

TARTU UNIVERSITY
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
INSTITUTE OF COMPUTER SCIENCE
Computer Science

Nguyen Hoang Anh
**Selection of Trust Mechanism in
Recommender Systems**

Master's Thesis (20 AP)

Supervisors:

Peeter Laud
prof. Sasu Tarkoma

Author:.....", June 2009
Supervisor " June 2009

Allowed to defend
Professor " June 2009

TARTU 2009

Contents

Acknowledgements	5
Abstract	7
Abbreviations and Acronyms	8
1 Introduction	9
1.1 Background	9
1.2 Problem statement	12
1.3 Structure of the thesis	13
2 Background	14
2.1 Collaborative filtering recommender systems	14
2.2 Weaknesses of recommender systems	18
2.3 Trust-aware recommender systems	19
2.3.1 Defining trust	20
2.3.2 Trust and user similarity	21
2.3.3 Trust metrics	21
2.3.4 Using trust to alleviate recommender systems' weak- nesses	22
3 Collaborative filtering algorithms	24
3.1 LSI/SVD	24
3.1.1 Dimension reduction in recommender systems	24

3.1.2	LSI/SVD	25
3.1.3	The algorithm	26
3.2	Trust inferences	27
3.2.1	Trust computation	27
3.2.2	Trust propagation	28
3.2.3	The algorithm	29
3.3	EigenTrust	32
3.3.1	Local trust scores	32
3.3.2	Global trust scores	33
3.3.3	The algorithm	33
4	Methodology	35
4.1	The dataset	35
4.2	Evaluation Metrics	40
5	Results and Evaluation	42
5.1	LSI/SVD	44
5.1.1	Experimental steps	44
5.1.2	Results	46
5.1.3	Discussions	47
5.2	Trust Inferences	47
5.2.1	Experimental steps	47
5.2.2	Results	48
5.2.3	Discussions	49
5.3	EigenTrust	50
5.3.1	Experimental steps	50
5.3.2	Results	50
5.3.3	Discussions	51
5.4	Discussions	51
6	Security and privacy issues in recommender systems	53

6.1	Privacy in recommender systems	54
6.1.1	Privacy risks of information	54
6.1.2	Privacy preserving for recommender systems	55
6.2	Security of recommender systems	56
6.2.1	Attack motivations	56
6.2.2	Required knowledge for attacks	57
6.2.3	Defending against attacks	57
7	Conclusions	59
	Resume (in Estonian)	61
	Bibliography	63

Acknowledgements

I would like to thank professor Peeter Laud at UT for supporting and giving me the permission to start this thesis. I am grateful to professor Sasu Tarkoma at TKK for encouraging me to get my work done every time I met him to talk about my concerns for the scope of the topic.

A special thanks goes to my instructor Mozhgan Tavakolifard at NTNU for the topic and for her helpful instructions throughout this thesis work.

I very much appreciate the opportunity European Commission gave me to participate in NordSecMob program. I would like to thank NordSecMob consortium for their constant support during my two years of master studies. Without their support, my life abroad shouldn't be easy.

I am eternally grateful to my parents and my sisters for standing by me all the time. They have always encouraged me to do my best and follow things that I am passionate about. I am also very grateful to my friends for supporting and listening patiently to my complaints whenever I felt down.

Espoo June 11th 2009

Nguyen Hoang Anh

Abstract

Recommender Systems (RS) have emerged as an important response to the so-called information overload problem. They enable users to share their opinions and benefit from each other's. Recommender algorithms are best known for their use on e-commerce Web sites to help users find products they would appreciate from huge catalogues. The products may vary from books (e.g., Amazon.com), movies (e.g., Netflix), photographs (e.g. Flickr.com), or web sites (e.g., del.icio.us)...

To date, there are basically two types of recommending techniques used on the Internet: content-based - also dubbed item-item - recommenders and collaborative filtering recommenders. The interesting point of the latter is that, knowledge about items is not required, it is based only on the ratings of the user community. Therefore, they promise to scale well to large databases. The traditional collaborative filtering techniques are able to provide high-quality recommendations by leveraging the preferences of similar users. However, recent researches have suggested that the traditional focus on user similarity may not be sufficient. Additional factors, especially trust may have an important role when it comes to making recommendations.

In this thesis, we study the different algorithms and the use of trust to improve the performance of collaborative filtering recommender systems. Our evaluation on MovieLens dataset shows that the dimensionality reduction method that uses LSI/SVD technique helps in providing better quality of recommendations. Trust also has positive impact on overall prediction error rates, however, global trust metrics may not be appropriate for trust-aware recommender systems due to their non-personalized nature.

Keywords: Recommender Systems, Collaborative Filtering, Trust, LSI, SVD, EigenTrust, Trust Inference.

Abbreviations and Acronyms

CF	Collaborative Filtering
LSI	Latent Semantic Indexing
MAE	Mean Absolute Error
RS	Recommender System
SVD	Singular Value Decomposition

Chapter 1

Introduction

1.1 Background

People nowadays are logging into the net not only to surf or browse, but also to produce or share pieces of information. As the result, there's an excess amount of information provided, making the information processing tasks more difficult for the individual. In that context, Recommender Systems (RS) [23] have emerged as an important response to the so-called information overload problem. They enable users to share their opinions and benefit from each other's. After the era of search, now the web is entering the era of discovery. The difference here is: search is what one does when s/he looking for something; discovery is when something wonderful that one didn't know existed, or didn't know how to ask for, finds one.

Recommender systems are widely used by e-commerce sites to help users find products they should appreciate from huge catalogues. The products vary from books (e.g., Amazon.com), movies (e.g., Netflix), photographs (e.g. Flickr.com), or web sites (e.g., del.icio.us). The products can be recommended based on the top overall sellers on a site, based on the users' interests and/or their interactions history on the sites. Recommender systems benefit not only customers, but also e-commerce sites. According to [26], recommender systems may enhance e-commerce sales in three ways:

- **Browsers into buyers:** Recommender systems can help customers find products they wish to purchase on a site that they may miss, because they often just look over the site and don't have time or patience

to browse into every catalogue.

- **Cross-sell:** Recommender systems improve cross-sell by suggesting additional products for the customer based on the products in her shopping cart at the checkout step.
- **Loyalty:** Recommender systems improve loyalty by creating a value-added relationship between the site and the customer. Sites invest in collecting their customers' preferences, and then use recommender systems to aggregate that information in order to deliver personalized services to customers. The result is that customers will stick to the site because of good recommendations that match their needs. In return, they will give more information, and the recommendation quality will get better and better. The more a customer uses the recommendation system - the more they teach it what they want - the more loyal they are to the site.

To date, there are basically two types of recommending techniques used on the Internet [19]: content-based - also dubbed item-item - recommender systems and collaborative filtering (CF) recommender systems. The content-based approach uses rich content descriptions of the recommending items for finding then suggesting to the user items that are similar to their preferences or previous selections. For instance, a movie recommender system will rely on information such as genre, actors, director... then match this against the user's preferences in order to select a set of promising movie recommendations.

To achieve this, content-based RSs need to rely heavily on humans to create and maintain attributes of the items that are not machine-parsable such as movies and songs (some machine-parsable such as news or papers can be automatically tagged). Evidently, this activity is expensive, time-consuming, error-prone, and highly subjective.

Pandora¹, one of the most popular music recommendation and discovery services on the Internet today, bases its recommendations on data from the

¹<http://www.pandora.com>

Music Genome Project². The Music Genome Project assigns up to 400 attributes to every song. Attribute assigning has to be done by a bunch of music experts and the process can take up to half an hour per song. While the results of this method are often great, this approach simply doesn't scale very well and Pandora's database is somewhat limited.

As an alternative, collaborative filtering (CF) approach provides a reasonable solution. A collaborative filtering recommender system collects opinions from users by asking them to rate items. When asked for a recommendation by an user, the system tries to identify the similar users and suggests the items these users have liked in the past. By similar users, we mean the users who share similar or highly correlated rating histories with the target user.

The interesting point of this approach is that, knowledge about items is not required, it is based only on the ratings of the user community. Because of this, collaborative filtering recommender systems can be applied to any kind of items: papers, news, web sites, movies, songs, books, locations of holidays, stocks...

In addition, since collaborative filtering techniques don't require recommender systems' designers for tagging content themselves, they promise to scale well to large databases. Some of the well known successful e-commerce sites that make use of collaborative filtering techniques are Amazon and Netflix.

Collaborative filtering offers three major advantages over content-based filtering [13]:

- It supports to filter items whose content is not easily analyzed by automated processes. This is because collaborative filtering asks humans to determine the relevance, quality, and interests of items. The burden on the system's designers of producing input data is now distributed to users of the system.
- It is capable of filtering items based on both quality and taste. Because it can measure how well an item meets a user's need or interests, instead of just taking into consider the simple content like in content-based ap-

²<http://www.pandora.com/mgp.shtml>

proach. For example, a content-based search of Recommender Systems could retrieve all the articles related to Recommender Systems, but by combining content-filtering with collaborative filtering, a search could return only those relevant articles that are highly rated by users.

- It is capable of providing serendipitous recommendations, i.e, recommending items that the target user would like, but do not content that the user was expecting. Content-based filtering can help finding relevant items about a specific content, but collaborative filtering can predict and return all the items that would meet users' eclectic tastes.

In the rest of the paper, we focus on recommender systems based on collaborative filtering techniques.

1.2 Problem statement

Although classic CF recommender systems are successful in providing high quality recommendations, they face problems such as the cold start problem, the scalability and the vulnerability to attacks as described in Section 2.2. The vulnerability to attacks comes from the centralized nature of a collaborative filtering system and the fact that there are always users that have malicious intent and want to subvert the system. To do this, the attacker just need to simply create a user profile with very similar preferences to that of the targeted user and thus he becomes highly influential to the victim.

The cold start problem is the main issue hindering CF recommender systems to be adopted and further developed. It is due to the low number of ratings to items that new users contribute to the system, the system could not provide high quality recommendation, or even could not provide any recommendation to the user.

Meanwhile, trust [4] is considered to be very important in e-commerce. Trust is the basis for partnering because trust is the central component in effective working relationships. There is the fact that people tend to believe recommendations from people they know and trust such as their friends and family-members, other than strangers. So what is the practical usage of trust

in current collaborative filtering approaches?

This thesis studies collaborative filtering-based recommender systems and methods to improve their performance and alleviate the existing challenges using trust and dimensional reduction techniques.

1.3 Structure of the thesis

The rest of the thesis is structured as follows: Chapter 2 is about the background knowledge for the thesis. Chapter 3 focuses on three algorithms used in CF recommender systems. Chapter 4 describes the methodology for our experiments. Chapter 5 elaborates the experimental steps and results, then based on that, evaluation is done. Chapter 6 discusses the concerns about security and privacy in recommender systems. Finally, Chapter 7 makes conclusions.

Chapter 2

Background

2.1 Collaborative filtering recommender systems

It is often necessary to make choices without sufficient personal experience of the alternatives. In everyday life, we rely on recommendations from other people either by word of mouth, recommendation letters, movie and book reviews printed in newspapers, or general surveys.

