ROYAL INSTITUTE OF TECHNOLOGY

Exploiting Dynamic Privacy in Socially Regularized Recommenders

RAMONA BUNEA

Master's Thesis at ICT
Supervisor: Shahab Mokarizadeh, Nima Dokoohaki
Examiner: Mihhail Matskin

TRITA xxx yyyy-nn

Abstract

Users of social networks have shown an increasing concern for exposing their personal data to untrusted entities in order to receive recommendations. In this work, we describe the components of a privacy-aware collaborative filtering based recommender framework which targets two important issues in recommender systems operating in a social network: privacy concern of profile owners and sparsity of social trust among users in a social network. Assuming an initial global privacy in the social network, the framework employs a probabilistic matrix factorization technique to estimate the quality of the missing trust relation between each pair of users. Because of the latent features inferred by matrix factorization, the resulting trust is an augmentation of both social relation and user similarity driven trust. We introduce a privacy inference model which exploits the underlying inter-entity trust information to obtain a personalized privacy view for each individual in the social network. Using this personalized privacy view, we employ an off-the-shelf collaborative filtering recommender system to make predictions. Experimental results show that the proposed approach obtains better accuracy than similar non-privacy aware recommender systems, while at the same time meeting profile privacy concerns.

Contents

Chapter 1

Introduction

The continuous growth of social networks in the past decade has led to a bombardment of information, hence the challenging task of filtering out the irrelevant parts became an active subject of research [22], [12], [27], [1]. To this end, varieties of *recommender systems* have been designed to present users with only those items that are of interest to them. Among these, the most popular recommenders are those based on *collaborative filtering* [4], which is a technique relying on the presumption that users with similar profiles will probably enjoy each other's items of interest. Thus, the more a user reveals her profile, the more meaningful the recommendations are. However, revealing profile content goes against the user's requirement of privacy. In practice, users have shown an increasing concern for sharing their private data, especially in the case of untrusted parties [25]. Therefore, the main challenge is to design an effective privacy mechanism that shields against unauthorized access to user's profile, while at the same time exposing sufficient amount of profile information to trusted parties in order to receive helpful recommendations.

Among many existent approaches to recommender systems that pride themselves in providing accurate recommendations, very few tackle the privacy issues and aim to deal with the privacy risk of recommender systems as addressed by [26]. To this end, different techniques such as data disguising [5], maintaining a separate profile for recommendation purpose [33] and differential privacy [20] have been suggested and evaluated. At the same time, identifying and formalizing the relation between trust and privacy in social context has recently gained significant attention [8, 35]. In this light Li et. al. [14] devised trust-aware privacy control over social profiles and He et al. [10] proposed a method to infer profile privacy based on social relations. However, exploiting the correlation between privacy and trust in the context of collaborative filtering recommender systems with the aim of maximizing both privacy control and recommendation quality has not received adequate attention.

In most social networks users explicitly express their trust relationship towards other parties and tend to have more relaxed privacy concerns with respect to these trusted members (friends, neighbors). However, these neighbors are often not sufficient to provide recommendations. To counter this, users need to broaden their trusted network by also considering similarity as a measure of trust. In order to better fit the user's need of privacy, the portion of revealed profile content should be computed individually towards each friend. Inferring privacy from trust is not a trivial matter. However, the general intuition is that privacy is inversely proportional with trust. This thesis presents a privacy-aware collaborative filtering based recommender framework, as well as a set of components that can be used in its implementation. Our work targets two key problems in recommender systems: privacy concern of profile owners and sparsity of social trust among users in a social network.

Chapter 2

Background

2.1 Definition and Classification

Recommender systems are software tools and techniques that handle information filtering and provide suggestions for items to be of use to a user [29]. The purpose of this type of tool is to help individuals who, for various reasons, are unable to decide for themselves what items are useful. Therefore, users who either lack personal experience (known as *cold-start* users) or are overwhelmed by the immense amount of information existent in the social network have to rely on recommender systems to provide them with personalized information. Arguably the most famous example of such a system employed in e-commerce is Amazon.com, a large retailer website that uses a recommender system to personalize the online store for each customer [15].

Commonly, the recommendation problem is described as providing estimated ratings for the items that have not been seen by a user based on the user's ratings for other items and some other types of information. Adomavicius and Tuzhilin [1] formally state the recommendation problem. They define the set of all users as C and the set of all possible items to recommend as S, both of which can be extremely large (up to millions in some cases). Also, they consider an utility function u that measures the usefulness of an item s for a user c (this usually is represented by a rating provided by the user and that indicates how much she liked that particular item). The recommendation problem can then be formally stated:

$$\forall c \in C, s'_c = argmax_{s \in S} u(c, s) \tag{2.1}$$

where $s' \in S$ is the item we want to choose such that for each user $c \in C$ we have a maximized utility. However, the main issue with this approach is that u is usually not defined on the whole $C \times S$ space (users do not provide ratings for all items existent in the network). This gives rise to the need to extrapolate the missing utilities/ratings, which is usually done by either employing heuristics for the utility function and empirically validating the performance or estimating an utility function that maximizes the performance.

Depending on the method used to provide recommendations, recommender systems are classified as follows:

1. *Content-based recommender systems*

   These methods try to recommend items that are similar to the ones the user preferred in the past. In order to do so, they rely on gathering information about and characteristics of the items that are going to be recommended, compare them with items previously rated by the user and recommend the best-matching ones.

   A key problem with this type of recommenders is whether they have the ability to deal with different content types. More precisely, a recommended content of the same type as the user is already using is less valuable than a new content type. For example, a user viewing news articles is more likely to be interested in related videos, interviews, pictures than in similar news articles.

2. *Collaborative recommender systems*

   In the case of these algorithms, the user will be recommended items that users with similar profiles like. This involves the analyze and collection of large amounts of data on user's behavior and attempts to mimic the social process of asking a friend for a recommendation. Among the collaborative filtering recommenders, a particular type relies on *matrix factorization.*

3. *Hybrid recommender systems*

   This approach combines the two methods presented above: content-based and collaborative filtering.

   Hybrid methods can be implemented in several ways:

   a) Make content-based and collaborative-based predictions and then combine them.
   b) Add content-based capabilities to a collaborative-based approach (or vice versa).
   c) Unify both methods into one model.

2.2 Collaborative Filtering

Breese et al. [4] classify collaborative filtering (CF) algorithms into two types:

1. *memory-based algorithms*

   The entire database (all ratings, items and users) is stored in memory and used to make predictions. The desired behavior for this type of CF algorithms is to make predictions for a target user based on the analysis of a sample population. Considering that a user i rates an item j with $r_{i,j}$ and I_i is the set of items rated by user i, the mean vote of user i is:

$$\overline{r}_i = \frac{1}{|I_i|} \sum_{j \in I_i} r_{i,j} \tag{2.2}$$

The idea behind memory-based approaches is to predict the rating the target user t would give to an item j by taking into account partial information about the target user (her mean vote) and a set of weights:

$$r_{t,j} = \overline{r}_t + k \sum_{i=1}^{n} w(t,i)(r_{i,j} - \overline{r}_i) \tag{2.3}$$

where n is the number of users with nonzero weights and k is a normalizing factor that makes the absolute values of weights to unify.

Memory-based algorithms can be further classified in terms of how the above weights $w(t,i)$ are computed. The two main approaches are:

a) *Correlation*

The weights are computed using the Pearson correlation coefficient:

$$w(t,i) = \frac{\sum_{I_j}(r_{t,j} - \overline{r}_t)(r_{i,j} - \overline{r}_i)}{\sqrt{\sum_{I_j}(r_{t,j} - \overline{r}_t)^2 \sum_{I_j}(r_{i,j} - \overline{r}_i)^2}} \tag{2.4}$$

where I_j is the set of items rated by both users t and i.

b) *Vector Similarity*

This method is inspired by the model used to compute the similarity between two documents in the field of information retrieval — each document is viewed as a vector of word frequencies and the similarity is given by the cosine of the angle between these two vectors:

$$w(t,i) = \sum_{I_j} \frac{r_{t,j}}{\sqrt{\sum_{k \in I_t} r_{t,k}^2}} \frac{r_{i,j}}{\sqrt{\sum_{k \in I_i} r_{i,k}^2}} \tag{2.5}$$

2. *model-based algorithms*

This type of algorithms create a model of the social network based on training data using techniques like data mining and machine learning. The developed model is then used to make predictions for real data. Depending on the method used to create the model, these algorithms can be further classified. The most popular models include Bayesian networks, clustering models and latent semantic models. A special category of latent models include the *matrix factorization* technique discussed in more detail in the next section.

Another classification of algorithms used in recommender systems is provided in [31]:

1. *Probabilistic Algorithms*

   Algorithms are considered to be probabilistic if they rely on a probabilistic model and employ a probabilistic distribution when computing predictions (either ratings or ranked recommendation lists). The aim is to calculate the expected value of a vote for an unobserved item, given what is currently known of the target user.

   More explicitly, most probabilistic CF algorithms compute the probability with which the target user u would rate an item i with a rating $r : p(r|u,i)$. Using this probability, the expected rating for item i is calculated using the following formula:

   $$E(r|u,i) = \sum_r r \times p(r|u,i) \tag{2.6}$$

   The two most popular types of probabilistic CF frameworks are Bayesian-network models and clustering/dimensionality reduction techniques.

2. *Non-probabilistic Algorithms*

   The most widely spread non-probabilistic CF algorithms are nearest neighbor algorithms. Among these, two distinct types stand out:

   a) *user-based nearest neighbor* These methods use ratings from similar users (called neighbors) to make predictions.

   b) *item-based nearest neighbor* These algorithms are the mirrored version of the user-based ones: instead of generating predictions based on the similarity between users, they rely on item-similarity. More precisely, the prediction for an item is based on the target user's ratings for similar items. This is formalized in the following formula:

   $$r(u,i) = \frac{\sum_{j \in ratedItems(u)} itemSim(i,j) \times r_{u,j}}{\sum_{j \in ratedItems(u)} itemSim(i,j)} \tag{2.7}$$

   where $itemSim(i,j)$ is a measure of the similarity between items i and j.
   In the related literature, there are several variations on how to compute this item similarity, but the most popular and accurate one remains the adjusted-cosine similarity formula:

   $$itemSim(i,j) = \frac{\sum_{u \subset RB_{i,j}} (r_{u,i} - \overline{r}_u)(r_{u,j} - \overline{r}_u)}{\sqrt{\sum_{u \subset RB_{i,j}} (r_{u,i} - \overline{r}_u)^2} \sqrt{\sum_{u \subset RB_{i,j}} (r_{u,j} - \overline{r}_u)^2}} \tag{2.8}$$

   where $RB_{i,j}$ is the set of users who rated both item i and j.

