Alcatel-Lucent, Holmdel, New Jersey. E-mail: Yang.Guo@alcatel-lucent.com*
- *Yong Liu is with the Department of Electrical and Computer Engineering, Polytechnic Institute of NYU, Brooklyn, New York, 11201. E-mail: yongliu@poly.edu*

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 recommendation and the Bayes Mean Square Error (MSE) recommendation for a querying user.
- 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 synthesize social networks to evaluate the proposed algorithm. We show that the Bayesian-inference based recommendation is more accurate than the traditional Collaborative Filtering (CF) recommendation and existing trust-based recommendations. 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 briefly describe the related work in the rest of this section. We present our Bayesian-inference based recommendation algorithm in Section 2. Implementation details of the proposed algorithm in online social networks is described in Section 3. The MAP algorithm is developed in Section 4. We present the evaluation results in Section 5. The paper is concluded with summary and future directions in Section 7.

1.1 Related Work

Recommender systems are usually classified into three categories: content-based recommendations, Collaborative Filtering (CF)-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]–[3]. 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.

Trust has recently been identified as an effective means to utilize social network to improve the recommendation quality. Empirical studies in [17], [18] showed the correlation between trust and user similarity. Various techniques have been proposed to incorporate trust into CF approaches [16], [19]–[23], [25]. 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, and allow us to employ Bayesian network inference to conduct multiple-hop recommendation in online social networks.
[27] proposes 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. [28] explores a model-based approach for recommendation in social networks using matrix factorization techniques. It learns the latent user features from user rating matrix and improves the accuracy of latent user features through latent features propagation over the social network. Different from [27] and [28], 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 [12]–[15]. In particular, [12] 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.

2 Bayesian Inference based Recommendation

2.1 Motivating Example

We develop distributed recommendation algorithms for social networks. As shown in Figure 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. 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 Figure 1, John is a friend of Amy, Bob, Casey, and Dwayne. Suppose John is thinking about if 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 mostly likely

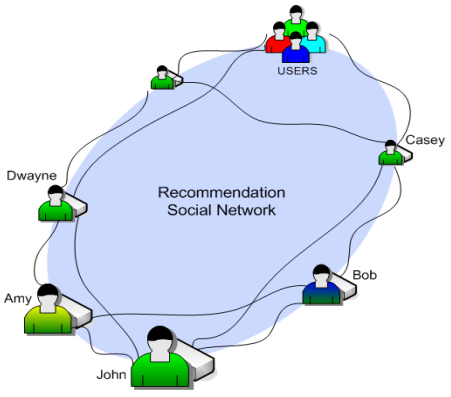

Fig. 1. Recommendation social network

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 [16], [19]. However, it is challenging to represent the rating closeness between friends using one numerical value. And it is difficult to propagate the trust values through a social network when some recommendations come from indirect friends several social hops away. 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

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

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:

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

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: a) John is the root of the tree; b) 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

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

2.2 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 multi-level Bayesian network to estimate the most probable rating for the querying user.

2.2.1 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 pre-defined number of hops.
- Recommendation agent calculates a score for the querying user using a Bayesian network.

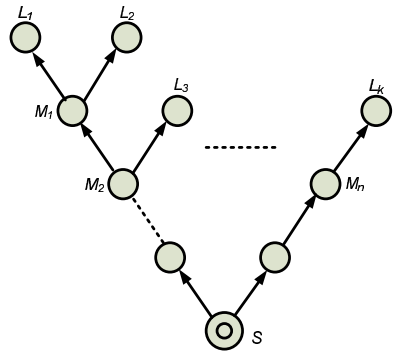

Fig. 2. Recommendation Propagation Tree as a Bayesian network

2.2.2 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.

Figure 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 the structure of the recommendation propagation tree and the ratings on all leaf users, the querying initiator computes the final recommendation rating.

2.2.3 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 non-descendant 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 non-descendant users. His rating dependence with non-descendant 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

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

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

$$P(\Psi_S|R_S = s) = P(\bigcap_{c \in C_S} \Psi_c|R_S = s) = \prod_{c \in C_S} P(\Psi_c|R_S = s), \tag{2}$$

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

$$\begin{aligned} P(\Psi_c|R_S = s) &= \sum_{i=1}^{N} P(\Psi_c \cap \{R_c = i\}|R_S = s) \\ &= \sum_{i=1}^{N} P(\Psi_c|R_c = i)P(R_c = i|R_S = s), \end{aligned} \tag{3}$$

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.