In this era of information overload, it is very difficult to find items that would suit our needs from huge catalogues. Recently, we have become increasingly dependent on the Internet to help us in making such decisions. And recommender systems [23] have emerged as a core tool to assist and augment the natural social recommending process. Online recommender systems provide users with recommendations of products that they might appreciate, based on their past rating profiles and their history of purchase or interests.

In e-commerce web sites, recommender systems are widely employed to help their customers select suitable products that meet their personal needs. Recommender system architecture is presented in Figure 2.1.

To date, two typical types of filtering approaches have been used to produce recommendations: content-based and collaborative filtering. The former makes recommendations by analyzing the similarity between the contents of the items that are ready to be recommended and those that have previously been marked as liked by the user. For instance, a recommender system on Amazon.com (www.amazon.com) suggests books to customers based on other books the customers have told Amazon they like. Another recom-

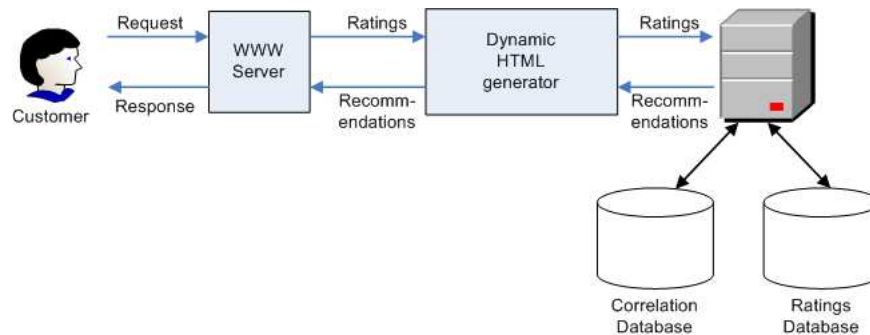


Figure 2.1: Recommender system architecture

mender system on CDnow (CDnow.com) helps users choose CD based on the CDs that they liked in the past.

The latter, collaborative filtering approach, makes recommendations by matching the target user's preferences to other users. This is the most successful recommender system technology to date, and it is used in many of the largest e-commerce web sites, including Amazon.com and CDnow.com.

Most collaborative filtering based recommender systems try to identify users that have relevant interests and preferences by calculating their similarities to the target user (those similar users are called neighbors). The idea behind this approach is that, when searching for recommendations, the target user can probably trust the other users' opinions if they share the same or relevant interests. The main value of a recommender system lies in its information aggregation method and its ability to match the recommendations with appropriate people seeking information.

The classic input to a CF algorithm is a user-item matrix that has a row for every user and a column for each of the items. Each entry of the matrix is the user's rating of the item. Table 2.1 gives an example of such a matrix.

	Matrix Reloaded	Lord of the Ring 2	Titanic	La vita e bella
Alice	2	5		5
Bob	5		1	3
Carol		5		
Dean	2	5	5	4

Table 2.1: The user-item matrix as classic input for CF

A standard CF algorithm performs 3 steps as follows [18, 13]:

- It computes the similarities between users. To date, different techniques have been proposed for this task. Of which, Pearson correlation coefficient is the best performing and most used in recommender systems (e.g. original GroupLens system)[13]:

$$w_{a,u} = \frac{\sum_{i=1}^m (r_{a,i} - \bar{r}_a)(r_{u,i} - \bar{r}_u)}{\sqrt{\sum_{i=1}^m (r_{a,i} - \bar{r}_a)^2} \sqrt{\sum_{i=1}^m (r_{u,i} - \bar{r}_u)^2}} .$$

where m is the number of items rated by both a and u , $r_{a,i}$ is the rating given by user a to item i and \bar{r}_a is the mean of ratings of a .

The coefficient is in $[-1,1]$, and it can be computed only if there are items rated by both the users. Moreover, if two users only have rated one item in common, then the coefficient is not meaningful. Therefore, the correlation coefficient is identified only for two users who share at least two co-rated items.

Proposed alternatives are Constrained Pearson correlation coefficient, Spearman correl and cosine similarity [13]. In cosine similarity method, we view different users as vectors in the n -dimensional space of items, their n coordinates are the rating values of users to n items (0 if no such rating exists). Then the cosine similarity between two users can be measured by computing the cosine of the angle between the corresponding user vectors:

$$w_{a,u} = \cos(a, u) = \frac{\sum_{i=1}^n r_{a,i} r_{u,i}}{\sqrt{\sum_{i=1}^n r_{a,i}^2} \sqrt{\sum_{i=1}^n r_{u,i}^2}} .$$

The similarity values are then used to weight the ratings given by other users.

- The final step is to produce either a prediction - a numerical value representing the predicted rating of the target user for an item that s/he has not yet rated, or a recommendation - a list of top-N items that the target user might be interested in.

The predicted rating that user a might give to item i is the mean rating of a plus the weighted sum of deviation from mean rating for every user where the weight is the user similarity with user a . Predictions can be done using Resnick's formula [23]:

$$p_{a,i} = \bar{r}_a + \sum_{u=1}^n w_{a,u}(r_{u,i} - \bar{r}_u) .$$

The intuition behind this formula is, if user A is very similar to user B, then the opinions of user A are given importance when creating a recommendation for user B.

As can be seen from the formula, the prediction is dependent on the number of correlated entities n and becomes noisy and unreliable when few entities are involved. Therefore, higher accuracies become possible with the combination of more entities in the calculation. The prediction is personalized for customer a . There are also some *naive non-personalized* prediction methods, for example, the technique to obtain prediction by taking the average ratings of items being predicted over all users [13]. We test some such *naive non-personalized* techniques in Section 5.

It is possible to consider only the k most similar users or all the users.

In a recommender system, a prediction engine collects all the ratings and uses collaborative filtering technique to create predictions. Most prediction engines also provide a recommendation mode, where the prediction engine simply returns the top N highest predicted items for the target user. Latency of a prediction request must be less than 1 second and prediction engines must often support throughput of several hundred prediction requests per second [13].

2.2 Weaknesses of recommender systems

Collaborative filtering recommender systems has gained success in several domains, but it also has been exposed to limitations [24, 18] listed as follows.

- **Sparsity and cold-start problem**

In practice, the database of customers and products in major e-commerce recommendations is very large. Even active customers may have rated just a few of the the total number of products available in the database, and vice versa, even very popular products may have been rated by only a few of the total number of customers available in the system. This problem, which is referred to as sparsity problem, causes a major negative impact on the performance of recommender systems using neighborhood-based algorithms.

Because of sparsity, it is possible that the similarity between two users cannot be defined (if they haven't co-rated at least two products, see Section 2.1). As the result, the collaborative filtering algorithm is unable to make many product recommendations for a particular user. Even when the computation of similarity is possible, the accuracy might be poor because the users have too few ratings in common. Therefore, a new customer has to rate a sufficient number of products in order to get reliable recommendations from the system - this is referred to as cold-start problem.

The sparsity problem has been identified as one of the main aspects that limits the further development and adoption of collaborative filtering recommender systems, because the system should be able to provide high quality recommendations, even with small rating information, in order to attract new users to continue using the system, and hence, providing more ratings, which would be critical for better recommendation provision.

- **Scalability**

Neighborhood-based recommendation algorithms are computationally expensive. The computation required grows with both the number of customers and the number of products.

- **Synonymy**

Correlation based recommender systems rely on exact matches in names of products. But in real life scenario, different product names can refer to similar objects. If that is the case, the recommender systems can't discover the latent association and treat these products differently.

- **Attacks by malicious insiders**

Recommender systems are often used in e-commerce sites, hence they are attractive targets to attackers. However, subverting standard CF techniques is very easy [21]. The simplest attack is the copy-profile attack: The attacker can easily copy the ratings of target user and the system will think the attacker is the most similar user to target user. By this way, the attacker's future ratings will highly affect the recommendations to target user.

In addition, recommender systems also raise concerns about personal privacy. In general, the more information individuals have about the recommendations, the better they will be able to evaluate the quality of those recommendations. On the other hand, because of the fact that everything people do online tells a little about them, people may not want their habits or rating activities to be publicly known. Anonymous participation or participation under a pseudonym is not a complete solution since some people may desire an attributed credit for their effort.

2.3 Trust-aware recommender systems

Although algorithms based on similarity is effective, there's an intuition that taking trust into account would improve the accuracy of the recommendations and help dealing with the previously mentioned weaknesses. This assumption is based on the observation that in real life, people tend to believe recommendations from people they know and trust such as their friends and family-members, other than strangers.

2.3.1 Defining trust

Before making any computation with trust in recommender systems, it is important to know and to agree on the concept of trust. Because trust is so important, it has been studied extensively in many science research disciplines [4]. In electronic business, trust has been the subject of investigation in order to enable service providers and consumers to know how much reliance to place on each other.

One of the most often cited work on trust is Marsh's PhD dissertation at the University of Stirling [17]. However, in his work, the model for formalizing trust is highly theoretical and based heavily on social and psychological factors, thus it's difficult to realize in computer applications.

Deutsch [7] gives a frequently referenced definition of trust. The author states that trusting behavior occurs when a person (say Alice) encounters a situation where she perceives an ambiguous path. The result of following the path can be good or bad, and the occurrence of the good or bad result is contingent on the action of another person (say Bob). Furthermore, the negative impact of the bad result is greater than the positive impact of the good result. This further motivates Alice to make the correct choice. If Alice chooses to go down the path, she has made a trusting choice. She trust that Bob will take the steps necessary to ensure the good outcome.

Golbeck [10] adopts this definition for her work and explains:

Trust in a person is a commitment to an action based on a belief that the future actions of that person will lead to a good outcome.

The action and commitment does not have to be significant. We could say Alice trusts Bob regarding movies if she chooses to watch a film (commits to an action) that Bob recommends (based on her belief that Bob will not waste her time).

Two types of trust are listed in [30], known as direct trust - directly trusting an entity about a particular subject; and indirect trust - trusting through a trusted entity. The indirect trust raises a question of how one can traverse a whole chain of intermediate trusted entities to decide how much to trust a distant one. This is addressed in Section 2.3.3 and Section 3.2.2.

2.3.2 Trust and user similarity

User similarity is the key component in classic collaborative filtering recommender systems. There also have been several studies on the correlation between user similarity and trust. Sinha et al. [28] showed that users tend to trust systems more if they could recommend items that they previously liked. The lesson here is, a system that demonstrates similarity to the user's preference is considered more trustworthy by the user. Users also tend to prefer recommendations from their friends because they trust them.

The connection between user similarity and trust has been discussed and established in several research. Abdul-Rahman et al. [2] maintained that in a predefined knowledge domain, users develop social connections with people who share similar preferences. Later, Ziegler and Lausen in [31] extended these results and claimed the strong correlation between trust and user similarity in an empirical study of a real online community. Golbeck [9] showed that there was a strong and significant correlation between trust and similarity: the more similar two users were, the greater the trust they would place on each other.

2.3.3 Trust metrics

In a system that has a trust component, the system can ask a user to explicitly rate how much they trust other users. These statements form a trust network. One simple example of a trust network is depicted in Figure 2.2.