Collaborative recommender systems can mainly suffer from three types of problems:

1. *Cold start* — It is difficult to make accurate recommendations to new users (users who have not previously rated any items).
2. *Scalability* — Social networks usually deal with millions of users and items, therefore analyzing such a huge amount of data and generating recommendations requires a lot of computation power.
3. *Sparsity* — Given the large number of items in a social network, users only rate a small portion of the total existing products. Therefore, it is difficult to evaluate the similarity between users who have very few items in common.

2.2.1 Matrix Factorization for Collaborative Filtering

At its core, matrix factorization (MF), also known as matrix decomposition, is a mathematical tool for the decomposition of a matrix into some canonical form. In other words, given a matrix M, we would like to find two matrices A and B that, when multiplied, give a very close approximation of M:

$$M \approx AB \tag{2.9}$$

However, from a practical standpoint, matrix factorization can be utilized for the discovery of *latent features* underlying the interaction between two kinds of entities. When applied to the case of recommender systems, these two types of entities are represented by the social network's users and the items they have rated. With this background in mind, matrix factorization attempts to predict ratings by characterizing both items and users with the help of a relatively small number of latent features. For example, let us assume the items are movies. In this situation, the latent features could be measures of obvious dimensions like genre, actors, production year or less obvious characteristics like character development. However, the latent features could as well be completely uninterpretable. Ultimately, it is irrelevant what exactly these latent features represent as long as we find the level of interest users have in each of them.

When dealing with MF in the context of recommender systems, the matrix M becomes the user-item matrix R in which each cell $R[i, j]$ represents the rating user i gave to item j or 0 if item j is unrated. Given the large dimensions of this matrix, the only way to reduce the complexity of the MF method is to choose a smaller number of latent features. The assumption that the number of latent features is smaller than the number of users and the number of items is rather intuitive: if each user was associated with a unique feature, the recommendation process would become useless (users would not be interested in items rated by others). The same logic can be applied in the case of each item being represented by a different latent feature.

A Basic Model

In order to better illustrate the concept of the MF technique, we consider a very basic case in which predictions are made based on this approach. Let us assume a social network with a set U of users and a set D of items. Also, let R be the user-item matrix of size $|U| \times |D|$ that accommodates all the existing ratings in the system. We consider the number of latent features as K. The purpose of running a MF algorithm is to obtain two matrices P of size $|U| \times K$ and Q of size $|D| \times K$ that when multiplied approximate as closely as possible the ratings matrix R:

$$R \approx P \times Q^T = \hat{R} \tag{2.10}$$

Accordingly, each user i has associated a row of features P_i from P and, similarly, each item j has associated a row Q_j. As we stated previously, Q_j represents how strongly related the item is with each type of latent feature. On the other hand, P_i shows how important each latent feature is for the user in case. When this two types of information are combined, the resulting rating is considered the user's predicted interest in that particular item. In practice, the dot product of the two vectors P_i and Q_j gives the predicted rating:

$$\begin{aligned} \hat{R}_{i,j} &= P_i^T Q_j \\ \hat{R}_{i,j} &= \sum_{k=1}^{K} P_{i,k} Q_{j,k} \end{aligned} \tag{2.11}$$

The problem now lies in finding the two matrices P and Q. The basic approach consists of initializing them both with random values, check how different their product is from R and then try to minimize this error iteratively. In other words, we aim to find the minimum of the difference between R and $\hat{R}$. This proposed method is called *gradient descent.*

For each user-item pair, the error between the real rating and the estimated one can be computed using the formula (the square value is used in order to account for the case in which the estimated rating is higher than the real one):

$$\begin{aligned} \hat{e}_{i,j}^2 &= (R_{i,j} - \hat{R}_{i,j})^2 \\ \hat{e}_{i,j}^2 &= (R_{i,j} - \sum_{k=1}^{K} P_{i,k} Q_{j,k})^2 \end{aligned} \tag{2.12}$$

The next step is computing the gradient at the current values of $P_{i,k}$ and $Q_{j,k}$. This allows us to find the direction in which we have to modify their values.

$$\begin{aligned} \frac{\partial e_{i,j}^2}{\partial P_{i,k}} &= -2(R_{i,j} - \hat{R}_{i,j}) Q_{j,k} = -2 e_{i,j} Q_{j,k} \\ \frac{\partial e_{i,j}^2}{\partial Q_{i,k}} &= -2(R_{i,j} - \hat{R}_{i,j}) P_{i,k} = -2 e_{i,j} P_{i,k} \end{aligned} \tag{2.13}$$

After obtaining the gradient, we can correct the current estimations for both $P_{i,k}$ and $Q_{j,k}$.

$$\begin{aligned} P'_{i,k} &= P_{i,k} + \alpha \frac{\partial e^2_{i,j}}{\partial P_{i,k}} = P_{i,k} + 2\alpha e_{i,j} Q_{j,k} \\ Q'_{j,k} &= Q_{j,k} + \alpha \frac{\partial e^2_{i,j}}{\partial Q_{j,k}} = Q_{j,k} + 2\alpha e_{i,j} P_{i,k} \end{aligned} \tag{2.14}$$

The use of the α constant is closely tied to the gradient descent method — it usually takes small values close to 0 (and less than 1) and is a measure against oscillating around the minimum value.

Ideally, we would like our algorithm to end when in each cell of the matrix $\hat{R}_{i,j}$ we find the real rating user i provided for item j (evidently this applies only for the cells which correspond to a real rating existent in the database). All other cells represent the predicted ratings. However, in practice it is not possible to obtain such an exact approximation. For this reason, we need to compute a global error which we then set a maximum accepted value for in order to end our iterative algorithm.

$$\begin{aligned} E &= \sum_{R_{i,j} \neq 0} e_{i,j} \\ E &= \sum_{R_{i,j} \neq 0} (R_{i,j} - \sum_{k=1}^{K} P_{i,k} Qj, k)^2 \end{aligned} \tag{2.15}$$

2.3 Privacy and Trust in Recommender Systems

While most literature on recommender systems focuses on problems like accuracy, data sparsity and scalability, fewer papers confront the issues of trust and privacy, the relation between them and how they influence the quality of recommendations.

2.3.1 Trust

Among these two key problems, trust has received more attention from the researchers in the field of recommender systems. Traditionally, CF algorithms generate recommendations by relying on user similarity. However, in most real social networks this similarity becomes irrelevant in the face of problems such as sparsity and cold start users. In order to alleviate these shortcomings, a new method of inferring similarity has been devised based on interpersonal trustworthiness. Intuitively, trust is regarded as being user-defined and measures the level of confidence a user has in the opinions of others.

Predicting the Sign of Links

Leskovec et. al. [13] divided the trust relationship in social networks in two categories: positive and negative links and devised a model for predicting the signs of links. Positive links signify positive relations such as friendship, support, approval, while negative links indicate disapproval of or distrust in the other party. This mix of positive and negative links is encountered in the majority of social networks which can be generalized with the use of models.

The authors formulate the *edge sign prediction problem*: given a social network with defined signs on all existent edges except the sign on the edge from node u to node v, labeled as $s(u, v)$, which has been hidden, how reliably can the missing sign $s(u, v)$ can be inferred using the information from the rest of the network? The paper's approach to solving this problem is based on a machine-learning framework. The framework uses a collection of features divided into two classes: the first class relies on the signed degree of nodes, which provides an estimate of the local relations of a node to the rest of the network, and the second class is based on a social psychology principle that states that the relationship between two individuals u and v can be approximated with the help of their joint relationship with a third party u.

Trust and Distrust

DuBois et. al. [7] proposed a similar approach based on random graphs to infer values for trust and distrust between unknown users. The trust inference problem is important in the context of social networks with hundreds of users where it is impossible for one user to make a trust statement with respect to every other user in the network.

They base their approach on the method for computing trust based on path probability in random graphs devised in one of their previous works [6]. This method involves the placement of an edge between each pair of users (u, v) with a probability that is dependent on the direct trust value between them $t_{u,v}$. It follows that the trust between two users is inferred from the probability of them being connected in the resulting graph. The main difficulty in extending this method is the integration of distrust. The distrust between users is a more complex measure in the context of this model, mainly because, unlike trust, it is not transitive.

The algorithm the authors designed has its roots in physics and simulates the use of springs in a two dimensional space. The basic idea is that edges between nodes are regarded as springs — they pull nodes together (trust), but at the same time a reasonable space between nodes need to be maintained by making them repel each other (distrust). This spring-embedding algorithm is used in conjunction with the path probability technique for inferring trust in the following manner: path probabilities are computed using only positive edges (only trust information), while the spring embedding algorithm is used iteratively to resolve competing trust/distrust information.

Trust and Similarity

Another approach of integrating trust in recommender systems is to associate it with user similarity: the more similar two user profiles are, the more likely they will trust each other's ratings [39]. The authors consider that recommendations are meaningful only when they are obtained from like-minded people with similar tastes. Therefore, trust should reflect user similarity, at least to some extent.

Studies like the one made by Sinha and Swearingen [34] have provided evidence that users tend to rely on recommendations received from trusted parties such as friends and family members more than on those from online systems. However, the authors dismiss this study on the base of using only a small number of users (nineteen) and on the fact that it did not investigate the reasons behind their findings (what exactly made people more inclined to consider their friends opinions rather than the results of a CF approach).

The paper states that, given an application domain, trusted peers are considerably more similar to their sources of trust than arbitrary peers. A formal description of this statement is given below:

$$\forall x \in A : \frac{\sum_{y \in trust(x)} sim(x,y)}{|trust(x)|} \gg \frac{\sum_{z \in A \setminus trust(x)} sim(x,z)}{|A \setminus trust(x)|} \tag{2.16}$$

where A denotes the set of all community members and $trust(x)$ is the set of all users trusted by x.

Computational Models of Trust

O'Donovan and Smyth [24] presented two computational models of trust based on past behavior of users: profile-level and profile-item-level. They also explain how these models can be easily incorporated into existing collaborative filtering frameworks in order to improve recommendation accuracy.