2.2.4 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

$$\hat{S}^{MP} \triangleq \underset{1\leq s\leq N}{\mathbf{argmax}} P(R_S = s|\Psi_S). \quad (4)$$

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

$$\hat{S}^{MSE} \triangleq E[S|\Psi_S] = \sum_{s=1}^{N} sP(R_S = s|\Psi_S). \quad (5)$$

Note that, $\hat{S}^{MP}$ calculated by (4) is always an integer value, but $\hat{S}^{MSE}$ calculated by (5) can be fractional.

3 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.

3.1 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.

3.2 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 (1), (2) and (3). From the root point of view, the conditional probabilities in (2) and (3) 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 2.2.3, 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 (2) and (3), we have $P(\Psi_m|R_m = r) = \prod_{c \in C_m} P(\Psi_c|R_m = r)$. 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, $P(\Psi_c|R_m = r) = \sum_{i=1}^{N} P(\Psi_c|R_c = i)P(R_c = i|R_m = r)$. 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 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 (1), (2) and (3). 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 (4) and (5) respectively.

4 Coping with Cold Start and Rating Sparseness

Learning modules use friends' ratings to commonly watched movies to construct conditional probability distributions. Frequency based approach is used in probability estimation. Even though the frequency based estimate is MLE, it is not accurate when the number of common movies watched by a pair of friends is small. In addition, a newly joined user has not made any rating, thus friends cannot construct history based conditional probabilities against the new user. These rating sparseness issue and cold start issue have baffled the collaborative filtering (CF) based recommendation approaches.

Social network offers a new venue that employs users' inputs to tackle the above issues. In the context of social network, a new user is likely to know his/her friends, and has general feelings/opinions about them. A new user can give a guideline on how the conditional probability distributions should look like. For instance, the new user may indicate to the recommendation engine that with likelihood of eight (in the range from one to ten), his rating is the same as a neighbor, and with likelihood of five, his recommendation rating is probably off by one, so on and so forth. With this information, the recommendation engine can construct informative prior distributions for a new user's neighbors. Similarly, the neighbors of a new user apply the same principle and construct the informative prior distributions for the new user. The prior distribution, or prior, attributes uncertainty to the probability distribution. As the actual recommendation ratings accumulate over the time, the prior distribution is rectified by the actual ratings to reflect the true similarity between two users. Below we describe a maximum a posteriori (MAP) based probability estimation.

Let $\mathbf{p} = [p_1, p_2, \ldots, p_n]$ be a n-state discrete probability mass function (p.m.f.), and $\mathbf{x} = \{x_i\}, i = 1, 2, \ldots, n$, be the number of observed samples for each state. If $g(\mathbf{p})$ is the prior distribution of $\mathbf{p}$, the MAP estimate, $\mathbf{p^{MAP}}$, is defined as the distribution that maximizes the posterior p.d.f. of $\mathbf{p}$.

$$\mathbf{p^{MAP}} = \underset{\mathbf{p}}{\mathbf{argmax}}\ f(\mathbf{x}|\mathbf{p})g(\mathbf{p}), \tag{6}$$

where $f(\mathbf{x}|\mathbf{p})$ is the probability of $\mathbf{x}$ given $\mathbf{p}$.

We select the Dirichlet distribution as the prior distribution. The Dirichlet distribution is conjugate to discrete probability distribution which simplifies the derivation. The Dirichlet distribution is defined as

$$g(p_1, p_2, \ldots, p_{n-1}; \alpha_1, \alpha_2, \ldots, \alpha_{n-1}, \alpha_n) = \frac{1}{Z}\Pi_{i=1}^{n} p_i^{\alpha_i - 1},$$

where $\alpha_i > 0, p_i > 0, \sum_{i=1}^{n-1} p_i < 1, p_n = 1 - \sum_{i=1}^{n-1} p_i$, and Z is a normalization constant, $Z = \frac{\Pi_{i=1}^{n}\Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{n}\alpha_i)}$ ($\Gamma(\cdot)$ is the gamma function). The Dirichlet distribution defines the distribution of $\mathbf{p}$ given parameters $\{\alpha_i\}$. Therefore