Generally, every user has expressed trust values for only a handful other users. On a trust network, it is possible to run a trust metric [31, 32] in order to predict how much a certain user should be trusted by the other users, using the direct trust information on the trust network. The basic assumption of trust metrics is that trust can be propagated in some way. The reason is that you trust your friend more than a stranger, and so, a friend of your friend is possibly more trustworthy than a random stranger. For example, if A trusts B and B trusts D, it is possible to infer something about how much A could trust D (shown as dotted line in Figure 2.2). It is obvious that trust is subjective, as the same user can be trusted by someone and distrusted by someone else. We also weight the trust statements to express different levels of trust on another user. The small values represent low trust, while the large

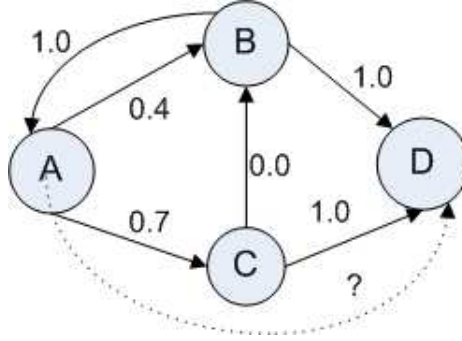


Figure 2.2: Trust network. Nodes are users and edges are trust statements. The dotted edge is one undefined and predictable trust statement

values represent high trust. We assume the range of trust weights is $[0,1]$.

Trust metrics can be categorized into local and global trust metrics. Local trust metrics consider the subjective views of the users and end up predicting different values of trust in other users for every single user. Global trust metrics, instead, predict a global trust value that the community as a whole considers a certain user. PageRank [22] and its variation EigenTrust [15] are the most cited global trust algorithms.

2.3.4 Using trust to alleviate recommender systems' weaknesses

Trust allows us to base recommendations on ratings given by directly or indirectly trusted users. By allowing a user to rate other users, the system can quickly make recommendations using the explicit neighbor set. This means that new users will soon receive good recommendations and thus sparsity and cold-start problem can be addressed.

Moreover, trust metrics have an important property of attack resistance. Trust metrics help the systems not to be overly influenced by malicious users who try to subvert the recommendations quality by abusing their ratings.

To identify malicious users, the concept of distrust can also be relevant. Guha et al. [12] maintains that distrust is at least as important as trust. Studying

the meaning of distrust and how to exploit it is recent topic and requires much work to fully understand it. However, one big challenge of employing distrust is the metrics to propagate it.

If A trusts B, and B trusts C, then by transitive rule, A should trust C at a certain positive degree. However, this transitivity might not hold for distrust. Assume A distrusts B, and B distrusts C, then we can not be sure about the view A has of C. A might trust C if A thinks that B's entire view is so misaligned with A's that anyone B distrusts is probably trusted by A ("the enemy of your enemy is your friend"). However, if A concludes that B's judgements are inferior to A's, then A should strongly distrust C ("don't respect someone not respected by someone you don't respect").

Chapter 3

Collaborative filtering algorithms

This chapter discusses three techniques used in recommender systems. The first technique is to use dimensionality reduction to improve performance for recommender systems. While the second - EigenTrust, and the third - trust inferences techniques make use of trust in improving recommender systems.

3.1 LSI/SVD

3.1.1 Dimension reduction in recommender systems

Classic collaborative filtering has been shown to produce high quality recommendations, but the performance degrades with the number of customers and products, due to the expensive computation of similarities when the number of products in database increases (scalability problem). This weakness raises the needs for new recommender system technologies that can quickly produce high quality recommendations even for very large-scale databases. This section presents one technique called Latent Semantic Indexing (LSI) that uses a dimensionality reduction technique - Singular Value Decomposition (SVD) - to use in recommender systems. The rationale behind this method is that, by reducing the dimensionality of the product space, we can increase density and thereby find more ratings.

3.1.2 LSI/SVD

Deerwester et. al. [6] introduced a new method for automatic indexing and retrieval from document collections. They called this method Latent Semantic Indexing (LSI). This method is proposed to overcome the deficiencies of existing retrieval techniques that try to match words of queries exactly with words of documents. Though LSI is good at what it is doing, it is also time consuming and computationally expensive when applied to large datasets.

Singular Value Decomposition - SVD [11] is a well-known matrix factorization technique for calculating the singular values, pseudo-inverse and rank of a matrix. SVD technique factors an $m \times n$ matrix R into three matrices as the following:

$$R = USV' .$$

Where,

- U is a matrix whose columns are the eigenvectors of the RR' matrix (left eigenvectors). Here matrix R' is the transpose of matrix R .
- S is a matrix whose diagonal elements are the singular values of R .
- V' is the transpose of matrix V , whose columns are the eigenvectors of the $R'R$ matrix (right eigenvectors).

The size of U and V is $m \times r$ and $n \times r$ respectively; r is the rank of the matrix R . S is a diagonal matrix of size $r \times r$. Singular Value Decomposition function is available in matrix calculators/programs like MATLAB.

The matrices obtained by performing SVD are useful because of the important property that SVD provides the best lower rank approximations of the original matrix R . If we reduce the $r \times r$ matrix S to have only k largest diagonal values (k most "meaningful" dimensions) to obtain a matrix S_k of size $r \times r$, $k < r$, and the matrices U and V are reduced accordingly to U_k and V_k of size $m \times k$ and $n \times k$ respectively, then the reconstructed matrix $R_k = U_k S_k V_k'$ is the closest rank- k matrix to R . R_k minimizes the Frobenius norm $\| R - R_k \|$ over all rank- k matrices.

We use SVD in recommender systems for two different purposes. First, we use SVD to capture latent relationships between customers and products. For example, products which are rated by similar users will be near each other in the k -dimensional factor space, even if they have never been rated by the same user. This means that some users which do not share any rated products may none the less be near it in k -space. Second, we use SVD to produce a *low-dimensional* representation of the original customer-product space and then compute the similarities between users in the reduced space. The similarities then can be used to generate the recommendations (predictions) for customers.

3.1.3 The algorithm

1. We start with a customer-product (user-item) ratings matrix and called this matrix R . This matrix is very sparse.
2. We produce the normalized matrix R_{norm} from R .
 - First, we remove sparsity by filling our customer-product ratings matrix using the average ratings for a product.
 - Then we subtract customer average from each rating.

Essentially, after this step, $R_{norm} = R + R_{NPR}$, where R_{NPR} is the fill-in matrix that provides *naive non-personalized* recommendations.

3. We factor the matrix R_{norm} and get a *low-dimension* approximation. This phase follows the steps described by Deerwester et al. [6]:
 - Factor R_{norm} using SVD to obtain U , S , and V .
 - Reduce the matrix S to dimension k .
 - Compute the square-root of the reduced matrix S_k to obtain $S_k^{1/2}$, then compute two resultant matrices: $U_k S_k^{1/2}$ and $S_k^{1/2} V_k'$. These matrices are used to compute the predictions in the next phase.

4. We compute the predictions.

The inner product of the c -th row of $U_k S_k^{1/2}$ and the p -th column of $S_k^{1/2} V_k'$ gives us a prediction score of customer c to product p . We then add the customer average \bar{C} back to get the final rating score of the

customer to the product:

$$C_{P_{pred}} = \overline{C} + U_k S_k^{1/2}(c) S_k^{1/2} V_k'(p) \ .$$

Choosing an optimal value for k is critical to high accurate prediction generation. The value of k should be large enough to capture all the important structure in the matrix, yet small enough to reduce the computation needed. Generally, k is much smaller than the number of products in the database, thus this technique can address the scalability problem. Such a k value can be selected experimentally by trying several different values.

3.2 Trust inferences

When one of your friends gives you a recommendation for a book or a movie, how do you decide how much you trust her judgment? We trust people and information based on a complex mix of past experiences, peer influence, personal motives, and many other social factors. All those factors make it extremely difficult to produce algorithms in order to compute trust and decide its weight to add to Resnick's formula for recommendation generation.

In this section, we present trust inferences as a method for using trust in collaborative filtering recommender systems, which helps alleviating the sparsity problem as previously mentioned in Section 2.2.

3.2.1 Trust computation

One problem when using trust in a recommender system is the fact that users involved in the system usually rate items, rather than give direct trust statements of other users. That means, we need to somehow transform all the information that has been expressed in the form of item ratings into trust scores. And that requires a transformation method.

In order to achieve that, we consider the ratings that users have given to items as the behavioral data required for the composition of their opinions.

In Section 2.3.2, we claimed that similarity can be used as an expression of trust between users: the more similar the two users are, the greater they would trust each other. Based on that assumption, we compute trust scores between users as their similarity in the form of Pearson coefficients, as it has proven to be effective and suitable for this type of application.

Pearson coefficients take values between -1 and 1. The trust between two users are considered higher when the Pearson coefficient is close to 1, and they completely distrust each other when the Pearson coefficient value is -1. A value of 0 would mean that there is no relationship established between the two users.

3.2.2 Trust propagation

As we mentioned in Section 2.2 about weaknesses of collaborative filtering recommender systems, there are cases in which insufficient information makes the recommendation algorithms useless. This is also applicable to trust-based recommender systems due to the lack of direct trust statements. To deal with this problem, we adopt a method of inferring trust between users without direct associations (or no co-rated items/products). Figure 3.1 describes the idea behind trust inferences.

Suppose that users S and I have rated products P_1 and users I , T have rated products P_2 . Classic collaborative filtering will associate user S with user I , and user I with user T , but not user S with user T . However, transitive approach would recognize the associative relationship between user S and user T and infer it indirect association.

In this way, it is possible to infer trust between the source user S and the target user T through the intermediate user I . If we continue this process, trust is propagated in the network and more users are connected, even if they have no co-rated items.

As an intuitive example for trust propagation: if Alice wants to buy an used camera and Dave wants to sell one. Alice does not know Dave and therefore she is not sure whether she should trust his description about the camera. However, Alice knows Bob and Carol, and they both know Dave. In this case, we can apply trust metrics to suggest to Alice to trust (or not) Dave,

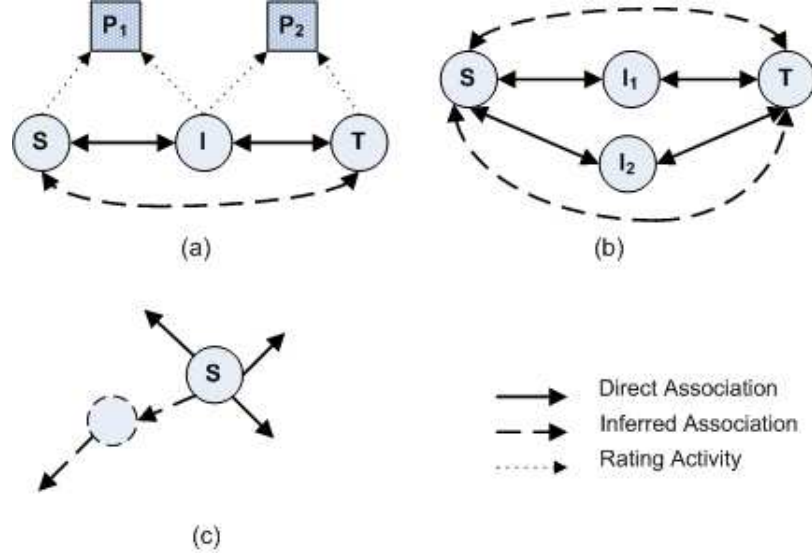


Figure 3.1: Trust inferences

leading to her decision to buy or not buy his camera.