They ground their work on the idea that user similarity should not be the only criteria for choosing recommendation partners. In their view, the neighbor selection process should also take into account the users history of making reliable recommendations — in other words, the users trustworthiness. To this end, they propose two computational models of trust built around the past rating behavior of individual profiles. In order to define these models, it becomes relevant to make the distinction between two types of profiles: the user receiving the recommendation is referred to as *the consumer* and the user selected as a recommendation partner is called *the producer*. The recommendation process for an item i to a consumer c involves the services of a number of producer profiles and combines their individual predictions. Moreover, a ratings prediction $p(i)$ of an item i by a producer p for a consumer c is considered *correct* if $p(i)$ is within ϵ of c's actual rating $c(i)$. This can be formalized with the following equation:

$$Correct(i,p,c) \leftrightarrow |p(i) - c(i)| < \epsilon \tag{2.17}$$

In building these models, the authors assign a binary success/fail score to every recommendation $p(i)$ made by each producer p to each consumer c depending on whether the prediction is within a distance of ϵ from the actual rating:

$$T_p(i, c) = Correct(i, p, c) \tag{2.18}$$

Furthermore, the paper defines two basic trust metrics based on the relative number of correct recommendations a given producer has made. The full set of recommendations that a given producer has participated in is

$$RecSet(p) = (c_1, i_1), ..., (c_n, i_n) \tag{2.19}$$

while the subset of these that is correct is given by

$$CorrectSet(p) = (c_k, i_k) \in RecSet(p) : Correct(i_k, p, c_k) \tag{2.20}$$

The first model is the profile-level model and uses *profile-level trust* to select suitable recommendation partners. Profile-level trust is the percentage of correct recommendations that this producer has contributed, as depicted in:

$$Trust^P(p) = \frac{|CorrectSet(p)|}{RecSet(p)|} \tag{2.21}$$

The second proposed model is the item-level which uses a more fine-grained metric called *item-level trust.* This metric measures the percentage of recommendations for an item i that were correct:

$$Trust^I(p, i) = \frac{|(c_k, i_k) \in CorrectSet(p) : i_k = i|}{|(c_k, i_k)\ inRecSet(p) : i_k = i|} \tag{2.22}$$

A T-Index Approach

A different view on integrating trust into recommender systems is presented in [37]. The authors depicted users as agents that are connected to each other in a "web of trust" so that the social network takes the form of an agent-based overlay. Based on this model, they propose an ontological model of trust where user's trustworthiness is estimated by a value called T-index. Using this T-index measure, a list of top trustee users per item is constructed, which serves as a base for making recommendations.

The T-index value is similar to the notion of Indegree — the number of edges pointing to the current node. However, in addition to the number of incoming edges, the T-index measure also takes into account the weight of the incoming trust relationships. Therefore, the higher the T-index, the more trusted a user is. The computation of T-index for a given user is rather intuitive. First, all the trust values towards our target user are multiplied by the maximum preset value of T-index and then this set is sorted in a descending order. Then, the members of this set are counted from the beginning of the list until the counter becomes higher than the trust value of a member. This final counter represents the T-index value.

Once the T-index values are known, a list of an item's raters called *TopTrustee* is formed for storing the most reliable users that rated the respective item. This allows users that rate an item to gain access, through the TopTrustee list, to trustworthy users who are otherwise not accessible withing the upper bound of the traversal path length.

The trust notion in the above mentioned works either considers social trust or user profile similarity driven trust, but not both together. Our work aims to combine both users' social trust and their profile similarity aspects into a form of mixed trust which could better represent a user's interest in others.

2.3.2 Privacy

Based on the fact that trust can be regarded as a means of achieving a better privacy scheme [16], the main challenge lies in devising a recommender system that considers the privacy concern of profile owners.

Distributed Aggregation of Profiles

Recommender systems usually employ a central authority (server) that is allowed unconditional access to all user profiles in order to generate accurate recommendations. However, this type of system goes against users' requirement for privacy in two ways. Firstly, if the central authority is aware of the user's real identity (this is usually the case for e-commerce social networks), her private information can be associated with her real identity. Secondly, even if the server does not know the real identity of a user, it can still try to correlate the information from the user's profile to some other data obtained from other sources in order to de-anonymize the user. These threats can be mitigated by hiding crucial information from profiles, but this comes at the cost of accuracy.

In order to mitigate this tradeoff between privacy and accuracy, Shokri et al [33] proposed a method of preserving privacy in a collaborative filtering approach. The method relies on the assumption that each user maintains two separate profiles (which are often synchronized): an off-line profile (which contains the actual user profile) and an on-line profile (which is the altered version of the off-line profile). The recommendations are generated by a central server based on the on-line profile and using a collaborative filtering approach. However, the on-line profile contains added noise in the form of ratings taken from similar users in order to maintain privacy. This allows users to mask some of their personal information in a way that affects the quality of recommendations the least: by mashing data from trusted users into their on-line profile.

Considering a user u whose profile is to be aggregated with items from user v's profile, the authors propose two types of aggregation approaches:

1. *similarity-based aggregation* — user v gives for aggregation a proportion of his items equal with the similarity value between users u and v. Furthermore, these items can be chosen either as the subset of user v's items that have been least rated by users in the system or uniformly at random.

2. *fixed random-selection aggregation* — user v gives for aggregation a *fixed* proportion of his items and these items are chosen uniformly at random

Semi-Trusted Third Party

Aïmeur et al. [2] introduced a hybrid recommender system called *ALAMBIC* relying on content-based, demographic and collaborative filtering approaches which involves using a semi-trusted third party as a means of splitting data. This privacy-preserving recommender system in presented in the context of electronic commerce.

The principle behind their approach is called *division of trust principle* and can be informally stated as "Trust no one, but you may trust two...". In more detail, this means that customers can trust merchants with information about what they want, but not trust them with personal information related to their identity or their past behavior. However, they may trust someone else with identity related data, but not with information about what they want to purchase. In the case of an electronic commerce recommender system, this translates to trusting merchants with the query to the database of available products and trusting another party (known as the *semi-trusted third party*) with demographic information.

The components involved in the ALAMBIC architecture are:

1. *The merchant platform* — The merchant's computing platform is in charge of the catalogue of available products and maintains a database for storing encrypted customer profiles and rating tables. Even though the merchant platform holds this encrypted customer data, it can not interpret it without the help of the *Alambic agent.*

2. *The Still Maker* — This is in fact the semi-trusted third party. Its purpose is to generate a separate *Alambic agent* for each supported merchant platform.

3. *The customer* — wishes to receive product recommendations from the merchant.

4. *The Alambic agent* — serves as an intermediary between the merchant and the customer.

The Still Maker generates the Alambic agent which is then deployed as his surrogate within the merchant platform where he performs the necessary tasks for the recommendation process.

Privacy Inference Model

A model for inferring privacy from trust is suggested by [23] with the aim of protecting profile content against exposure to low trusted users. This model considers privacy as an inverse function of trust, based on the intuition that people are more relaxed to exposing data to friends rather than strangers.

Although this paper presents a framework for automatic selection and composition of web services, this process revolves around the idea of trustworthiness as a measure of the quality of service composition. This trustworthiness is defined as service reputation extracted from user profiles. The relevant part, from a recommender system perspective, revolves around the presentation of a privacy inference model.

While the proposed approach uses one of the already existing trust inference models, the authors define a new privacy inference model. The inferred trust value of user s towards user u is referred to as $trust(s,u)$. Privacy is considered as an inverse function of trust, meaning that a decrease in the confidence in someone results in an increase in the privacy level towards that user:

$$\begin{aligned} &privacy \propto (1 - trust) \\ &trust \rightarrow [0,1], privacy \rightarrow \{0, 0.1, ..., 1\} \end{aligned} \tag{2.23}$$

Although trust can take any value in the interval $[0,1]$, privacy values use more of a coarse grained model because privacy is usually associated with different visibility levels of profile information. The intuition behind formula 2.23 is that people generally have a more relaxed privacy towards their highly trusted friends, but they are not willing to reveal important information to less trusted users. Grounded on this idea, the privacy inference model is explained. Assuming two users s and u and p_s the privacy value of the profile of user s, the inferred privacy rating of user s from user u's perspective is:

$$privacy(s,u) = \begin{cases} \alpha(1 - trust(s,u)) + \beta p_s, \text{if } trust(s,u) \geq Min_{trust} \text{ else} \\ \gamma(1 - trust(s,u)) + p_s, 0 < \gamma < 1,\ 0 \leq \alpha,\ \beta < 1,\ \alpha + \beta = 1 \end{cases} \tag{2.24}$$

where Min_{trust} represents the trust threshold for considering another user trusted. The γ parameter controls how fast the privacy towards untrusted users increases. Similarly, if we want to reward highly trusted users with a fast decreasing privacy, we can enforce the constraint $\alpha \gg \beta$.

Our approach extends the privacy inferences model of [23] by considering a exponential correlation between trust and privacy measures and then exploits the model to make recommendations.

2.3.3 Trust and Privacy Correlations

The Trust-Privacy Relation

Olson et. al. [25] showed in their study that users' desire to share personal information varies with respect to the other parties involved. As a result of their work, they managed to cluster into fine grained categories both information items that are subject of sharing and people whom these items are meant to be shared with.

These findings can provide guidance to the further development of access control and interfaces as a means to make specifications easier for the end user. They also illustrate the case of how a preference-specification tool would permit its users to specify generic permissions for each category of people — family, co-workers, the public — but, at the same time, allow them to make exceptions for some particular user or even information type. This method would be the most apparent and probably most efficient way of dealing with privacy and trust — letting users decide for themselves who is trusted with what.

However, if we take into account the large amount of information that user profiles contain and the number of connections each user has, this decision process becomes quite laborious for a person to handle themselves. Therefore, it would be ideal for this process to be automated and done transparently of the user's knowledge.

Trust-Privacy Trade-off

Seigneur and Jensen [32] have shown that there is a trade-off between trust and privacy and proposed an approach to disclose a minimal amount of personal information for the required trust. They consider information to become personal (therefore more important to the user) when it can be used to trace back the individual or link two individuals together.

The idea they promote in their work is based on a technique that controls the dissemination of this type of personal information. In other words, their illustrated approach tries to limit or, in the best case, prevent the linkability of information to individuals. However, because knowledge about the other party lies at the very foundation of trust, perfect privacy protection prevents the exploitation of trust in the online world, but also its formation and evolution. As a means of achieving their goal, they propose a computational trust/risk-based security framework with two sub-components: a trust engine (for a dynamic trustworthiness evaluation based on evidence) and a risk engine (for the evaluation of the risk involved in the interaction). They also define a privacy/trust trade model, which states that there are two main consequences of the increase in knowledge about an entity: a more accurate evaluation of trustworthiness and a privacy decrease (this is a one-way function because privacy recovery is hard to achieve).

Privacy-Accuracy Trade-off

On the other hand, Dokoohaki et al. [5] pointed out the conflict between privacy and accuracy and suggested an optimized trade-off between these two opposing goals in the form of a Pareto set. To this end, they proposed a framework for inferring this optimal set by experimenting with the privacy and accuracy factors.