$$\begin{aligned} f(\mathbf{x}|\mathbf{p})g(\mathbf{p}) &= \Pi_{i=1}^{n} p_i^{x_i} \cdot \frac{1}{Z}\Pi_{j=1}^{n} p_j^{\alpha_j - 1} \\ &= \frac{1}{Z}\Pi_{i=1}^{n} p_i^{x_i + \alpha_i - 1}. \end{aligned} \tag{7}$$

Solving (6), we get:

$$p_i^{MAP} = \frac{x_i + \alpha_i - 1}{\sum_{i=1}^{n}(x_i + \alpha_i - 1)}. \tag{8}$$

Interestingly, at the beginning with no observations/samples,

$$p_i^{MAP} = \frac{\alpha_i - 1}{\sum_{i=1}^{n}(\alpha_i - 1)}. \quad (9)$$

Parameters $\{\alpha_i\}$ shall be set based on users' inputs. The value of $\sum_{i=1}^{n}(\alpha_i - 1)$ plays an important role in determining how fast the impact of prior distribution diminishes as the number of available samples increases.

• **Setting Parameters of Prior distribution**. The parameters of prior Dirichlet Distribution, $\{\alpha_i\}$, have to be set in such a way that (i) they reflect the confidence level of a user towards his neighbor in terms of the similarity of their opinions about movies; and (ii) the impact of the prior distribution should diminish as more samples are collected. 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. For example, c_0 represents his confidence level that their ratings fully agree with each other. The confidence level is quantified into one to ten: the higher the number, the higher the confidence. A user can input his confidence levels towards his neighbors using a simple GUI interface.

Now we describe how to set the Dirichlet parameters. Suppose user X has a neighbor Y. Denote by R_X and R_Y user X and Y's recommendation ratings. User X sets the initial conditional probability distributions, $P(R_Y = h|R_X = l)$, $l, h = 1, 2, \ldots, N$, as $P(R_Y = h|R_X = l) = c_{|l-h|}/\sum_{i=1}^{N} c_{|l-i|}$. Denote by $\{\alpha_i^l, i = 1, 2, \ldots, N\}$ the parameters of the Prior Dirichlet distribution for $P(R_Y|R_X = l)$. $\alpha_h^l = KP(R_y = h|R_x = l) + 1$. MAP based probability estimation can then be executed according to Equation (8). The value of K is selected to properly control the influence of Prior distribution on the actual samples.

4.1 Setting parameters of Prior distribution

The parameters of Prior Dirichlet Distribution parameters, $\{\alpha_i\}$, have to be set in such a way that (i) they reflect the confidence level of a user toward the neighbor in terms of the similarity of their taste/opinion towards movies; and (ii) the impact of Prior distribution should diminish as more samples are collected.

The confidence level of a user towards the neighbors can be collected using a simple GUI interface as depicted in Figure 3. The users move the sliding bar to select the confidence level. Confidence level ranges from one to ten: the higher the number is, the more similar two users are. There are $N-1$ confidence-level bars, where N is the highest rating. Denote by $\{c_i\}_{i=0}^{N-1}$ the user's confidence levels. The first bar sets the confidence level of c_0 for two users fully agreeing with each other. The i-th bar set the confidence level of c_i for the ratings off by i stars.

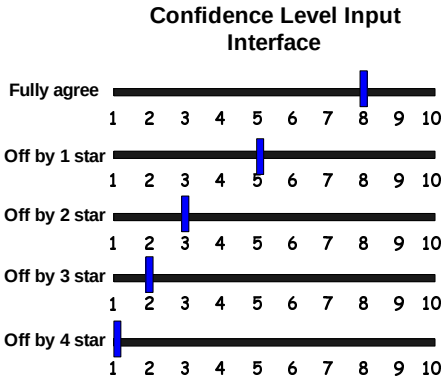

Fig. 3. User Confidence level Input Interface

Now we describe how to set the Dirichlet parameters. Suppose user X has a neighbor user Y. Denote by x and y user X and user Y's recommendation ratings. User X sets up the parameters of N conditional probability distributions, $p(y = h|x = l)$, $l, h = 1, 2, \ldots, N$, as follows:

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

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.