This example of trust propagation also implies the existence of trust paths in the network. Combination of consecutive direct associations between all intermediate users creates a trust path from a source user to a target user. The length of a trust path is the number of associations that one needs to traverse in order to reach the target user. If the target user cannot be reached by the source user, the length of the supposed path is considered infinite. Direct associations are of length 1.

There might exist several trust paths from a source to a target user, resulting in several different inferred trust values between them. To obtain the final inferred trust value between them, we compute the average of inferred trust values from all the trust paths.

3.2.3 The algorithm

1. The inferred trust algorithm starts with computation of direct trust scores between users. While computation of trust in direct associations is based on user-to-user similarity, for length- k associations we need to

adopt a transitivity rule that facilitates the computation of the inferred trust between the source user and the target user.

2. To compute the trust value between user S and user T through an intermediate user N , we need to compute the inferred trust $T_{S \rightarrow T} = T_{S \rightarrow N} \oplus T_{N \rightarrow T}$. Here we employ the symbol \oplus to denote that we need to apply a "special operation" in order to compute the inferred trust in the path.

We assume that the role of each trust score between a pair of users in the equation is proportional to the number of their co-rated items. If I_x is the set of items that user u_x has rated, and $n(I_x)$ is the cardinality of the set I_x , then the "special operator" \oplus is interpreted as:

$$\begin{aligned} T_{S \rightarrow T} &= T_{S \rightarrow N} \oplus T_{N \rightarrow T} \\ &= \oplus \left(\frac{n(I_S \cap I_N)}{n(I_S \cap I_N) + n(I_N \cap I_T)} |T_{S \rightarrow N}| + \frac{n(I_N \cap I_T)}{n(I_S \cap I_N) + n(I_N \cap I_T)} |T_{N \rightarrow T}| \right) \end{aligned}$$

$$\text{where } \oplus \text{ is } \begin{cases} +, \text{ if } & T_{S \rightarrow N} > 0 \quad \text{and} \quad T_{N \rightarrow T} > 0 \\ -, \text{ if } & T_{S \rightarrow N} > 0 \quad \text{and} \quad T_{N \rightarrow T} < 0 \\ -, \text{ if } & T_{S \rightarrow N} < 0 \quad \text{and} \quad T_{N \rightarrow T} > 0 \\ \text{not applicable, if } & T_{S \rightarrow N} < 0 \quad \text{and} \quad T_{N \rightarrow T} < 0 \end{cases}.$$

The intuition behind this computation is that:

- If user S trusts user N and user N trusts user T then it is inferred that user S trusts user T .
- If user S does not trust user N and user N trust user T then it is inferred that user S does not trust user T .
- If user S trusts user N and user N does not trust user T then it is inferred that user S does not trust user T .
- If user S does not trust user N and user N does not trust user T then inference is not applicable and the length of the supposed path between user S and user T is considered infinite (as previously mentioned by distrust propagation).

If we continue this process, we will be able to compute the trust score between a pair of users through any number of intermediate users. We denote $N = \{N_i: i=1,2, \dots,k\}$ is the set of all intermediate nodes in a trust path that connects user S and user T , then their associated inferred trust is given by the following equation:

$$T_{S \rightarrow T} = (((T_{S \rightarrow N_1} \oplus T_{N_1 \rightarrow N_2}) \oplus \dots) \oplus T_{N_{k-1} \rightarrow T_{N_k}}) \oplus T_{N_k \rightarrow T} \ .$$

As a simple example, consider the following:

- Alice trusts Bob 0.7.
- Bob trusts Carol 0.35.
- Alice and Bob have co-rated 5 items.
- Bob and Carol have co-rated 2 items.

Then inferred trust score that Alice may give to Carol is calculated as the following:

$$\begin{aligned} T_{Alice \rightarrow Carol} &= \frac{T_{Alice \rightarrow Bob} \times n(I_{Alice} \cap I_{Bob}) + T_{Bob \rightarrow Carol} \times n(I_{Bob} \cap I_{Carol})}{n(I_{Alice} \cap I_{Bob}) + n(I_{Bob} \cap I_{Carol})} \\ &= \frac{0.7 \times 5 + 0.35 \times 2}{5 + 2} \\ &= 0.6 \ . \end{aligned}$$

And another example:

- Alice trusts Bob 0.7.
- Bob trusts Carol -0.35.
- Alice and Bob have co-rated 5 items.
- Bob and Carol have co-rated 2 items.

Then inferred trust score that Alice may give to Carol is calculated as the following:

$$\begin{aligned}
T_{Alice \rightarrow Carol} &= - \frac{T_{Alice \rightarrow Bob} \times n(I_{Alice} \cap I_{Bob}) + T_{Bob \rightarrow Carol} \times n(I_{Bob} \cap I_{Carol})}{n(I_{Alice} \cap I_{Bob}) + n(I_{Bob} \cap I_{Carol})} \\
&= - \frac{0.7 \times 5 + 0.35 \times 2}{5 + 2} \\
&= -0.6 .
\end{aligned}$$

We compute the average of all the inferred trust scores that are associated with each of the trust paths to obtain the actual predicted trust score.

3. Finally, we compute the predictions by using the Resnick's formula, using trust scores as weighted coefficients.

3.3 EigenTrust

EigenTrust is a global trust algorithm. It is the most cited trust algorithm for P2P networks. In EigenTrust algorithm, the global reputation of each peer i is given by the local trust values assigned to peer i by other peers, weighted by the global reputations of the assigning peers.

3.3.1 Local trust scores

As previously mentioned, the local trust score $trust_{ij}$ between user i and user j can be computed from the similarity between them. However, we need to normalize them somehow to prevent malicious peers from subverting the systems by making their local trust values high. We defined a normalized local trust value c_{ij} as following:

$$c_{ij} = \frac{\max(trust_{ij}, 0)}{\sum_j \max(trust_{ij}, 0)} .$$

3.3.2 Global trust scores

After obtaining the normalized local trust values, we wish to aggregate them to get the global trust values. Generally, peer i can ask its acquaintances for their opinions about other peers that i doesn't know, then weight their opinions by the trust that i places in them:

$$t_{ik} = \sum_j c_{ij} c_{jk} \ .$$

where t_{ik} is the trust that peer i places in peer k based on asking i 's friends. Rewrite the formula in matrix notation we get:

$$t_i = C' c_i \ .$$

where C is a matrix with entries c_{ij} and t_i is a vector containing the values t_{ik} .

In this way, each peer will gain a wider view of the network, other than only his own experience about other other peers. Peer i also wish to ask his friends' friends ($t = (C')^2 c_i$). If he continues this manner, ($t = (C')^n c_i$), he will have a complete view of the network after $n =$ large number of iterations.

If n is large, the trust vector t_i will converge to the same vector for every peer i .

3.3.3 The algorithm

Although EigenTrust algorithm is target at P2P systems, the basic EigenTrust algorithm ignores the distributed nature of P2P network and thus can apply to regular systems.

$t^{(0)} = e;$ repeat $t^{(k+1)} = C' t^{(k)};$ $\delta = \ t^{(k+1)} - t^{(k)} \ ;$ until $\delta < \varepsilon;$

Table 3.1: Simple non-distributed EigenTrust algorithm

In the above algorithm, e is the m -dimensional initialization vector (m is the number of users). Coordinates of e represent a uniform probability distribution over all m peers, $e_i = \frac{1}{m}$.

Chapter 4

Methodology

4.1 The dataset

In order to compare the results of different recommendation techniques, we use a dataset from the MovieLens movie recommendation site [1]. The dataset consists of 100.000 ratings on a scale of 1-5 from 943 users on 1682 movies. Rating-record is defined to be a triplet $\langle user, movie, rating \rangle$. Each user has at least 20 ratings. The user-item matrix produced from the database is sparse, with values in 6% of its entries.

Table 4.1 lists the database statistics.

Overall average of ratings	Average number of ratings to each movie	Average number of ratings by each user
3.53	60	106

Table 4.1: Database statistics

Figure 4.1 shows the histogram of all ratings.

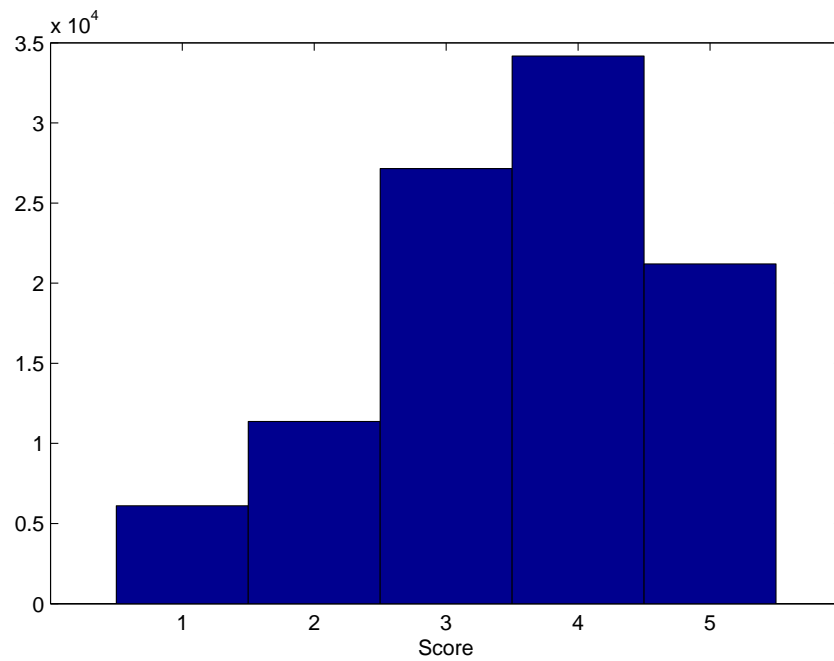


Figure 4.1: Histogram of all movie scores

The above histogram and average score show that users tend to rate movies they liked (high score) more than movies they disliked (low score).

Figure 4.2 shows the histogram of the number of ratings given to each movie.