Their framework is build on the foundations of the generic architecture for a trust-aware recommender system presented in [18]. This architecture takes as input the rating matrix and the trust network and is composed of two main building blocks: a similarity metric and a rating predictor. Their role is rather intuitive: the similarity metric finds similar users (neighbors), while the rating predictor generates predicted ratings based on the weighted sum of ratings given by the similar users to these respective items. The authors extended this architecture by introducing a new way of discovering neighbors (in addition to the existing similarity module): a trust metric module. In more detail, this trust metric module consists of two parts: a private trust metric and a private rating predictor. The overall functionality of the added module can be resumed to the generation of a private trust estimation from a masked rating set taken as input. This obtained trust is used by the private rating predictor. The effect of adding this trust metric module to the main generic architecture is the computation of the predicted rating matrix based on the combination of the private rating predictor and the rating prediction module. The paper also proposes a protocol for implementing the private trust metric that relies on data disguising using z-scores and a private trust calculation algorithm. The authors conclude their approach by identifying the conflicting goals of privacy and accuracy and demonstrate how a Pareto set can be inferred in order to mediate the tradeoff between these two conflicting goals.

2.4 Challenges and Problems

When designing an algorithm or a framework for a recommender system, one needs to keep in mind the fundamental problems that recommender systems suffer from and try to address as many of them as possible. Good approaches should be able to work in any type of social network and produce relevant results. In this section we will present the main challenges a new recommendation algorithm should tackle [36].

1. *Quality of Recommendations* — It is also referred to as *accuracy*. Users of a social network who employ the service of a recommender system expect to receive recommendations they can trust. In order to do this, a good recommender system should attempt to minimize the false positive errors. These errors occur when a user is recommended (positive) a product which she does not like (false).

2. *Sparsity* — Even in the case of active social networks it is still impossible for users to rate a large percentage of items. Because there are hundreds of

thousands of items available and users usually rate a small number of them, the user-item matrix becomes sparse. In turn, this leads to difficulties or even the impossibility of finding users who have items in common. Finally, this waterfalls into the generation of weak recommendations.

3. *Scalability* — Recommendation processes require heavy computing which increases with both the number of users and items available in the social network. The rapid evolution of social networks can make an efficient algorithm become unable to generate a satisfactory number of recommendations in reasonable time. A workaround this problem would be the use of a distributed approach, but generally this comes at the cost of accuracy: without a global knowledge the recommendations drop in quality.

4. *Loss of Neighbor Transitivity* — Given two users u_i and u_k, who share a high correlation, and user u_k is also highly correlated with user u_l, it is a possibility that users u_i and u_l are also highly correlated through their common correlation with u_k. However, collaborative filtering approaches only capture this possibility if users u_i and u_l have rated many common items.

5. *Synonymy* — In some social networks it might happen that two items with different names refer to similar objects. In other words, there is some latent association between products. As a result, better recommendations can be generated if this effect is taken into account. Also, synonymy can lead to the alleviation of the sparsity problem: even though users do not have items in common, they might have rated similar items.

6. *Cold Start Problem* — It is also known as *first rater problem.* New users of a social network cannot receive a recommendation without having any rating history (users have empty profiles). This also applies to newly introduced items: unless a user independently decides to rate a new item, there is no way for this item to be recommended to other users. Although it diminishes, the problem can also occur in the case of unpopular items who receive few ratings or the case of users who only rated such unpopular items.

7. *Unusual User Problem* — It is also known as the *Gray Sheep Problem* and occurs in the case of users with unusual opinions who tend to not agree or disagree consistently with any group of users. In this case it becomes difficult for recommender systems to find likeminded neighbors and therefore the quality of predictions drops.

2.5 Evaluation Metrics

Although a recommender system's evaluation in terms of time and space requirements is also important, in this section we focus on the evaluation of predictions. While the first type of requirements can be improved to a certain level with the use of a more computationally powerful machine, it requires a reevaluation of the entire algorithm to change which items are recommended.

A prediction is a numerical value that the recommender system considers as corresponding to the rating r_{ij} given by the user u_i to the item i_j. These predictions can be evaluated from two perspectives:

1. *accuracy* — There are two general approaches to estimating the accuracy of a recommender system:

   a) *Statistical Accuracy* — measures how close is the predicted rating r_{ij} to the actual numerical rating ar_{ij} in the case of a user u_i and an item i_j. The most commonly used metric is the *Mean Absolute Error* (MAE) which measures the deviation of the predicted ratings from the real values. For a user u_i, the MAE is:

   $$MAE_i = \frac{\sum_{j=1}^{n_i} |ar_{ij} - r_{ij}|}{n_i} \tag{2.25}$$

   The total MAE is computed by averaging the MAEs for all user:

   $$MAE = \frac{\sum_{i=1}^{m} MAE_i}{m} \tag{2.26}$$

   where m is the total number of users in the social network.

   A similar metric for statistical accuracy is the *Root Mean Square Error* (RMSE):

   $$RMSE = \sqrt{\frac{\sum_{(u_i,i_j)\in S}(ar_{i,j} - r_{i,j})^2}{m}} \tag{2.27}$$

   where S is the total user-item space.

   b) *Decision Support Accuracy Metrics* — are used to distinguish between high quality items and the rest of the items. These metrics are used when the rating scheme is binary.

2. *coverage* — is computed as the fraction of items for which a prediction was generated:

$$Coverage = \frac{\sum_{i=1}^{m} np_i}{\sum_{i=1}^{m} n_i} \tag{2.28}$$

Chapter 3

Problem Description

Users of a social network employing a recommender system have two main concerns: receiving accurate recommendations and maintaining the sensitive information from their profile private. Unfortunately, these two issues represent conflicting goals — in order to produce meaningful recommendations, information about users is required. Collaborative filtering techniques output the best results when all profiles are shared and users have many items in common. However, this is an ideal case which is not encountered in real world social networks.

Social networks provide the means for users to relax their privacy settings towards trusted parties in exchange for better recommendations. In other words, a user reveals more of her profile to friends so they can recommend her items of interest. This would seem like a plausible solution for the aforementioned problem: user explicitly express trust relationships which are used to disclose profile content and result in accurate recommendations. However, this approach behaves poorly in a real world social network because of *sparsity*. With thousands of items available for rating, users only manage to review a small portion of them. This leads to a sparse user-item matrix, which is already pruned by the privacy settings, and results in a poor recommendation process. To make matters worse, sparsity also characterizes the social trust matrix: a user has a small number of trusted friends.

In this thesis we propose a method of dealing with sparsity while maintaining a user privacy setting. In order to deal with the sparsity of the trust matrix, we need to infer the missing trust relations between each pair of users. We choose to employ a probabilistic matrix factorization approach to predict all the trust pairs. The effect of using this type of method is that the resulting trust will be a mixture of social trust and similarity. This process has the effect of extending our list of friends to every user in the system. However, it must be noted that not all users are trusted equally. It is rather intuitive that this trust diversity will result in a privacy diversity — a user will reveal a different proportion of profile to each other user.

The main contribution of this work is the proposal of a privacy inference model based on the *direct network effect*. We model the trust-privacy dependency as a logarithmic function. Given that the trust between each pair of users is known, we can also infer the privacy between all users pairwise. Providing recommendations for one user is now a straightforward process: each user will have her own view of the ratings in the social network according to the privacy requirements of the other users towards our target user. This view is personalized with regards to user privacy and allows the direct use of collaborative filtering methods.

Chapter 4

Approach

Considering an initial global privacy, our framework estimates the quality of all the missing trust relations in the social network by employing a probabilistic matrix factorization technique. A consequence of the manner in which matrix factorization infers the latent features of the network is that the resulting trust becomes an augmentation of both social trust and user similarity. We introduce a privacy inference model which infers a personalized privacy view for each individual in the social network based on the underlying inter-entity trust information. Using this personalized privacy view, we employ an off-the-shelf collaborative filtering recommender system to make predictions. We provide empirical evidence to support the claim that our method shows an improvement in performance (in terms of both accuracy and coverage) over similar non-privacy aware recommender systems, while meeting profile privacy concerns.

4.1 The Framework

The proposed framework operates on a social network of users where the friendship among users is regraded as social trust. Moreover, we assume that each user in this network maintains a profile of ratings over items of interest and retains a predefined privacy setting over her profile. This makes a certain proportion (or alternatively certain ratings) of profile visible (accessible) by other social network members. The goal of the framework is to make rating recommendation while respecting the privacy concern of the profile owners.

The architecture of our proposed privacy aware recommendation framework is illustrated in Fig.4.1. The profiles of social network members are presented as a matrix of ratings where rows refer to users and columns refer to items. Similarly, the social trust between users is captured by a trust matrix where each entry $t_{i,j}$ shows that there is a trust relation from user i toward user j with quality $t_{i,j}$. To mitigate the problem of data sparsity, given the rating and trust matrices, we use a matrix factorization technique to estimate the missing entries in both matrices simultaneously. We refer to the matrix accommodating the estimated trust values as

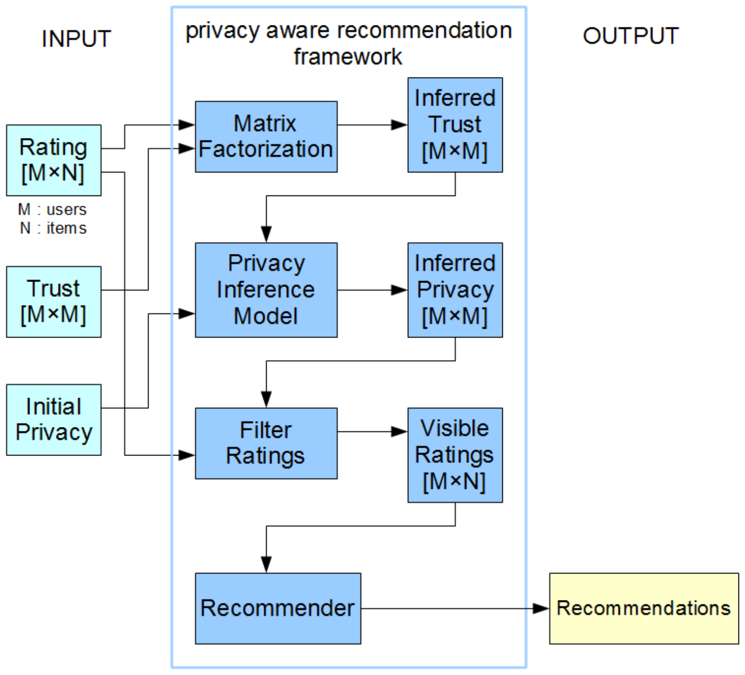


Figure 4.1. Architecture of Privacy-aware Collaborative Filtering Recommender Framework

inferred trust matrix. Next, we use our model to compute a secondary privacy from the inferred trust matrix, resulting in an asymmetrical privacy coefficient between each pair of users. In the next step, we apply the pair-wise privacy matrix over the rating matrix and obtain matrices of visible ratings for each user. Each visible rating matrix denotes a personalized view over the set of visible ratings in the entire network for a specific user. Finally, given the inferred trust matrix and matrices of visible ratings we generate recommendations. It should be noted that although we provide an example of implementation, the components of our framework are highly flexible and can be customized according to the requirements in hand.