Denote by $\{\alpha_i^l, i = 1, 2, \ldots, N\}$ the parameters of Prior Dirichlet distribution for $p(y|x = l)$. $\alpha_h^l = K \cdot p(y = h|x = l) + 1$, where $K = \sum_{h=1}^{N}(\alpha_h^l - 1)$. The value of K is selected to properly reflect the influence of Prior distribution vs. actual samples.

5 Evaluation

In this section, the performance of the Bayesian inference based recommendation is evaluated using two data sets: Epinions data set [6] and MovieLens data set [26]. 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 dataset, 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 [6] 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 dataset was collected by Paolo Massa in a 5-week crawl of the Epinions Web site 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 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 further assume that the social network topology is the same 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 dataset and then compared with the recommendation rating obtained using the rest of the data. Such procedure is iterated through the entire dataset. Leave-one-out method is commonly used in comparing two recommendation systems.

We compare the performance of Bayesian inference based recommendation with TidalTrust [16] and MoleTrust [24], 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 (MP) 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 traditional Collaborative Filtering (CF), Bayesian inference based recommendation, MoleTrust, and TidalTrust 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 Collaborative Filtering (CF) are cited from [24]. 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 CF column, there are two sub-rows, 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 sub-row is their common coverage, the first sub-row (in Italic font) is the MAE for Bayesian inference, the second and third sub-rows are the MAEs for MoleTrust and TidalTrust respectively.

CF 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 vs. items. Bayesian inference based recommendation and two trust-based recommendations out-perform the CF in terms of MAE for most of the views. CF 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 CF.

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%. 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% smaller than that of MoleTrust, and 12% 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.

5.2 Evaluation using MovieLens Data Set

MovieLens data set [26] 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

Mean Absolute Error				
views	CF	1-hop	2-hop	3-hop
All	0.843	*0.747*	*0.786*	*0.791*
		0.832	0.846	0.829
		0.842	0.826	0.833
	51.28%	28.33%	60.47%	74.37%
Cold users	1.094	*0.761*	*0.893*	*0.878*
		0.674	0.833	0.854
		0.761	0.875	0.878
	3.22%	11.05%	25.02%	41.74%
Heavy raters	0.850	*0.745*	*0.780*	*0.785*
		0.873	0.869	0.846
		0.843	0.823	0.828
	57.45%	30.85%	64.82%	77.81%
Contr. items	1.515	*1.339*	*1.484*	*1.518*
		1.425	1.618	1.687
		1.420	1.610	1.701
	45.42%	25.09%	60.64%	81.80%
Niche items	0.822	*0.460*	*0.633*	*0.699*
		0.734	0.806	0.828
		0.527	0.710	0.771
	12.18%	8.32%	24.32%	20.43%
Opin. users	1.200	*1.114*	*1.205*	*1.202*
		1.020	1.102	1.096
		1.044	1.091	1.092
	50%	23.32%	57.31%	74.24%
Black sheep	1.235	*1.137*	*1.224*	*1.237*
		1.152	1.238	1.242
		1.185	1.228	1.237
	55.74%	23.66%	59.21%	76.32%

TABLE 1
MAE and recommendation coverage comparison of traditional Collaborative Filtering (CF), Bayesian Inference based recommendation, Moletrust and TidalTrust.

yet rich enough to evaluate the algorithm. The ratings are on the numerical five-star scale. Ideally, we would like to have the topology of a social network together with the ratings. Unfortunately, such data set with both social network information and movie rating information are not available to us. Below we construct a synthesized social network topology which observes the *homophily* principle [4].

The data set is divided into two parts: the first 70% 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% ratings are used for testing/evaluation. In Section 5.2.3, we further evaluate the Bayesian inference based recommendation for the entire data set to study the impact of the MAP Prior distribution.