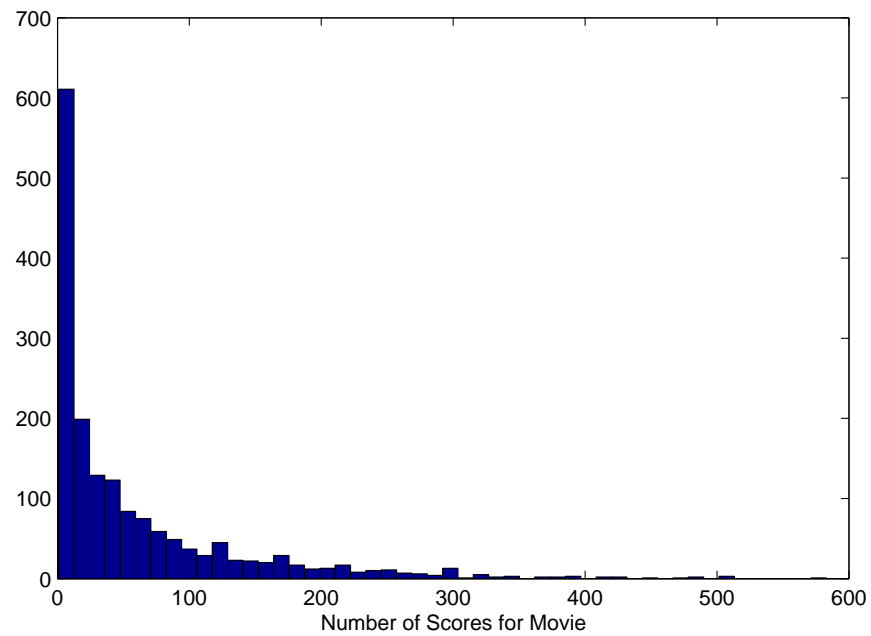


Figure 4.2: Histogram of number of scores given to each movie

The histogram shows that while some movies have more than 500 ratings, many others have 10 ratings or less.

Figure 4.3 shows the histogram of the number of ratings given by each user.

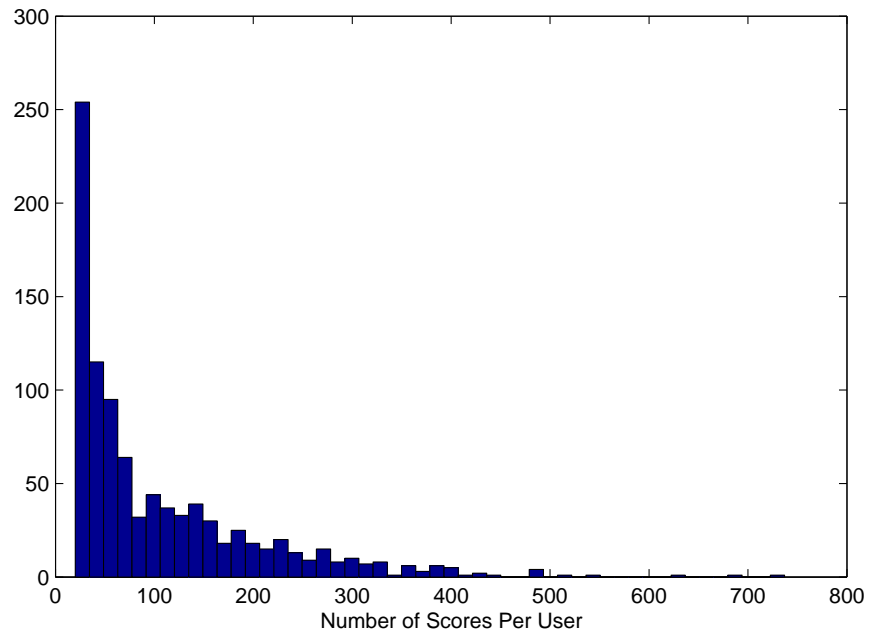


Figure 4.3: Histogram of number of scores given by each user

This histogram shows that, while some users rated as many as 700 movies, many others rated the minimum allowed (20).

Figure 4.4 shows a graphical representation of the database.

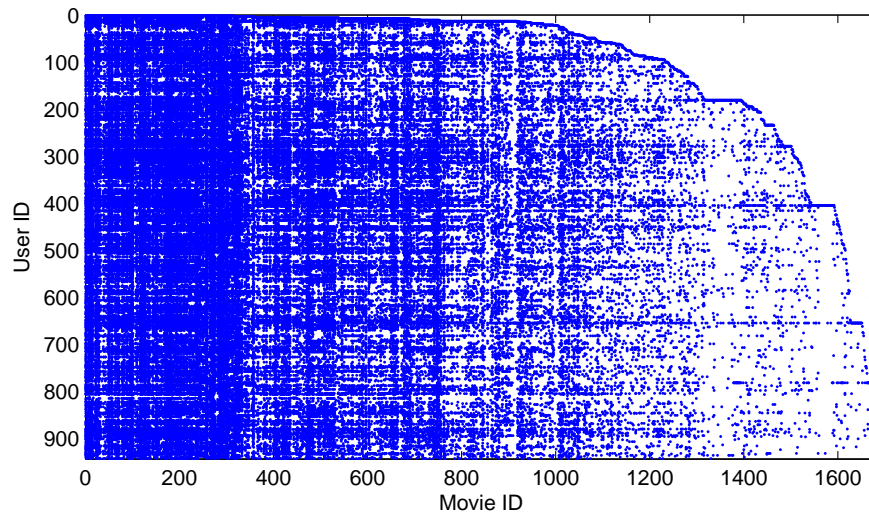


Figure 4.4: Graphical representation of rated movies

This graphical representation implies that when the database was established, each new user was provided with a more elaborate list of movies, so a new user (high user ID) tends to rate a wider range of movies. It is also obvious that movies with higher IDs (recently added movies) were given fewer ratings than those with lower IDs (old movies).

We then divide the dataset into pairs of training sets and test sets according to different ratios. We call this training ratio and denote it by x . A value of $x = 0.2$ means that we divide the 100.000 ratings dataset into 20.000 train cases and 80.000 test cases. The training set is used as input for the algorithm. After that, the predicted results will be compared against the test set to evaluate the algorithm's performance.

4.2 Evaluation Metrics

In recommender systems literature, several types of evaluation criteria have been proposed for evaluating the success of a recommender system. Here we only consider the quality of prediction, because we are only interested in the output of a recommender system for the evaluation purpose.

- *Coverage.* Coverage metrics evaluate the percentage of items that the system could provide predictions. The importance of coverage depends on the requirement of particular applications. Some applications may require predictions for most items in the database. Others may choose to compromise coverage for improved accuracy, for instance, systems that choose to have small neighborhood sizes of the most similar users.

Almost all the experimental results demonstrated in this thesis had maximal coverage. Maximal coverage would probably be a little less than 100% because there may be no ratings in the data for certain items, or because very few people have rated an item, but those people have no correlation with the target user.

We compute coverage as percentage of items in the test data set that the system was able to produce a prediction.

- *Statistical accuracy.* Many metrics have been introduced to assess the accuracy of a collaborative filtering system. They are divided into two main categories: statistical accuracy metrics and decision-support accuracy metrics.
 - Statistical accuracy metrics evaluate the accuracy of a filtering system by comparing the numerical predicted rating values against user actual rating values, over all the items that have both predictions and ratings. Some of the most frequently used metrics are *Mean Absolute Error (MAE)*[25, 27], *Root Mean Squared Error (RMSE)*[25], and *Correlation* between ratings and predictions [25, 14].

- Decision-support accuracy metrics evaluate how effectively a prediction engine could help a user select high-quality items from the set of all items. They are based on the assumption that for many users, filtering is a binary process - either they will or will not take the item. With this observation, the difference between a prediction of 1.5 and 2.5 is irrelevant if the user only chooses to consider the items recommended with a prediction of 4 or higher. The most commonly used decision-support accuracy metrics are *reversal rate*, *weighted errors* and *ROC sensitivity* [24].

Related papers show that MAE is consistent with other metrics and proves to be the most widely used. In addition, it is easy to interpret directly, hence, we used MAE as our choice of evaluation metric to report prediction experiments.

If n is the number of actual ratings in an item set, then MAE is defined as the average absolute difference between the n pairs $\langle p_i, r_i \rangle$ of predicted ratings p_i and the actual ratings r_i :

$$MAE = \frac{\sum_{i=1}^n |p_i - r_i|}{n} .$$

The lower the MAE, the more accurate the overall prediction is, allowing better recommendations to be formulated.

Chapter 5

Results and Evaluation

In this chapter, we present the experiments we have conducted for evaluating the performance of three different algorithms for recommender systems. The dataset used is MovieLens as described in Section 4.1. The dataset is then divided into different pairs of training sets and test sets by different training ratios. First, we list the experimental steps we follow, then we present the actual experiments results and finally, the results are discussed. Each algorithm's result is evaluated with regard to MAE (see Section 4.2) and compared to results from a pure collaborative filtering algorithm. The results are presented graphically in order to give a visual grasp.

As a very first step, we tried three very simple, trivial, non-personalized algorithms. The first algorithm simply returns 5 as the predicted rating a user would give to an item. This trivial algorithm is not meaningful from a point of view of recommender systems, because it does not allow to differentiate and prioritize the different items. However, it allows us to start exploring which MAE a simple algorithm would achieve. As can be seen from the result table 5.1, this algorithm works badly (the MAEs over all the ratings are around 1.45). The reason for the bad performance is that in our dataset, most of the rating values are not 5, and the median rating value is 3.53 (see Section 4.1).

The second trivial algorithm predicts the rating value as the mean of all the ratings provided by one user. The third trivial algorithm predicts the rating value as the mean of all the ratings given to one movie. We observed that these two algorithms achieve not so bad performance, especially when we compare them with more complex algorithms as we will do in the following. In addition, the latter seems to perform better than the former because it

produces smaller MAEs. That is also the reason we choose the movie average approach in our attempt to fill in the sparse user-item matrix used in the LSI/SVD algorithm (see Section 3.1.3).

		Prediction Algorithms		
		Always 5	User Average	Movie Average
Training ratio	0.2	1.4685	0.8794	0.8574
	0.3	1.4673	0.8557	0.8430
	0.4	1.4652	0.8496	0.8352
	0.5	1.4625	0.8487	0.8326
	0.6	1.4603	0.8491	0.8305
	0.7	1.4607	0.8488	0.8286
	0.8	1.4641	0.8502	0.8285
	0.9	1.4412	0.8472	0.8231

Table 5.1: Non-personalized prediction algorithms MAE

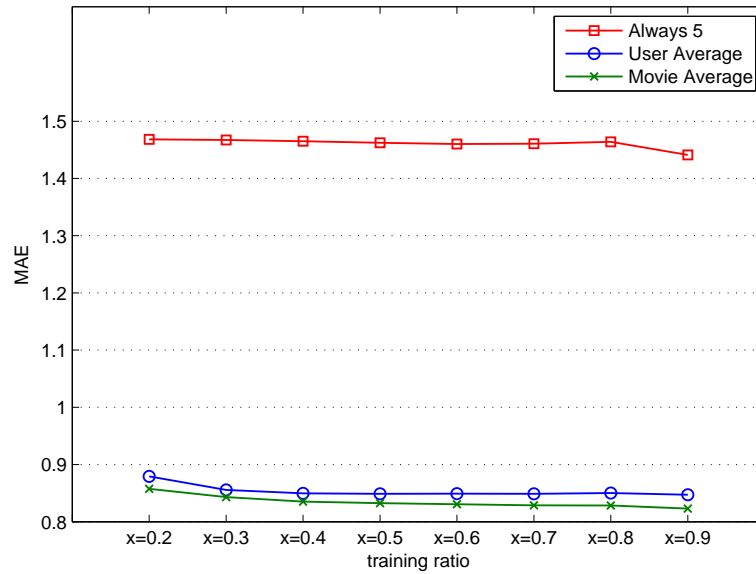


Figure 5.1: Non-personalized algorithms

In all the above and the following experiments, we use different algorithms only for calculating the weighted factors. After that, the predictions are all

computed using Resnick's formula:

$$p_{a,i} = \bar{r}_a + \sum_{u=1}^n w_{a,u} (r_{u,i} - \bar{r}_u) .$$

5.1 LSI/SVD

5.1.1 Experimental steps

We started by loading a training data set into data matrix R representing a rating on 1-5 scale, if the user i doesn't rate the movie j then the entry $r_{i,j}$ is 0. We then performed the following experimental steps.