In the rest of this section, first we describe how a state-of-the-art matrix factorization technique can be used to deal with the sparsity problem in the trust matrix. Next, a privacy inference model is presented where the objective is to design a dynamic privacy scheme that ensures user profile information (in our case ratings) is disclosed proportionally to the trustworthiness of the audience. Finally, we describe how an off-the-shelf collaborative filtering system can be plugged in into our framework.

4.2 Obtaining trust matrix

As can be seen in Fig. 4.1, the proposed framework relies on the availability of quality of trust relationship between each two individuals in the social network. We project the quality of trust relationship into a real value in range $[0, 1]$, where zero denotes distrust and complete trust is represented by one. Consequently, two major questions arise:

- What exactly is *trust*? Is it *social trust* which is depicted as the quality of friendship between members of a social network [19] or is it the trust which can be derived based on *similarity* between items of interest among individuals [11]? Social trust offers the advantage of taking into account the user's social relations and personal preferences when it comes to entrusting others with her profile or parts of it. However, there is no guarantee that the recommendations received solely using our social relations are accurate and useful enough. On the other hand, despite the fact that similarity driven trust would offer more accurate predictions, a targeted user might not be willing to reveal his ratings to unknown individuals just because they share some items of interest. It follows that both social trust and similarity driven trust have their drawbacks. A usable compromise would be finding a form of *mixed trust* that incorporates social trust and similarity.

- In most real world scenarios the social relationship between each two individuals in a social network either is not available or it is not defined. In fact, data sparsity has been recognized as one of the major challenges confronted by every recommender system [17]. As our proposed approach heavily relies on the knowledge of the quality of the trust relation between each two social network users, the question is how data sparsity in the trust matrix can be handled?

Fortunately, both issues can be resolved simultaneously by recruiting a specific extension of probabilistic matrix factorization [17]. This allows us to combine social trust with similarity into one metric and estimate missing trust values.

4.2.1 Foundations of matrix factorization

In its basic form, matrix factorization refers to the decomposition of a matrix into a product of other matrices. In recommender systems however, this technique is used to reflect the latent features of the user-item space. To further detail this aspect, let us assume a social network consisting of users rating movies: although a user provides only one rating for a movie, it is rather intuitive that this rating is influenced by the user's opinion on the movie's genre, its actors or some other features (which are known as latent features).

Given a social network with m users, n items and its ratings matrix $R^{m \times n}$, we consider there are l different latent features. Furthermore, user i has a certain opinion in ranking items that have a latent feature U_{ki} of type k. U_{ki} can also be regarded as the user's confidence in that particular latent feature. In addition, each feature k has a certain quality value for each item j, represented as V_{kj}. As a result, user i would rate item j as $\sum_{k=1}^{d} U_{ki}V_{kj}$.

With the same reasoning being applied to the trust matrix $C^{m\times m}$, we use matrix factorization in our social system in order to simultaneously factorize the social network trusts and ratings matrix: $C = U^T Z$ and $R = U^T V$, where $U^{l\times m}$, $Z^{l\times m}$, $V^{l\times n}$.

4.2.2 Probabilistic matrix factorization

We use probabilistic matrix factorization as presented in [17] to infer the missing ratings and trusts. For our implementation we need to map, without any loss of generality, the ratings to the interval $[0, 1]$ as detailed below. Finding the matrices U, V and Z is equivalent to minimizing the following function:

$$\begin{aligned} L(R, C, U, V, Z) =& \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}^{R}(r_{ij} - g(U_i^T V_j))^2 + \\ & \frac{\lambda_C}{2}\sum_{i=1}^{m}\sum_{k=1}^{m} I_{ik}^{C}(c_{ik}^{*} - g(U_i^T Z_k))^2 + \\ & + \frac{\lambda_U}{2}\|U\|_F^2 + \frac{\lambda_V}{2}\|V\|_F^2 + \frac{\lambda_Z}{2}\|Z\|_F^2 \end{aligned}$$

where:

- I_{ij}^{R} is 1 if user i rated item j and 0 otherwise
- $g(x) = \frac{1}{1+e^{-x}}$
- I_{ik}^{C} is 1 if user i trusts user k and 0 otherwise
- $c_{ik}^{*} = \sqrt{\frac{d^-(v_k)}{d^+(v_i)+d^-(v_k)}} \times c_{ik}$, where $d^+(v_i)$ represents the outdegree of node v_i, while $d^-(v_k)$ is the indegree of node v_k
- $\|.\|_F^2$ denotes the Frobenius norm
- $\lambda_U = \lambda_V = \lambda_Z$ (in order to reduce the complexity of the model)
- R is the matrix of the original ratings $1 \cdots R_{max}$ mapped to the interval $[0, 1]$ by using the function $f(x) = \frac{x-1}{R_{max}-1}$

In order to find the U, V and Z matrices corresponding to the minimum value of the objective function L, we perform gradient descent in the column vectors U_i, V_j and Z_k:

$$\begin{aligned}\frac{\partial L}{\partial U_i} =& \sum_{j=1}^{n} I_{ij}^{R} g'(U_i^T V_j)(g(U_i^T V_j) - r_{ij})V_j + \\ &+ \lambda_C \sum_{k=1}^{m} I_{ik}^{C} g'(U_i^T Z_k)(g(U_i^T Z_k) - c_{ik}^{*})Z_k + \lambda_U U_i\end{aligned} \tag{4.1}$$

$$\frac{\partial L}{\partial V_j} = \sum_{i=1}^{m} I_{ij}^{R} g'(U_i^T V_j)(g(U_i^T V_j) - r_{ij})U_i + \lambda_V V_j \tag{4.2}$$

$$\frac{\partial L}{\partial Z_k} = \lambda_C \sum_{i=1}^{m} I_{ik}^{C} g'(U_i^T Z_k)(g(U_i^T Z_k) - c_{ik}^{*})U_i + \lambda_Z Z_k \tag{4.3}$$

where $g'(x) = \frac{e^x}{(1+e^x)^2}$ is the derivative of $g(x)$.

4.2.3 Gradient descent

The gradient descent method [21], also known as the steepest descent method, is one of the main algorithms for finding the local minimum of a general nonlinear function of the form $f(x_1, x_2, \cdots, x_n) = 0$.

The idea behind this method is the simple observation that a continuous function should decrease if we take a step along the direction of the negative gradient. In our small example, we would start at a point $P_0(x_{10}, x_{20}, \cdots, x_{n0})$ and repeatedly move from P_i to P_{i+1} by going in the direction of the negative gradient of f at P_i. Therefore, the only required condition for using this method is that one should be able to compute the gradient of the function f.

Algorithm 1 Gradient Descent

```
choose initial $P_0$
for $i = 0$ to $maxiter$ do
    $P_{i+1} = P_i - step * \nabla f(P_i)$
    if converged then
        break
    end if
end for
```

There are several aspects that require further discussion:

I. when does the algorithm converge (the *converged* condition becomes true)?

II. what is the optimal value for *step*?

III. how is P_0 chosen?

The convergence of the gradient descent method can be tested in several ways which can all be reduced to how much progress was made in the current iteration. More precisely, we test if the module of the difference $P_{i-1} - P_i$ is less than a tolerated error and if it is, then the algorithm has converged. Other variations include testing whether the current value of the module of the gradient $\nabla f(P_i)$ is lower than a predetermined threshold.

The choice for the value of *step* is affects the convergence the most. If we have a small step the algorithm will become inefficient and converge slowly. On the other hand, large steps will provide potentially bad results by actually skipping over the minimum value.

Several algorithms that provide a way to initialize P_0 exist in the literature, but the most popular way remains the initialization with random values.

4.2.4 Solving matrix factorization with gradient descent

In Algorithm 2 we perform matrix decomposition using gradient descent.

Algorithm 2 Matrix Factorization

```
initialize $U_0, V_0, Z_0$ with random values
for $p = 0$ to $maxiter$ do
    for $i = 1$ to $m$ do
        $U_{ip+1} = U_{ip} - step * \frac{\partial L}{\partial U_{ip}}$
    end for
    for $j = 1$ to $n$ do
        $V_{jp+1} = V_{jp} - step * \frac{\partial L}{\partial V_{jp}}$
    end for
    for $k = 1$ to $m$ do
        $Z_{kp+1} = Z_{kp} - step * \frac{\partial L}{\partial Z_{kp}}$
    end for
    compute $L(U_{p+1}, V_{p+1}, Z_{p+1})$
    if converged then
        break
    end if
end for
```

We begin our algorithm by initializing all three matrices U, V and Z with random values between -0.001 and 0.001. Based on this algorithm, it can be seen that the matrices U, V and Z mutually influence each other. This shows that the inferred ratings and trusts are closely tied, following the intuition that a user's social connections will affect its behavior. Moreover, the obtained trust matrix C contains values that are a merge of social trust and trust as a form of similarity between the users' ratings.

The value of *step* is of main concern when dealing with the convergence of a gradient descent algorithm. As we get closer to the minimum of L, the values of the partial derivatives become smaller, which in turn leads to a slower descent down the gradient slope. A solution to this is to dynamically adjust the *step*: start with a slow value and increase it as we approach the minimum. Evidently, *step* has to be positive (otherwise we will end up going upwards the gradient and consequently finding the maximum of L) and less than 1 (we might end up going over the minimum value if $step > 1$). We found 0.1 to be a good initial value that we increase with the number of iterations using the following rule:

Algorithm 3 Adjust Step

```
if $p > 100$ then
    $step = (100 + p * 1.5)/1000$
    if $step > 0.5$ then
        $step = 0.5$
    end if
end if
```

We only start applying this rule after the first 100 iterations to make sure that L has decreased to a reasonable value, otherwise we might end up with big values for the partial derivatives. This method is used for big datasets with thousands of users so it can happen that if we start the descent with large steps, the values for the partial derivatives will not fit into a variable of type *double*. For the same reason we need to set an upper bound when computing the next value for *step*, in our case this limit is 0.5. Also, if the step is too big, we might end up oscillating around the minimum value, without being able to reach it.