We use the Pearson correlation coefficient to measure two users' similarity. Pearson correlation coefficient between user u and user w is defined as

$$sim(u,w) = \frac{\sum_{i\in I_C}(r_{ui}-\bar{r}_u)(r_{wi}-\bar{r}_w)}{\sqrt{\sum_{i\in I_C}(r_{ui}-\bar{r}_u)^2\sum_{i\in I_C}(r_{wi}-\bar{r}_w)^2}}, \tag{10}$$

where I_C is set of common movies rated by u and w, r_{ui} and r_{wi} are their ratings for movie $i \in I_C$, and $\bar{r}_u$ and $\bar{r}_w$ are their average ratings. Each user randomly picks 500 users out of the entire user population and calculates the Pearson coefficients with each of them. The ten 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 twenty. 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 twenty.

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

Each recommendation calculated by the recommendation engine as in (1) 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.

Figure 4 and 5 depict the recommendation coverage and MAE vs. 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 (MP) 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, however, is significantly 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. We suspect that closer friends offer higher quality recommendations. To verify this, we propose a *hybrid recommendation scheme*. Hybrid recommendation scheme

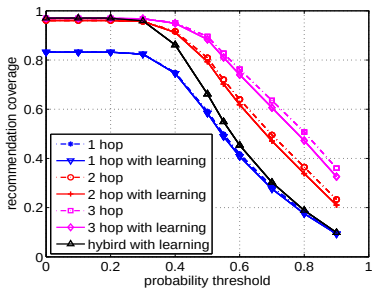

Fig. 4. Recommendation coverage vs. probability threshold.

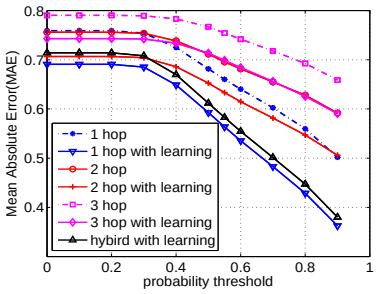

Fig. 5. Mean Absolute Error(MAE) vs. probability threshold.

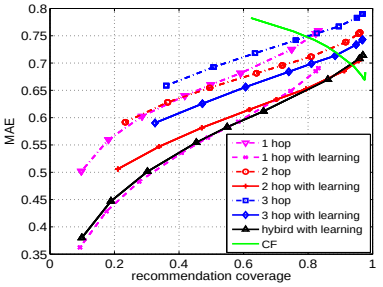

Fig. 6. Comparison of Bayesian inference based recommender vs. CF recommender.

looks for the closest recommender possible, and increases the querying scope only if no closer recommender can be found. A simple way to implement this scheme is to start with TTL=1, and increase TTL by one if no recommendation is found in the previous round. Figure 4 and 5 depict the recommendation coverage and MAE vs. different thresholds for the hybrid recommendation scheme with dynamic learning. Hybrid scheme generates recommendations with MAE similar to 1-hop query, yet the recommendation coverage is greater than that in 1-hop query, especially in the low/mid range of probability threshold.

5.2.2 Comparison with Collaborative Filtering

Now we compare the performance of Bayesian inference based recommendation with the traditional CF technique. We use the user-to-user nearest neighbor prediction algorithm based on Pearson Correlation for the comparison. The personalized recommendation is the weighted average of K nearest neighbors, in terms of Pearson Correlation, from the entire user population. We vary the value of K and plot the curve of recommendation coverage vs. MAE for CF. According to Figure 6, the hybrid scheme performs the best. The hybrid scheme can reach the best MAE of CF with recommendation coverage about 86.1% of CF recommendation coverage. If the probability threshold is set higher, the system can provide even more accurate recommendations. In addition, Bayesian inference based recommendation is a distributed solution and users only share information with direct friends, thus it is more scalable and provides better privacy protection than the traditional centralized CF solutions.

5.2.3 Handling Cold Start

Now we evaluate whether allowing users to set Prior is helpful in mitigating the cold start effect. We use the same topology as in the previous experiments, but simulate the entire data set with three different Prior distributions: good Prior, uniform+good Prior, and uniform+bad Prior. Good Prior uses the confidence levels of $\{3, 1.5, 1, 1, 1\}$. Since direct friends are similar to each other, the good Prior captures such similarity. In uniform+good Prior, half of users (randomly selected) employ good Prior, the other half use uniform Prior with all confidence levels equal to one. Uniform Prior indicates that users have no knowledge about their friends' rating habits. Finally, in uniform+bad Prior, half of users use uniform Prior, half of users use bad Prior, constructed using confidence levels of $\{1, 1, 1, 1.5, 3\}$. Bad Prior intentionally sets high values to the probabilities of recommendations are far off.