- We computed the average ratings for each user (row average) and for each movie (column average).
- We filled the matrix by replacing each 0-entry in the matrix with the corresponding column average (average rating for the movie).
- We normalized the matrix by replacing each entry $r_{i,j}$ in the matrix with $(r_{i,j} - \underline{r}_i)$, where \underline{r}_i is the row average of the i^{th} row (average rating for the i^{th} user).
- We used MATLAB to compute the SVD of the filled and normalized matrix, obtaining the three SVD component matrices U , S , and V' . S is the matrix that contains the singular values of matrix R sorted in decreasing order.
- We computed the reduced matrix S_k from S by keeping only k largest singular values.
- We computed the square root of S_k and computed the matrix products $U_k S_k^{1/2}$ and $S_k^{1/2} \cdot V'$. As previously mentioned, the dot product of a row from $U_k S_k^{1/2}$ and a column from $S_k^{1/2} \cdot V'$ will give us a prediction score. Therefore, the entry $p_{i,j}$ of the matrix product $P = U_k S_k^{1/2} \cdot S_k^{1/2} \cdot V'$ is the prediction score that the i^{th} user would give to the j^{th} movie.
- Finally, we denormalized the matrix P by adding the user average back into each prediction score. To compare the quality of SVD-based prediction, we computed the MAE of the prediction scores, and compared it with the MAE of a classic collaborative filtering prediction scores.

This CF prediction algorithm employed the *cosine vector similarity* to produce predictions.

We repeated the entire process for $k = 2, 5-21, 25, 50$ and 100 , and found 14 to be the most optimum value as shown in Figure 5.2. We then fixed this value for k and changed the training ratio x from 0.2 to 0.9 (increment step is 0.1) for testing the sensitivity of the algorithm on different sparsity levels.

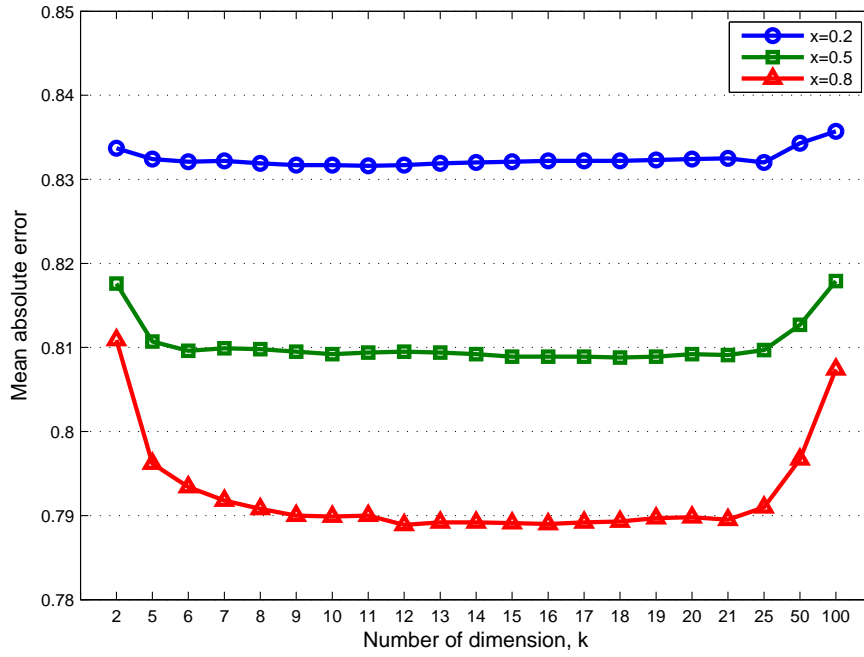


Figure 5.2: Determination of optimum value of k

5.1.2 Results

Figure 5.3 describes the results for our experiment. The data sets were generated from the database of 100.000 ratings, by varying the sizes of the training ratio x . The different values of x were used to determine the sensitivity of the different techniques on the sparsity of the training set (the smaller the ratio, the higher sparsity level of the training data set).

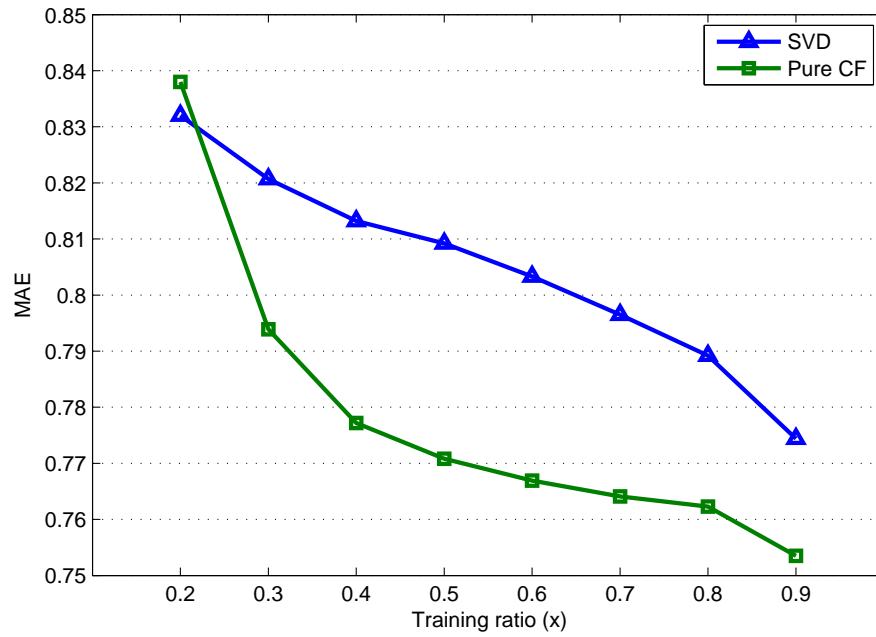


Figure 5.3: SVD vs. CF-Predict prediction quality

5.1.3 Discussions

From Figure 5.3, we observe that for small training ratio x ($x = 0.2$), SVD-based predictions are better than the classic CF predictions. However, for greater values of x , the classic CF predictions are better. This observation suggests that classic CF algorithms are susceptible to data sparsity, because the neighborhood formation process is hindered by the insufficiency of training data. Meanwhile, SVD based prediction algorithms can alleviate the sparsity problem by utilizing the latent relationships. As the training data is increased, both SVD and classic CF prediction quality improve, but the improvement of classic CF prediction surpasses the improvement of SVD-based prediction approach.

Overall, the results are encouraging for the use of LSI/SVD in collaborative filtering systems. The LSI/SVD algorithm provides quite good quality predictions.

5.2 Trust Inferences

5.2.1 Experimental steps

We started by loading a training data set into data matrix R representing a rating on 1-5 scale, if the user i doesn't rate the movie j then the entry $r_{i,j}$ is 0. We then performed the following experimental steps.

- We first built up the trust scores between users by using Pearson correlation method. The results were the direct trust scores between users (1-HOP). They would be used for the classic CF algorithms. In this experiment, we ignored the propagation of distrust, so we just kept the trust scores which are larger than 0.
- We computed the inferred trust scores for pairs of users whose trust scores were not achieved by our first attempt. The inferred trust scores were calculated through one intermediate user (2-HOP).
- Finally, we computed the predictions by Resnick's formula, using computed trust scores as weighted coefficients. We used 0.4 as our threshold for calculating predicted rating scores, so we only considered the neighbors who gained trust scores of 0.4 or greater.

5.2.2 Results

Figures 5.4 and 5.5 describe the results of our experiment. We would like to test the algorithm's performance by both measures, coverage and MAE.

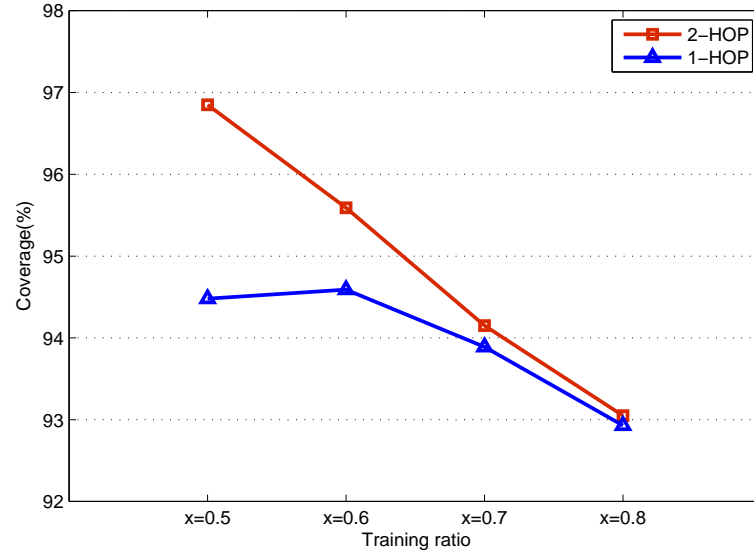


Figure 5.4: Trust inference vs. CF-Predict prediction coverage

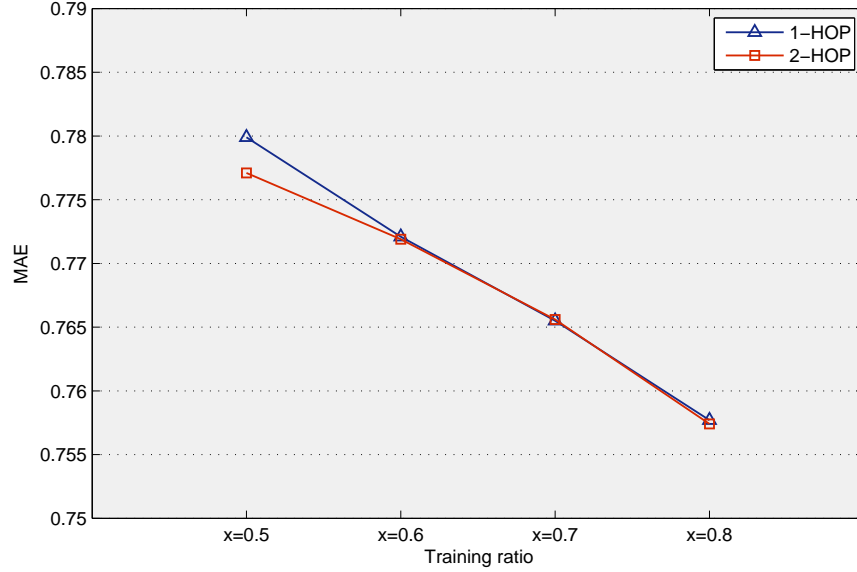


Figure 5.5: Trust inference vs. CF-Predict prediction quality

5.2.3 Discussions

By propagating trust, it is possible to reach more users and hence predict a trust score for more of them and to count them as neighbors, so it helps increase the prediction coverage. This can be seen from Figure 5.4, the prediction coverage of 2-HOP is greater than that of 1-HOP in all tests run.