In the classic gradient descent method, we would test the convergence by checking whether $|L(U_{p+1}, V_{p+1}, Z_{p+1}) - L(U_p, V_p, Z_p) < tolerated\_error$. In our particular case however, it is more suited to test it by measuring how close to the actual ratings matrix R our approximated solution at the current iteration $U_{p+1}^T V_{p+1}$ is. To do so, we compute the mean absolute error (MAE) between the ratings as given in the dataset and our predicted rating:

$$MAE = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} |r_{ij} - \hat{r}_{ij}|}{N}$$

where N denotes the total number of ratings in the initial matrix R. and $\hat{r}_{ij}$ is the rating that user i gave to item j as predicted by the matrix factorization.

$$\hat{r}_{ij}^* = \sum_{l=1}^{d} u_{li} * v_{lj}$$

In order to compute an actual rating between 1 and R_{max}, we need to firstly map the value $\hat{r}_{ij}^*$ to $[0, 1]$ using the logistic function $g(x)$, then to the interval $[1, R_{max}]$ by applying the inverse of the $f(x)$ function.

$$\hat{r}_{ij} = 1 + \frac{4}{1 + e^{-\hat{r}_{ij}^*}}$$

We only consider the MAE for the ratings matrix and not also for the trust matrix because we are interested in having accurate predictions for ratings, this in turn being reflected in the computed trust values and resulting in better recommendations.

After computing the matrix factorization using the gradient descent method, we obtain a matrix $\hat{R}$ which has in each cell that corresponds to a non-zero cell in R a value that comes as close as possible to its mirrored value in R. All the other cells in $\hat{R}$ reflect the predicted ratings for that particular social network. The same reasoning can be applied to the inferred trust matrix $\hat{C}$.

4.3 Privacy Inference Model

The notion of *privacy* is often introduced as an individual's capacity to control the conditions under which his identity information can be shared [3]. The idea of privacy inference is originated from the incentive that individuals in a social network adjust profile privacy with respect to their level of trust in others. In other words, decreasing confidence in someone leads to the strengthening of the privacy level towards that individual. This relation between trust and privacy is widely accepted in the literature [8, 10, 35] and on-line social networking sites. For example, *Facebook* users can group their friends into *FriendLists* which allows them to selectively present different profile content to each of these friend lists.

Despite of exhaustive studies, there is no consensus on the nature of the relationship between privacy and trust [35] and in different networks or applications trust and privacy may have different correlation forms [38]. We envision an exponential correlation between privacy and trust due to observed *direct network effect* [9] in social networks. The network effect (a.k.a network externality) is a theory in economics and business stating that for certain products (e.g. telephone) or services (e.g. social networks) the willingness to pay arises as the quantity of consumers or users rises, provided that: 1) there is zero utility in a network of zero size (i.e. zero utility for the first user) 2) there is immediate benefit in the expansion of a small network for a user [9]. On-line social networks work in the same way, being more useful as more users join. With respect to similarity driven trust, the more people expose their profile content, the more likely they are to become *friends* (and trust each other more) with other similar users [11].

At the same time, inverse demand curves are often used to represent the demand for products with direct network effect characteristics [30]. The curve considers the price as a function (linear or exponential) of quantity of consumers. In our case, the willingness to pay is resembled by the privacy metric (in form of exposed proportion of profile content) and the quantity of users is substituted by the number of trusted connections (which is in turn influenced by the trust quality of connections). Obviously there is no gain in exposing profile when the network is zero (the user is an isolated node in the network) and there is often immediate benefit in expanding the trust network considerably due to the quality and quantity of gained recommendations from friends. Hence, it can be seen that both direct network effect essential constraints are satisfied based on this analogy. Consequently, a similar correlation (either linear or exponential) between privacy and trust metric can be derived. In this work, we opted to exploit the the exponential function of this relationship, as depicted in Fig.4.2.

We exploit this correlation in order to devise a personalized privacy setting mechanism where the profile owner can decide on the quantity[1] of exposed content based on the trustworthiness of the audience. In such setting, the current (primary or default) visibility of profile content for less trusted connections is diminished, while highly trusted connections receive the opportunity to access a larger proportion of profile content. To this end, there is a need to define a trust threshold which acts as an adaptive barrier to control the visibility of the audience by imposing an inferred (*secondary*) privacy setting. We use the following formula to compute the secondary privacy P_{ij} of user u_i towards u_j knowing the quality of trust T_{ij} from user u_i toward u_j:

$$P_{ij} = e^{\beta \times T_{ij}} \tag{4.4}$$

Both P_{ij} and T_{ij} take values between 0 and 1 and β is a constant controlled by both trust threshold T_{th} and initial privacy value P_0. Ideally, each user can define his own default privacy setting and consider different trust threshold. Based on the aforementioned discussions regarding adjusting the privacy with respect to trust quality, the designated trust threshold needs to satisfy the following constraints:

$$\begin{aligned} &if\, T_{ij} > T_{th}\, then\, P_{ij} < P_0 \\ &if\, T_{ij} < T_{th}\, then\, P_{ij} > P_0 \\ &if\, T_{ij} = T_{th}\, then\, P_{ij} = P_0 \end{aligned}$$

From these constraints, we conclude that the point (T_{th}, P_0) must be located on the curve of equation 4.4, as shown in the Fig.4.2. This implies that:

$$\beta = \frac{\log P_0}{T_{th}}$$

After computing the secondary privacy for all social network users, each user i will now have a personalized view of the ratings in the system: every other user j will reveal $(1-P_{ji}) * N_j$ of his ratings to user i, where N_j is the total number of items user j has rated. Finally, it is worth mentioning that P_{ij} is not necessarily equal with P_{ji} due to the fact that trust between two individuals is not necessarily symmetric.

4.4 Making recommendations

Assuming default (and global) privacy P_0 for all social network members, we run matrix factorization on the trust matrix C and the ratings matrix R^* in which every

[1]Since the discussion on the quality of exposed profile is quite subjective and exhaustive, we leave this issue for the future work

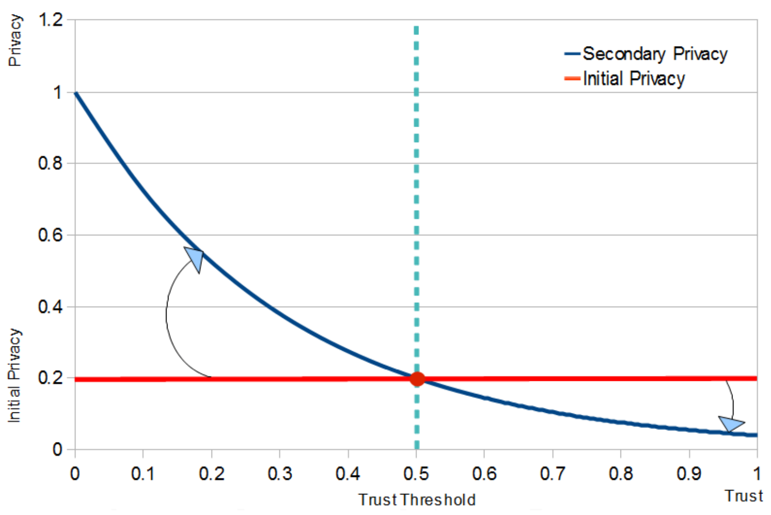


Figure 4.2. Hypothesized correlation between Privacy and Trust

user hides P_0 percent of her total ratings. As a product of matrix factorization we obtain the inferred trust matrix which serves as a basis for computing the secondary privacy. Running matrix factorization technique again for generating recommendation is computationally expensive due to the fact that the factorization needs to be done independently for every user. Instead we choose to use Resnick's prediction method [28] to predict user's i rating on item j:

$$r_{ij} = \overline{r}_i + \frac{\sum_{u \in U(j)} (r_{uj} - \overline{r}_u) * sim(i,u)}{\sum_{u \in U(j)} |sim(i,u)|}$$

where $\overline{r}_i$ is the mean rating of user i, $U(j)$ represents all the users that rated item j and these ratings are also visible to user i and $sim(a,b)$ denotes the similarity between user a and user b calculated using Pearson's correlation coefficient:

$$sim(a,b) = \frac{\sum_{i \in I(a,b)} (r_{ai} - \overline{r}_a) * (r_{bi} - \overline{r}_b)}{\sqrt{\sum_{i \in I(a,b)} (r_{ai} - \overline{r}_a)^2} * \sqrt{\sum_{i \in I(a,b)} (r_{bi} - \overline{r}_b)^2}}$$

where $I(a,b)$ are those items rated by both users a and b.

Although Resnick's prediction method is faster than matrix factorization, it does not guarantee to generate rating prediction for every item. The reason behind this is the possibility that no other user has rated the desired item or has not revealed her rating to the current user. The sparser the dataset, the less predictions will be made.

Chapter 5

Experiment Results

5.1 Dataset description

As the popularity of social networks increases, so does the number of available data mined for the purpose of social recommendations. Nowadays, great varieties of datasets are made public, but choosing a relevant one is an important step in assessing the feasibility of our recommender system. Given the fact that we require some sort of trust between users in order to compute the secondary privacy, we opted to use the *Epinions* dataset [1] to experiment on with our social recommender system.

This dataset is extracted from a who-trust-whom online social network of a general consumer review website *Epinions* where members can decide whether to *trust* each other or not. The dataset contains 40,163 users that rated 139,738 different items with a number from 1 to 5, providing a total of 664,824 reviews. A user can be regarded by others either as trusted (denoted by trust value of 1) or not known (denoted by trust value of 0). The density of the ratings is about 0.01184% and the average ratings per user is 16.55, concluding that this dataset is very sparse. One interesting characteristic is that the distribution of ratings in the network is rather uneven and it follows fat-tail power-law distribution, as it can be seen in Fig.5.1. Accordingly, a large number of users have rated few items (under 10), while very few users have more than 60 ratings. This type of distribution has a direct effect on the role privacy plays in the final results of our experiments. Because the majority of users have under 10 ratings in their profiles, decreasing privacy towards trusted users only results in revealing a small number of ratings. Thus, in the case of some users, the difference in the quantity of exposed ratings can be so small that it does not have a noticeable influence on the obtained recommendations. We consider this skewed distribution a limiting factor on the implication of privacy in the final predictions.

[1]Available at: `http://www.trustlet.org/wiki/Epinions_datasets`

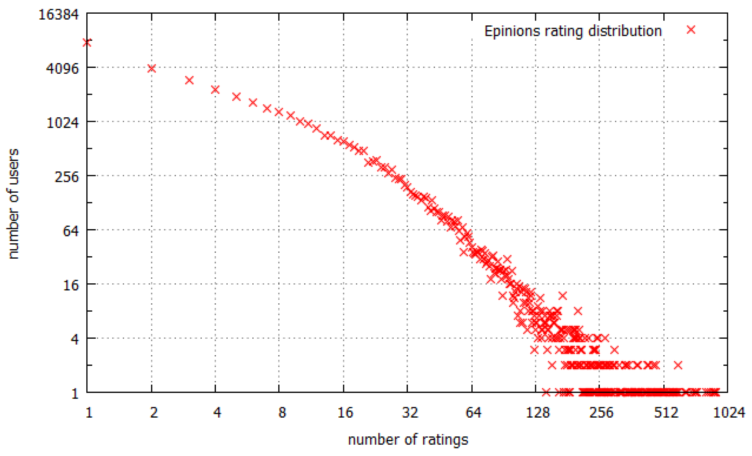


Figure 5.1. Distribution of ratings vs. users in *Epinions* dataset

5.2 Metrics

In order to assess the effectiveness of our recommender system, we need to estimate how close our predicted ratings are to the user's preferences. To do so, we split the dataset into two segments: a training set R^t that serves as the input data for our system and a hidden set (test set) R^h to which we will compare our predictions.