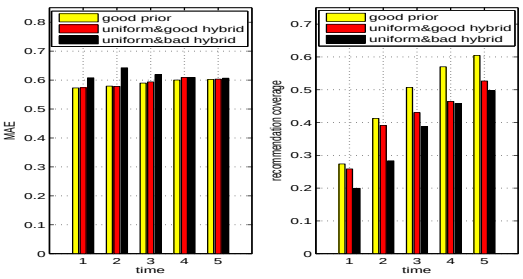

Fig. 7. Handling cold start. MAE and recommendation coverage with different Priors (probability threshold=0.5)

We divide the recommendations along the time line into five equal groups. The result is shown in Fig 7. We set the probability threshold to be 0.5. MAE of good prior and uniform+good Prior roughly remains constant, with marginal increase due to the increase of the number of recommendations. With uniform+bad Prior, the initial recommendation quality is bad. Bad Prior generates some recommendations that is not consistent with users' interests, hence high MAE. The value of MAE improves over time as more samples correct the bad Prior.

Good Prior is able to provide much more recommendations that other two Priors. Users with uniform Prior has no knowledge of their friend's habits, leading to recommendations with low associated probability and thus being filtered out by the probability threshold. Over time, the conditional probability distributions between friends become more accurate

with more samples. The recommendation coverage increases accordingly. Uniform+bad Prior generates the least recommendations. Probability distribution refinement, see Equation (8), transforms the initial conditional probability (bad Prior) to uniform distribution, and then towards more accurate distribution with more and more samples. During the process, the probability density is first spread out, which leads to a smaller number of recommendations compared to other two Priors. In summary, the experiment shows that setting accurate Prior helps new users to obtain ample accurate recommendations from the very beginning, and the uniform Prior is better off than an incorrect Prior.

6 Recommend Friends

A key for the success of CF-based recommendation is to cluster similar users together. Thus, if we can locate more similar users during the recommendation, we can hopefully improve both the recommendation accuracy and coverage. In this section, we propose a new feature for our RS, namely **FriendRec**. FriendRec allows users to query direct friends for recommendation of similar users, and set up connections with them. In other words, users can directly connect with other similar users that are currently 2-hop away in the social network.

6.1 The Design of FriendRec

FriendRec is a module running at individual users. It works as follows. A user, say X, queries its direct friend, say user Y, for like-minded users. When user Y receives the friend recommendation query from X, it compares the history rating of X with that of other friends. A list of candidate users, whose rating similarity with X is over a preset threshold, is returned to X. User X maintains a fixed number of direct friends. Upon receiving the candidate list from Y, X selects a subset of candidates whose similarity is greater than his existing friends. X then contacts these users, attempting to set up direct connections with them and replace the less-similar existing friends.

Algorithm 1 illustrates the algorithm for user Y to generate the candidate list. Y goes through its friend list, and chooses those users who have more than α commonly rated items with X, and whose Pearson Correlations with X are greater than β. Algorithm 2 describes how user X handles a candidate user. X first checks whether its friend list is full or not. If not, user X tries to add the candidate user as its direct friend. If the current number of friends has already reached the threshold, X picks up the least similar existing friend, and compares its similarity with that of the candidate user. If the candidate user has higher similarity, user X attempts to replace the least similar friend with the candidate user. Finally, Algorithm 3 describes how the candidate user, user Z, decides whether to accept the connection request from user X.