As far as statistical accuracy is concerned, we observe that the 2-HOP algorithm slightly outperforms the 1-HOP classic collaborative filtering for all the sparsity levels. From Figure 5.5, we also notice that the quality of predictions depends on the sparsity level. In cases that data is more sparse, we can see more clearly the prediction quality of 2-HOP is better than that of the classic 1-HOP algorithm.

So, we would conclude that the trust inference algorithm helps deal with the sparsity problem. It is able to provide high-quality predictions even when information is insufficient.

5.3 EigenTrust

5.3.1 Experimental steps

An additional experiment we performed is to test the performance of global trust metrics for trust-based collaborative filtering recommender systems. We performed the following experimental steps:

- We first built up the local trust scores $trust_{ij}$ between users by using Pearson correlation method. We considered only the positive trust scores.
- We normalized the local trust scores to as the following:

$$c_{ij} = \frac{\max(trust_{ij}, 0)}{\sum_j \max(trust_{ij}, 0)} .$$

- We initialized the trust vector by assigning the value $\frac{1}{n}$ to every peer i , where n is the number of peers.
- We then applied the basic EigenTrust algorithm to obtain the global trust vector whose coordinates are global trust scores of each user in the system.
- Finally, we computed the predictions using the Resnick's formula, with global trust scores as weighted coefficients.

5.3.2 Results

We tested the algorithms over 5 different pairs of training sets and test sets with training ratio of 0.8. The MAE was computed for each experiment. The experimental results are illustrated in Figure 5.6.

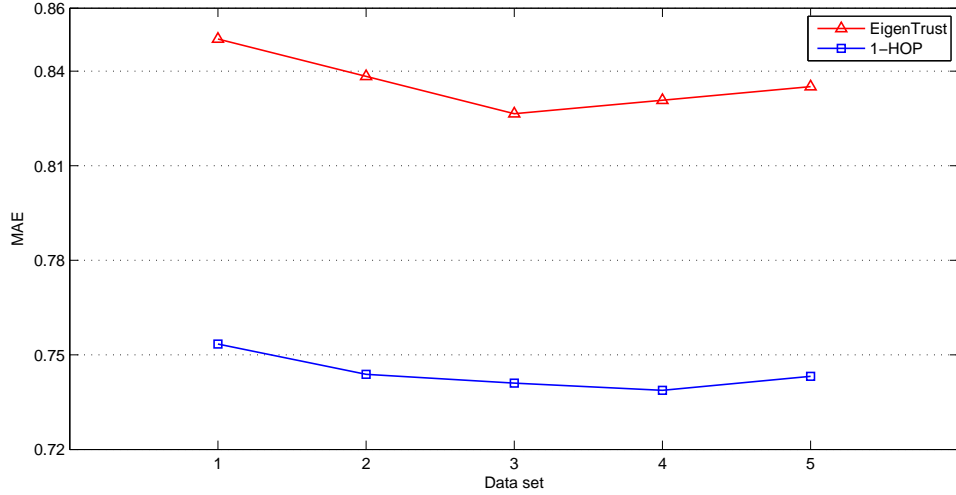


Figure 5.6: EigenTrust vs. CF-Predict prediction quality

5.3.3 Discussions

As can be seen from Figure 5.6, the performance of EigenTrust algorithm is worse than that of classic collaborative filtering algorithm. This might indicate that a global trust metric is not suited for a trust-based recommender system, although it has proven to successfully alienate malicious peers in P2P systems [3].

EigenTrust algorithm gives the same trust scores for each user. In the experiment, we use the global trust score as the weighted coefficient in the prediction formula. Hence, the result is not personalized while in a recommender system, the purpose is to leverage individual different opinions and not to merge all of them into a global average. The reason behind the bad performance is that globally trusted users tend to be peculiar in their rating patterns and provide more varied ratings so that averaging them generates larger errors.

5.4 Discussions

As far as statistical accuracy is concerned, by conducting the above experiments, we reach the following conclusions about the performance of the prediction algorithms:

- Singular Value Decomposition (SVD) has proven to be effective in recommender systems. This technique can drastically reduce the dimension of the ratings matrix, leading to very fast online performance because it requires just a few simple operations for each recommendation. Therefore, it helps to deal with scalability problem in collaborative filtering approach, yet it is capable of providing high quality predictions. Besides, SVD technique could be used in other ways, such as using SVD for neighborhood selection, or using SVD to create low-dimensional visualizations of the ratings space.
- Trust-aware algorithms are effective in recommender systems. Especially, when the level of sparsity is high, local trust propagation techniques have proven to perform well. However, global trust metrics are not appropriate for trust-aware recommender systems due to their non-personalized nature.
- Finally, simple algorithms seem very effective, at least in our chosen MovieLens dataset. This might be good news for small e-commerce sites who would like to use a recommender system as an add-on without having to spend so much effort on it. If this is a case, they would not need to employ expensive techniques while still could achieve reasonable quality recommendations based on simple, non-personalized techniques.

In all our experiments, we use MovieLens dataset in which each user has given rating to at least 20 items, therefore, the dataset is quite dense. It is still an open point to understand how much the performance of different algorithms is affected by the different datasets.

Another concern is that, the MovieLens dataset does not include trust ratings. In experiments of trust-based algorithms, we computed trust information by Pearson coefficient technique, based on the assumption that user similarities have a strong connection with trust values they would place on each other. In the future, we would like to test the trust-based algorithms over datasets which provide real trust rating information.

Chapter 6

Security and privacy issues in recommender systems

Recommender systems have proven to be effective in helping users deal with the problem of information overload. However, they also raise serious privacy and security concerns. In order to provide high quality recommendations, RSs need to collect the personal information and this raises the risks of *exposure* of that information. In addition, malicious users can manipulate their ratings to inappropriately change the items that are recommended (*bias*). Or they can *sabotage* the recommender systems by using denial of service attacks or defacement of the front page.

Figure 6.1 shows high-level data flow between a user and a recommender system: the user gives personal information in return for predicted preferences. The recommender system consists of three components: the model builder, the data store and the predictor engine.

- The *model builder* selects, combines, and computes a user model from personal information.
- The *data store* holds the results of model building as well as any other information needed for the application.
- The *predictor* uses the model to predicts preferences.

The model also encounters the same security issues as the classic client-server model: man-in-the-middle attacks, denial of service attacks, hacking into the

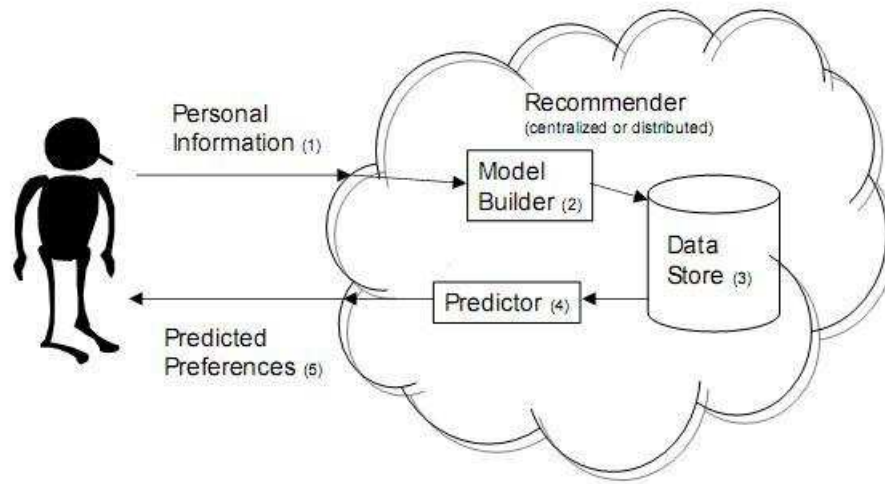


Figure 6.1: Conceptual model of the interaction between a user and a recommender system

server... Therefore, we will not consider those in this section. We will focus only on issues specific to recommender systems.

6.1 Privacy in recommender systems

6.1.1 Privacy risks of information

A recommendation algorithm requires input from the users before it could produce recommendations. The more input it collects, the more accurate recommendations it can provide, but also the risk of unwanted personal information exposure increases.

Recommender systems aggregate user preferences in ways similar to statistical database queries, which can be exploited to identify information about a particular user. There is the direct risk that someone will learn information that the user wished to keep private. There are also indirect risks of *re-identification*, that is, finding information about a user in one system that could identify her in another system. This is especially true for users with eclectic tastes who rate products across different types or domains in the systems. These straddlers highlight the conflict between personalization and privacy in recommender systems: while they enables serendipitous recom-

mendations (see Section 1.1), straddlers can be used to uncover identities and reveal personal details.

Revealing identifying information could lead to identity theft attacks. However, the sensitivity of information may vary from domain to domain and from time to time. In some domains, users may be open to sharing their tastes with others, while in other domains, such as medical information, users may have serious concerns about sharing their preferences with anyone.

6.1.2 Privacy preserving for recommender systems

We note that the dimensional reduction technique based on SVD also has properties to protect users' privacy. That is, it offers high compression of the original data, and therefore good protection of user data. In other schemes, such as Pearson correlation or trust-based collaborative filtering techniques, the whole user dataset is used to generate recommendations, therefore, they are exposed more to privacy risks.

As previously mentioned, collaborative filtering recommender systems are very much similar to statistical database queries. Hence, privacy preserving techniques for data mining could be applicable to them. Some of the techniques are listed as follows [29]:

- Heuristic-based techniques like adaptive modification that modifies only selected values that minimize the utility loss rather than all available values.
- Cryptography-based techniques like secure multi-party computation where a computation is secure if at the end of the computation, no party knows anything except its own input and the results.
- Reconstruction-based techniques where the original distribution of the data is re-constructed from the randomized data.

Users still can have a certain level of anonymity. They don't have to reveal everything in order for the system to be useful, and they have to trust the company who owns the e-commerce site to use their information appropriately. For example, recommender systems can work even if they don't tell a user who has similar tastes to her. And they ask her recommendations

to suggest things to other people, but they don't have to tell her who will benefit from her suggestions. They don't have to tell the people receiving the suggestions who they get the suggestions from. In this way, the recommender systems can actually work anonymously.

Anonymous techniques allow users to reveal their data without disclosing their identity information. However, there are cases when the recommender system owner would need to identify malicious users or competing companies, who intentionally subvert the system by their ratings. Therefore, it is important for the recommender system owner to be able to verify the identities of the users in order to guarantee the quality of the recommender system.

The privacy issue in recommender systems is interesting. However, it should not be the reason that stops users from using them. To the best of our knowledge, no recommender systems that take into account privacy concerns have been yet designed. Recommender systems are just add-ons to e-commerce sites. They are not the thing that drives online shopping. If users want the systems to be able to give personalized services to them, they have to trust the people who run the system.