We use the mean absolute error (MAE) to measure the quality of our estimated ratings with respect to the hidden set:

$$MAE = \frac{\sum |R^h_{ij} - R_{ij}|}{N}$$

where R^h_{ij} is user's i rating of item j from the hidden set, R_{ij} denotes the estimated rating for item j by our method for the same user, and N is the number of ratings from the hidden set for which we can offer a prediction.

In addition to the accuracy of our recommender, we are also interested in the *coverage* it provides: the fraction of the hidden ratings for which we can provide an estimate.

5.3 Results

5.3.1 Performance Comparison

In the first dimension, we compare the performance of our method (which we refer to as *secondary privacy method*) with two other competing methods: Social Matrix Factorization [17] and Resnick [28]. The performance is defined in terms of accuracy of provided predictions (i.e. MAE metric) and the achieved coverage. In this experiment we compare the results of each method across different training (and test)

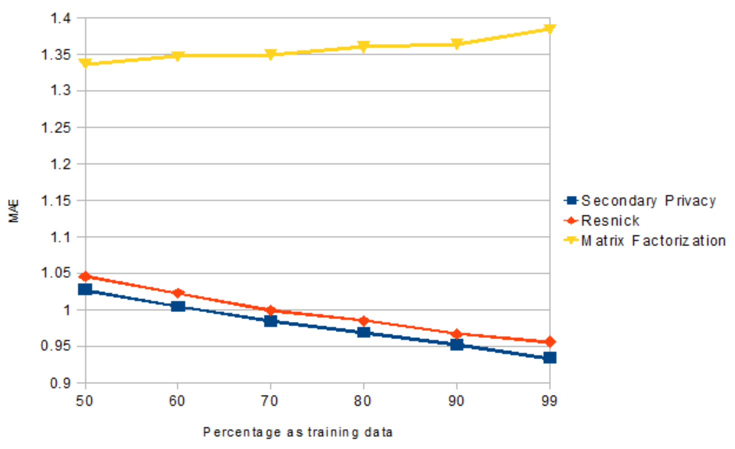


Figure 5.2. Comparison of MAEs for the three evaluated methods with initial privacy 20%

datasets, while the assumed trust threshold and initial settings are kept constant. Despite the fact that our approach shares some aspects with both methods, it is highly dependent on privacy awareness.

In the absence of any default (initial) privacy setting in *Epinions* dataset, we considered the arbitrary default privacy setting $P_0 = 20\%$ for this experiment. In addition, following the idea that we regard a user as trusted if our trust in her is higher than average, we use the average of all trusts values (those accommodated in inferred trust matrix in Fig.4.1) as the trust threshold:

$$T_{th} = \frac{\sum_{i=1}^{m} \sum_{j=1}^{m} T_{ij}}{m^2} \tag{5.1}$$

While Fig.5.2 accommodates the performance of these three methods across different sizes of training dataset, the respective coverage measures are shown in Fig.5.3. From the results presented in Fig.5.2, we can conclude that our approach has considerably better accuracy than the other two methods. At the first glance, the lower MAE provided by the Secondary Privacy over Resnick method can be puzzling. However, if we take a closer look at how the secondary privacy is inferred, the improvement in the MAE can be explained by the fact that even though we have fewer revealed ratings at our disposal, these ratings are personalized and more relevant for the target user. This is due to the effect of trust which is a mixture of social trust and similarity driven trust. In other words, trusted individuals are not only socially trusted, but are also exposing quite similar ratings, hence the error rate is decreased. In contrast, both Resnick and Matrix Factorization methods do not enjoy such personalized view for their users and take into account all ratings, including those from non-similar users, to make predictions. The advantage of our method over Resnick is due to the fact that a larger subset of similar users is taken into account by filtering out less similar users using a combined trust metric.

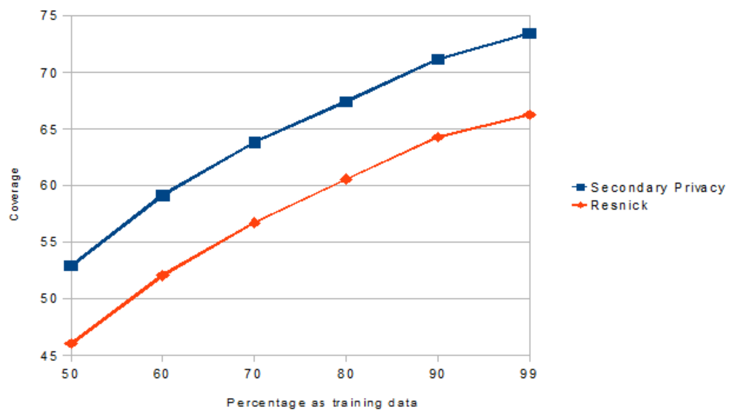


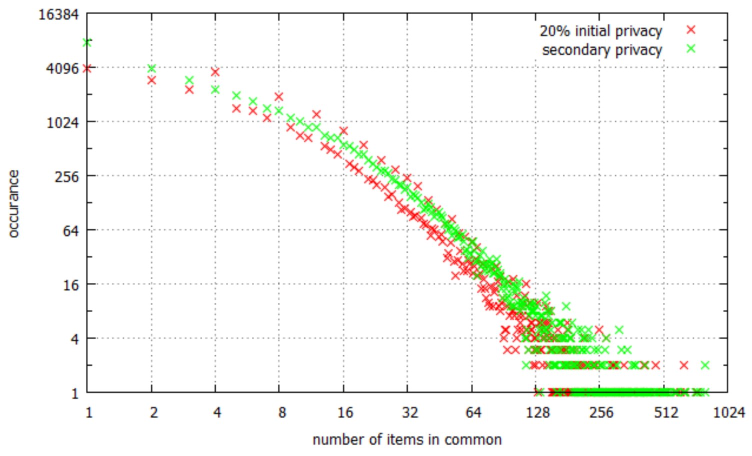
Figure 5.3. Comparison of gained coverage between Resnick and Secondary methods with initial privacy 20%

This results in more accurate predictions. Moreover pure matrix factorization is additionally punished by the skewed pair-wise trust distribution between users in this dataset, as can be seen in Fig.5.5, and the designated trust threshold which all together prevent it form taking full advantage of the trust factor in making recommendation.

The slight drop in MAE with the increase of the training set is intuitive: the more information about a social network we get, the better the predictions will be. Both Resnick and Secondary Privacy methods follow the same trend because the final predictions are decided in both cases using Resnick's formula with Pearson similarity, although on different rating pools.

Fig.5.3 presents the respective coverage measures. We omitted to illustrate the matrix factorization technique because it inherently achieves a 100% coverage [17]. It can be seen that the Secondary Privacy approach constantly has a better coverage by nearly 10% and both methods' coverage increases with the percentage of data used for training. This is because the more ratings available, the higher the chance to find a user who rated the item we are interested in predicting an estimated rating for.

Because we use Resnick's method to predict ratings, our results are influenced, both in terms of coverage and accuracy, by the subset of similar users. Since we employ Pearson correlation as a measure of similarity, this would translate into the subset of users that have at least one rated item in common with the current user. The larger this subset is, the more user opinions we consider when making predictions. This results in more accurate recommendations. In order to see how different initial privacy settings could affect our results, we plotted the distribution of the size of this subset for the two extreme cases of initial privacy: 20% and 80% before and after applying our privacy inference model. The plots are presented in



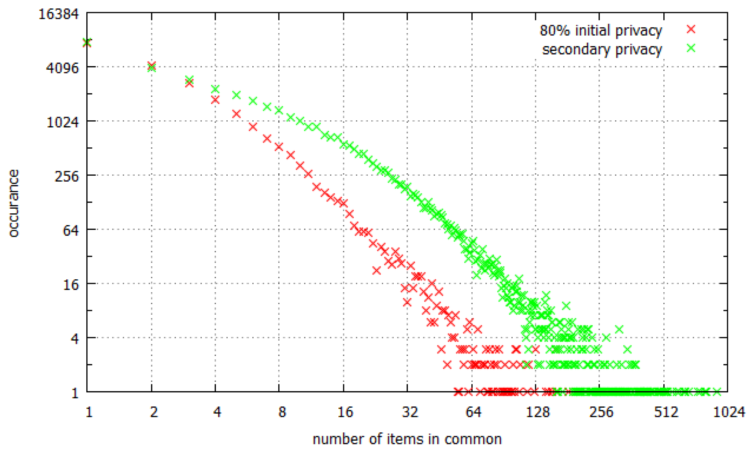


Figure 5.4. Distribution of commonly rated items for 20% (top) and 80% (bottom) initial privacy, with 90% training data

Fig.5.4. Although more relevant for $P_0 = 80\%$, in both cases we notice an increase in the size of these subsets. Hence, it can be concluded that our method yields better recommendation accuracy as the initial privacy setting is increasing.

5.3.2 Analysis of Trust Threshold

In this experiment we investigate how the trust distribution resulted from matrix factorization can influence the choice of a relevant trust threshold. We start by analyzing the trust matrix inferred after applying matrix factorization on a training set consisting of 90% of the dataset with an initial privacy of 20%. The distribution of different bounds of pair-wise trust level with respect to percentage of involved users is presented in Fig.5.5. It can be seen that the majority of users (51.23%) are regarded with very low trust (between 0 and 0.1). This is quite expected due to the sparsity of our dataset: only 487,181 issued trust statements for 40,163 users.