Algorithm 1 <Y returns friend candidate List to X>

```
FL: Friend List exclude X
CL: Candidate List to recommend to X
Sim(X,Z): Pearson Correlation between X and Z
CommonItem(X,Z): Number of Commonly rated Items between X and Z
α: Minimum commonly rated item number, system parameter
β: Minimum similarity, system parameter
CL ← ∅
for i = 1 to FL.lenght do
  if (CommonItem(X,FL(i)) ≥ α) then
    if Sim(X,FL(i))≥ β then
      CL← CL ⋃ FL(i)
    end if
  end if
end for
```

Algorithm 2 <Friend Selection at X>

```
FL: Friend List
SL: Similarity List
N: the maximum number of friend, system parameter
isFriend(X,Z):whether Z is a friend of X
isQualify: 1 means yes, 0 means no
isAccept(X,Z): X sends a request to Z for acceptance check
addFriend(X,Z): add Z to X's friend list
Sim(X,Z): Pearson Correlation between X and Z
removeFriend(Y): remove Y from friend list
if (!isFriend(X, Z)) then
  isQualify ← 0
  if (FL.length < N) then
    isQualify ← 1;
  else
    min_sim ← ∞;
    for i=1 to SL.length do
      if (SL(i) < min_sim) then
        min_sim ← SL(i);
        r ← FL[i];
      end if
    end for
    if (Sim(X,Z) > min_sim) then
      isQualify ← 1;
    end if
  end if
  if (isQualify == 1) then
    if (isAccept(X,Z)) then
      removeFriend(r);
      addFriend(X,Z);
    end if
  end if
end if
```

Algorithm 3 <Acceptance check at Z>

N: the maximum number of friend, system parameter
FL: Friend List
Sim(X,Z): Pearson Correlation between X and Z
addFriend(X,Z): add Z to X's friend list
removeFriend(Y): remove Y from friend list
if (FL.length $<$ N) **then**
  addFriend(Z,X);
  return $true$;
else
  min_sim $\leftarrow \infty$;
  for i=1 to SL.length **do**
    if (SL(i) $<$ min_sim) **then**
      min_sim $\leftarrow$ SL(i);
      r $\leftarrow$ FL[i];
    end if
  end for
  if (Sim(X,Z) $>$ min_sim) **then**
    removeFriend(r);
    addFriend(Z,X);
    return $true$;
  end if
  return $flase$;
end if

6.2 Evaluation

In this section, the performance of FriendRec module is evaluated using the MovieLens data set [26]. The initial social network topology is the same as that in section 5. We set the maximum friend number to be 300 in our experiments. For users with the friend number less than 20, they query all direct friends for friend recommendation. For users with more than 20 friends, they randomly choose five friends to query. We set commonly rated item number α to be 20 and the similarity threshold to be 0.4 in our simulation. We sort the data set according to timestamps and simulate the recommendation process over time.

The social network topology evolves every time the users utilize FriendRec feature. In the experiment, we vary the number of queries, or topology evolution times, from zero to five for each user to investigate the effectiveness of FriendRec (zero indicates no topology change). Users send out queries in random order. The result is shown in Figure 8. MAE decreases significantly as the number of topology evolution increases, and flattens out at the end. In contrast, coverage improves even with a single topology evolution.

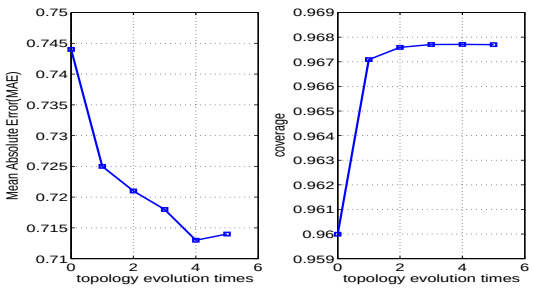

Fig. 8. Social topology evolution. MAE and recommendation coverage with different social topology evolution times .

7 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 the traditional centralized CF based approaches and trust-based approaches, and can flexibly trade off 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 traditional 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 recommendation information. Secondly, 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] Gediminas Adomavicius and Alexander Tuzhilin, "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions", IEEE Transactions on Knowledge and Data Engineering, vol. 17, no. 6, June 2005, doi:10.1109/TKDE.2005.99
[2] Kai Yu, Anton Schwaighofer, Volker Tresp, Xiaowei Xu, Hans-Peter Kriegel, "Probabilistic Memory-based Collaborative Filtering", IEEE Transactions on Knowledge and Data Engineering, vol. 16, no. 1, pp.56-69, Jan. 2004.
[3] Jonathan Herlocker and Joseph Konstan and Loren Terveen and John Ridel, "Evaluating collaborative filtering recommender systems", ACM Transactions on Information Systems (TOIS), vol. 22, no. 1, Jan. 2004.
[4] M. McPherson and L. Smith-Lovin and J.M. Cook, "Birds of a Feather: Homophily in Social Networks", Annual Review of Sociology, vol. 27, no. 1, pp. 415-444, 2001, doi:10.1146/annurev.soc.27.1.415.
[5] YouTube, available at http://www.youtube.com.
[6] Epinions, available at http://www.epinions.com/
[7] NetflixPrize, available at http://www.netflixprize.com/