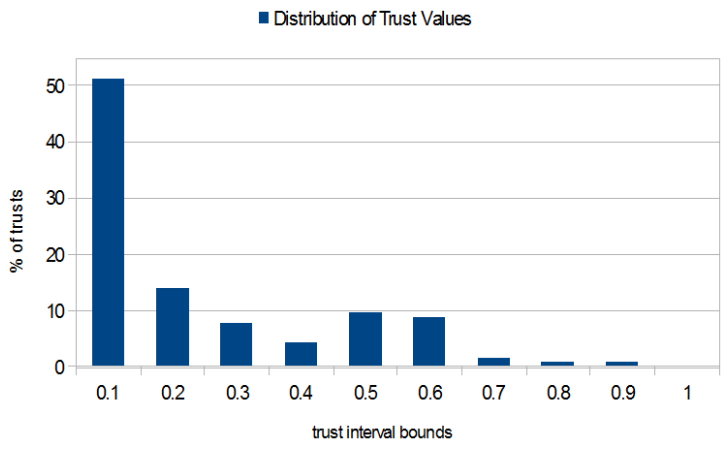


Figure 5.5. Distribution of trust values resulted from applying matrix factorization (initial privacy of 20%, 90% training data). Horizontal axis shows trust intervals while the vertical axis denotes the percentage of users involved in each trust interval

Moreover, due to the significantly low density of the ratings (around 0.01184%), the estimated trust measures for missing cases are also quite low. On the other hand, a small percentage of users (around 3%) have a trust higher than 0.6. However, when a user rates on average 16.55 items out of a total of 139,738 possible choices, it is hardly possible to find other users who rated the same items and provided similar ratings.

We experimented with varying the trust threshold by 0.1 starting with 0.1 and up to 0.9. However, the difference in results was insignificant both in terms of MAE (which differed only on the fifth decimal) and coverage (the best case provided 10 more predictions than the worse). This behavior is a direct result of the pair-wise trust distribution among the users: when we start with a trust threshold of 0.1 we consider the majority of users as untrusted, according to Fig.5.5. Furthermore, given our method of computing the secondary privacy, we reward the most trusted users with lowest privacy. But this type of users comprise only a small percentage of the total; hence, their influence becomes minimal when varying the trust threshold in this manner. This effect is amplified by the sparsity of the dataset: it might happen that even though we decrease our privacy towards a specific user, we still do not decrease it enough to reveal an entire rating.

Following this reasoning, it can be concluded that the setting of a suitable trust threshold needs to be done with respect to the dataset characteristics. To do so, we need to include the pair-wise trust distribution in the process of deciding which trust threshold is optimal. Following the intuition that a user would have an initial privacy P_0 towards the average trusted users, we consider the trust threshold to be the average of all the trusts in the network. In our case, this allowed us to incorporate the high percentage of users with trust lower than 0.1 in the first experiment presented before. We compared the results (both MAE and coverage measures) for trust thresholds greater than 0.1 and for trust threshold obtained based on mean of all pair-wise trust values (equation 5.1). We observed that the latter setting yields better results than the previous attempts at choosing a threshold unaware of the characteristics of the dataset: the MAE is lowered by 0.02 and the coverage increased by 2%.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

Social network users are more likely to expose personal data to other users who are both trusted and similar. Based on this premise and inspired by the direct network effect, we proposed a privacy model for user profiles and showed how this model can be applied by building a framework for generating privacy aware recommendations. Although we provided an example of implementation, the components of our framework are highly flexible and can be customized according to the requirements in hand. We employed a social matrix factorization technique to deal with the sparsity of social relations and infer the trust between each pair of users. In turn, this allows us to generate a privacy measure between all users pairwise. Our main contribution is the computation of secondary privacy in a way which is both adaptive to different datasets and consistent with the intuition that trust varies inversely proportional with privacy.

Our evaluation over Epinions dataset showed that, based on the proposed privacy inference model, our approach yields better results than matrix factorization in terms of accuracy and than Resnick's method, both in terms of accuracy and coverage, while providing users with a flexible privacy scheme. However, deciding on appropriate trust threshold as a means to differentiate between trusted and untrusted individuals is highly dataset dependent. Moreover, making an appropriate decision on this matter can improve the results of our method drastically.

6.2 Future Work

The main priority of our future work is to study how selective concealment of rating instead of random hiding can effect our analysis results.

The current work is performed based on hypothesis that dependency between trust and privacy can be depicted as an exponential function, but other models of this dependency can also be adopted depending on, for example, dataset characteristics and use-case scenario. Thus, as one of our future work directions, we aim to study and suggest different privacy-trust correlation models based on the requirements in hand.

Currently, we employ a global trust-privacy dependency, but a possible new dimension could envision expressing this relation for each user independently. This would provide a more accurate representation of privacy, not influenced as much by the dominant users in the network.

Bibliography

[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. Knowl. Data Eng.*, 17(6):734–749, 2005.

[2] E. Aïmeur, G. Brassard, J. M. Fernandez, and F. S. M. Onana. Alambic : a privacy-preserving recommender system for electronic commerce. *Int. J. Inf. Sec.*, 7(5):307–334, 2008.

[3] M. M. Anwar, J. Greer, and C. A. Brooks. Privacy enhanced personalization in e-learning. In *Proceedings of the 2006 International Conference on Privacy, Security and Trust: Bridge the Gap Between PST Technologies and Business Services*, PST '06, pages 42:1–42:4, New York, NY, USA, 2006. ACM.

[4] J. S. Breese, D. Heckerman, and C. M. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *UAI*, pages 43–52, 1998.

[5] N. Dokoohaki, C. Kaleli, H. Polat, and M. Matskin. Achieving optimal privacy in trust-aware social recommender systems. In *SocInfo*, pages 62–79, 2010.

[6] T. DuBois, J. Golbeck, and A. Srinivasan. Rigorous probabilistic trust-inference with applications to clustering. In *Web Intelligence*, pages 655–658, 2009.

[7] T. DuBois, J. Golbeck, and A. Srinivasan. Predicting trust and distrust in social networks. In *SocialCom/PASSAT*, pages 418–424, 2011.

[8] S. M. Feng Gao, Jingsha He. Research on the relationship between trust and privacy in network environments. *International Journal of Digital Content Technology and its Applications*, 2011.

[9] J. L. Funk. Direct network effects, small-world networks, and industry formation. *Telecommun. Policy*, 33(5-6):241–252, June 2009.

[10] J. He, W. W. Chu, and Z. V. Liu. Inferring privacy information from social networks. In *Proceedings of the 4th IEEE international conference on Intelligence and Security Informatics*, ISI'06, pages 154–165, Berlin, Heidelberg, 2006. Springer-Verlag.

[11] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *SIGIR*, pages 230–237, 1999.

[12] Y. Koren, R. M. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8):30–37, 2009.

[13] J. Leskovec, D. P. Huttenlocher, and J. M. Kleinberg. Predicting positive and negative links in online social networks. In *WWW*, pages 641–650, 2010.

[14] N. Li, M. Najafian Razavi, and D. Gillet. Trust-aware privacy control for social media. In *Proceedings of the 2011 annual conference extended abstracts on Human factors in computing systems*, CHI EA '11, pages 1597–1602, New York, NY, USA, 2011. ACM.

[15] G. Linden, B. Smith, and J. York. Industry report: Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Distributed Systems Online*, 4(1), 2003.

[16] C. Liu, J. T. Marchewka, J. Lu, and C.-S. Yu. Beyond concern - a privacy-trust-behavioral intention model of electronic commerce. *Information & Management*, 42(2):289–304, 2005.

[17] H. Ma, H. Yang, M. R. Lyu, and I. King. Sorec: social recommendation using probabilistic matrix factorization. In *CIKM*, pages 931–940, 2008.

[18] P. Massa and P. Avesani. Trust metrics in recommender systems. In *Computing with Social Trust*, pages 259–285. 2009.

[19] P. Massa and B. Bhattacharjee. Using trust in recommender systems: An experimental analysis. In *iTrust*, pages 221–235, 2004.

[20] F. McSherry and I. Mironov. Differentially private recommender systems: Building privacy into the netflix prize contenders. In *KDD*, pages 627–636, 2009.

[21] J. C. Meza. *Wiley Interdisciplinary Reviews: Computational Statistics*, chapter Steepest Descent, pages 719–722. John Wiley & Sons Inc., 2010.

[22] S. E. Middleton, N. Shadbolt, and D. D. Roure. Ontological user profiling in recommender systems. *ACM Trans. Inf. Syst.*, 22(1):54–88, 2004.

[23] S. Mokarizadeh, N. Dokoohaki, M. Matskin, and P. Küngas. Trust and privacy enabled service composition using social experience. In *I3E*, pages 226–236, 2010.

[24] J. O'Donovan and B. Smyth. Trust in recommender systems. In *IUI*, pages 167–174, 2005.

[25] J. S. Olson, J. Grudin, and E. Horvitz. A study of preferences for sharing and privacy. In *CHI Extended Abstracts*, pages 1985–1988, 2005.

[26] N. Ramakrishnan, B. J. Keller, B. J. Mirza, A. Y. Grama, and G. Karypis. Privacy risks in recommender systems. *IEEE Internet Computing*, 5(6):54–62, Nov. 2001.

[27] A. M. Rashid, I. Albert, D. Cosley, S. K. Lam, S. M. McNee, J. A. Konstan, and J. Riedl. Getting to know you: learning new user preferences in recommender systems. In *IUI*, pages 127–134, 2002.

[28] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *CSCW*, pages 175–186, 1994.

[29] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors. *Recommender Systems Handbook*. Springer, 2011.

[30] J. Rohlfs. A theory of interdependent demand for a communication service. *Bell Journal of Economics*, 1974.

[31] J. B. Schafer, D. Frankowski, J. L. Herlocker, and S. Sen. Collaborative filtering recommender systems. In *The Adaptive Web*, pages 291–324, 2007.

[32] J.-M. Seigneur and C. D. Jensen. Trading privacy for trust. In *iTrust*, pages 93–107, 2004.

[33] R. Shokri, P. Pedarsani, G. Theodorakopoulos, and J.-P. Hubaux. Preserving privacy in collaborative filtering through distributed aggregation of offline profiles. In *RecSys*, pages 157–164, 2009.

[34] R. R. Sinha and K. Swearingen. Comparing recommendations made by online systems and friends. In *DELOS Workshop: Personalisation and Recommender Systems in Digital Libraries*, 2001.

[35] H. T. Tavani and D. Arnold. Trust and privacy in our networked world. *Information*, 2011.

[36] E. Vozalis and K. G. Margaritis. Analysis of recommender systems algorithms. In *6th Hellenic European Conference on Computer Mathematics and its Applications (HERCMA-2003)*, 2003.

[37] A. Zarghami, S. Fazeli, N. Dokoohaki, and M. Matskin. Social trust-aware recommendation system: A t-index approach. In *Web Intelligence/IAT Workshops*, pages 85–90, 2009.

[38] C.-N. Ziegler and J. Golbeck. Investigating interactions of trust and interest similarity. *Decision Support Systems*, 43(2):460–475, 2007.

[39] C.-N. Ziegler and G. Lausen. Analyzing correlation between trust and user similarity in online communities. In *iTrust*, pages 251–265, 2004.