[8] Twitter, available at http://twitter.com
[9] Facebook, http://www.facebook.com
[10] Flixster, http://www.flixster.com
[11] Flixster, http://www.facebook.com/apps/application.php?id=111849295510739
[12] John S. Breese and David Heckerman and Carl Kadie, "Empirical Analysis of Predictive Algorithms for Collaborative Filtering", Technical Report, Morgan Kaufmann, pp. 43-52, 1998.
[13] Y.-H. Chien and E.I. George, "A Bayesian Model for Collaborative Filtering", Proceedings of Seventh International Workshop Artificial Intelligence and Statistics, 1999.
[14] A. Ansari and S. Essegaier and R. Kohli, "Internet Recommendations Systems", Journal of Marketing Research, vol. 37, no. 3 pp. 363-375, Aug. 2000, doi: 10.1509/jmkr.37.3.363.18779.
[15] Yi Zhang, Jonathan Koren "Efficient bayesian hierarchical user modeling for recommendation system", Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, 2007, pp. 47-54. doi:10.1145/1277741.1277752
[16] Jennifer Golbeck and James Hendler, "FilmTrust: Movie recommendations using trust in web-based social networks", Proceedings of the IEEE Consumer Communications and Networking Conference, pp. 282-286, Jan. 2006.
[17] Cai-Nicolas Ziegler and Jennifer Golbeck, "Investigating interactions of trust and interest similarity", Decis. Support Syst., vol. 43, no. 2, pp. 460–475, 2007, doi:http://dx.doi.org/10.1016/j.dss.2006.11.003.
[18] Cai-Nicolas Ziegler and Georg Lausen, "Analyzing Correlation Between Trust and User Similarity In Online Communities", Proceedings of Second International Conference on Trust Management, pp. 251-265, 2004.
[19] Paolo Massa and Bobby Bhattacharjee, "Using Trust in Recommender Systems: An Experimental Analysis", In Proceedings of iTrust2004 International Conference, pp. 221-235, 2004.
[20] Thomas DuBois and Jennifer Golbeck and John Kleint and Aravind Srinivasan, "Improving Recommendation Accuracy by Clustering Social Neworks with Trust", Proceedings of the ACM RecSys 2009 Workshop on Recommender Systems and the Social Web, Oct. 2009.
[21] John O'Donovan and Barry Smyth, "Trust in recommender systems", Proceedings of the 10th international conference on Intelligent user interfaces, 2005.
[22] Frank Walter and Stefano Battiston and Frank 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.
[23] Karan Sarda and Priya Gupta and Debdoot Mukherjee and Smruti Padhy and Huzur Saran, "A Distributed Trust-based Recommendation System on Social Network", Proceedings of the 10th Second IEEE Workshop on Hot Topics in Web Systems and Technologies, 2008.
[24] Paolo Massa and Paolo Avesani, "Trust-aware Recommender Systems", Proceedings of the 2007 ACM conference on Recommender systems, 2007.
[25] Paolo Massa and Paolo Avesani, "Controversial users demand local trust metrics: an experimental study on epinions.com community", Proceedsings of the 25th American Association for Artificial Intelligence Conference, 2005.
[26] GroupLens, "MovieLens Data Sets", available at http://www.grouplens.org/node/73#attachments, 2010.
[27] Erman Ayday and Faramarz Fekri, "A Belief Propagation Based Recommender System for Online Services" Proceedings of the 2010 ACM conference on Recommender systems, 2010.
[28] Mohsen Jamali and Martin Ester, "A Matrix Factorization Technique with Trust Propagation for Recommendation in Social Networks" Proceedings of the 2010 ACM conference on Recommender systems, 2010.