6.2 Security of recommender systems

Recommender systems are often used in e-commerce to provide high quality recommendations and therefore can drive sales. That is also the reason these systems are prone to manipulation from malicious producers or users. Here we focus on a *shilling attack*, [5], which attempts to manipulate the recommendations for a particular item by submitting misrepresented opinions to the system.

6.2.1 Attack motivations

Recent research has shown that most popular algorithms employed in current CF applications can be easily manipulated through biased profiles [16]. Attackers may create fake user profiles that highly rate a set of target items, and then rate other items, in such a way that they become similar to many other user profiles in the systems. This type of attacks is known as *shilling*

attacks.

Different shilling attacks may have different motivations. Two straightforward motivations are to increase the ratings of certain target items in order to have them recommended to more users (push attack), and to lower the ratings of target items in order to have them recommended to fewer users (nuke attack).

Another possible motivation is to simply damage the recommender system as a whole; that is, to reduce the quality of the recommendation, resulting in decreasing overall user satisfaction with the system, and eventually, users will stop using it. The successful attack might benefit competing recommender systems.

6.2.2 Required knowledge for attacks

In order to be able to maintain the attack for as long as possible before being spotted and stopped, the attack may require attacks certain knowledge about items, users, ratings, and algorithms in the target recommender system. Generally, an informed attack will be more effective than an uninformed attack. Knowledge about the system such as ratings sparsity, ratings distribution, and algorithm parameters can help in choosing which attack to employ and in varying attack parameters to maximize effectiveness and minimize detectability.

Furthermore, it might be beneficial that attackers wisely choose to target at a target subset of users and a subset of items in a recommender system. For example, it might raise suspicion if a heavy metal album is recommended to a group of users who are interested in classical music and would have absolutely no interest in such an album.

6.2.3 Defending against attacks

Shilling attacks may be addressed by developing attack-resistant algorithms [15], or increasing the cost of obtaining identities [8]. Trust-based algorithms have proven to be successful in preventing shilling attacks. O'Donovan et al. [20] claim that an algorithm based on implicit trust scores that are computed from the accuracy of past recommendations can make shilling attacks

less effective.

The other approach, making acquiring identities more expensive, would prevent an attack from even reaching the recommender algorithm. Some useful techniques are using CAPTCHA or requiring non-trivial commitment of resources such as fee or computational ability.

There are more traditional cryptographic solutions of identity management, such as, using a trusted third party to ensure each person can establish only one identity. This approach can raise the cost of an attack, but also raises privacy concerns because it requires users reveal their identity to use the system. Furthermore, if the trusted third party is used in multiple systems, then it becomes possible to track one person across them.

Chapter 7

Conclusions

Recommender Systems (RS) have emerged as an important response to the so-called information overload problem. They enable users to share their opinions and benefit from each other's. Recommender algorithms are best known for their use on e-commerce Web sites to help users find products they would appreciate from huge catalogues. They will become increasingly important in the future as a key to automate mass customization for e-commerce sites.

To date, there are basically two types of recommending techniques used on the Internet: content-based - also dubbed item-item - recommenders and collaborative filtering recommender systems. The interesting point of the latter is that, knowledge about items is not required, it is based only on the ratings of the user community. Therefore, they promise to scale well to large databases. The traditional collaborative filtering techniques are able to provide high-quality recommendations by leveraging the preferences of similar users. However, recent researches have suggested that the traditional focus on user similarity may not be sufficient. Additional factors, especially trust may have an important role when it comes to making recommendations.

In this thesis, we study the different algorithms and the usage of trust to improve the performance of collaborative filtering recommender systems. Our experiments are done on MovieLens dataset and we choose MAE as our measure of prediction quality. The results show that the dimensionality reduction method that uses LSI/SVD technique helps in providing better quality of recommendations.

Trust is a concept that receives increasing attention by the research community and currently be used in many online systems. In this thesis, we show that trust also has positive impact on overall prediction error rates, and local trust metrics for propagating trust allows us to achieve better prediction coverage. However, global trust metrics may not be appropriate for trust-aware recommender systems due to their non-personalized nature.

In conclusion, recommender systems are creating value for both e-commerce sites and their customers. There is not only one recommender system for the whole world, each site chooses its own recommender system that fits their needs and their condition the most. One big challenge faced by recommender systems is how to collect sufficient data to make high-quality recommendations, especially for new users to keep them stay. There is a possibility that a group of e-commerce sites should share information about their users. However, such a group needs to assure that the privacy of users will be carefully taken care of.

Usaldusmehhanismide valik soovitussüsteemides

Magistritöö

Nguyen Hoang Anh

Kokkuvõte

Soovitussüsteemid (*Recommender Systems*) on välja pakutud võimaliku lahendusena informatsiooni ülekülluse probleemile. Nad aitavad kasutajatel jagada üksteisega oma arvamusi. Tuntuim soovitusalgoritmide rakendamise koht on e-äri veebisaidid, kus nad aitavad kasutajatel teha valikuid oma hiiglaslikest tootekataloogidest. Need tooted võivad olla raamatud (näiteks Amazon.com), filmid (näiteks Netflix), fotod (näiteks Flickr.com) või ka veebilehed (näiteks del.icio.us).

Käesoleval ajal on internetis levinud põhiliselt kahte tüüpi soovitussüsteemid — sisul põhinevad soovitajad ja ühisfiltreerimisel põhinevad soovitajad. Viimaste huvitavaks omaduseks on, et teadmused soovitavate esemete kohta pole tarvis, toetutakse ainult kasutajate kogukonna antud hinnetele. Seega võib loota, et nad skaleeruvad hästi. Traditsioonilised ühisfiltreerimissüsteemid annavad kõrgekvaliteedilisi soovitusi, toetudes sealjuures sarnaste kasutajate eelistustele. Siiski võib hiljutistest uurimistulemustest järeldada, et ainult kasutajate sarnasusele toetumine ei pruugi olla piisav. Peale selle võib olla tarvis arvestada ka muude teguritega, näiteks usaldusega.

Käesolevas magistritöös uurime me erinevaid algoritme ja usalduse kasutamist ühisfiltreerimisel põhinevate soovitussüsteemide parendamiseks. Meie katsed *MovieLens* andmekorpusel näitavad, et LSI/SVD-tehnikat kasutav dimensioonide vähendamise meetod aitab paremaid soovitusi anda. Usaldus mõjutab positiivselt ka üldisi ennustamisvea hinnanguid. Samas ei pruugi siiski globaalsed usaldusmeetrikad oma mitteisikustatuse tõttu olla sobilikud usaldustundlike soovitussüsteemide jaoks.

Võtmesõnad: Soovitussüsteemid, ühisfiltreerimine, usaldus, LSI, SVD, Eigen-Trust, usalduse tuletused.

Bibliography

- [1] Movielens dataset. <http://www.grouplens.org/taxonomy/term/14>.
- [2] ABDUL-RAHMAN, A., AND HAILES, S. Supporting trust in virtual communities. In *HICSS '00: Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 6* (Washington, DC, USA, 2000), IEEE Computer Society, p. 6007.
- [3] ABRAMS, Z., MCGREW, R., AND PLOTKIN, S. A non-manipulable trust system based on eigentrust. *SIGecom Exch.* 5, 4 (2005), 21–30.
- [4] ARTZ, D., AND GIL, Y. A survey of trust in computer science and the semantic web. *Web Semant.* 5, 2 (2007), 58–71.
- [5] CHIRITA, P.-A., NEJDL, W., AND ZAMFIR, C. Preventing shilling attacks in online recommender systems. In *WIDM '05: Proceedings of the 7th annual ACM international workshop on Web information and data management* (New York, NY, USA, 2005), ACM, pp. 67–74.
- [6] DEERWESTER, S., DUMAIS, S. T., FURNAS, G. W., LANDAUER, T. K., AND HARSHMAN, R. Indexing by latent semantic analysis. *Journal of the American Society for Information Science* 41 (1990), 391–407.
- [7] DEUTSCH, M. Cooperation and trust: Some theoretical notes. In *Nebraska symposium on motivation* (1962), Lincoln, NE: University of Nebraska Press, pp. 275–319.
- [8] FRIEDMAN, E. J., AND RESNICK, P. The social cost of cheap pseudonyms. *Journal of Economics and Management Strategy* 10 (1998), 173–199.
- [9] GOLBECK, J. Generating predictive movie recommendations from trust in social networks. In *iTrust* (2006), pp. 93–104.

- [10] GOLBECK, J. Trust and nuanced profile similarity in online social networks. *Journal of Artificial Intelligence Research* (2006).
- [11] GOLUB, G., AND KAHAN, W. Calculating the singular values and pseudo-inverse of a matrix. *Journal of the Society for Industrial and Applied Mathematics, Series B: Numerical Analysis* 2, 2 (1965), 205–224.
- [12] GUHA, R., KUMAR, R., RAGHAVAN, P., AND TOMKINS, A. Propagation of trust and distrust. In *WWW '04: Proceedings of the 13th international conference on World Wide Web* (New York, NY, USA, 2004), ACM, pp. 403–412.
- [13] HERLOCKER, J. L., KONSTAN, J. A., BORCHERS, A., AND RIEDL, J. An algorithmic framework for performing collaborative filtering. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 1999), ACM, pp. 230–237.
- [14] HILL, W., STEAD, L., ROSENSTEIN, M., AND FURNAS, G. Recommending and evaluating choices in a virtual community of use. In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 1995), ACM Press/Addison-Wesley Publishing Co., pp. 194–201.
- [15] KAMVAR, S. D., SCHLOSSER, M. T., AND GARCIA-MOLINA, H. The eigentrust algorithm for reputation management in p2p networks. In *In Proceedings of the Twelfth International World Wide Web Conference* (2003), ACM Press, pp. 640–651.
- [16] LAM, S. K., AND RIEDL, J. Shilling recommender systems for fun and profit. In *WWW '04: Proceedings of the 13th international conference on World Wide Web* (New York, NY, USA, 2004), ACM, pp. 393–402.
- [17] MARSH, S. P. *Formalising Trust as a Computational Concept*. PhD thesis, University of Stirling, Apr 1994.
- [18] MASSA, P., AND AVESANI, P. Trust-aware recommender systems. In *RecSys '07: Proceedings of the 2007 ACM conference on Recommender systems* (New York, NY, USA, 2007), ACM, pp. 17–24.
- [19] O'DONOVAN, J., AND SMYTH, B. Trust in recommender systems. In *IUI '05: Proceedings of the 10th international conference on Intelligent user interfaces* (2005), ACM Press, pp. 167–174.

- [20] O'DONOVAN, J., AND SMYTH, B. Is trust robust?: an analysis of trust-based recommendation. In *IUI '06: Proceedings of the 11th international conference on Intelligent user interfaces* (New York, NY, USA, 2006), ACM, pp. 101–108.
- [21] O'MAHONY, M., HURLEY, N., KUSHMERICK, N., AND SILVESTRE, G. Collaborative recommendation: A robustness analysis. *ACM Trans. Internet Technol.* 4, 4 (2004), 344–377.
- [22] PAGE, L., BRIN, S., MOTWANI, R., AND WINOGRAD, T. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.
- [23] RESNICK, P., AND VARIAN, H. R. Recommender systems. *Commun. ACM* 40, 3 (1997), 56–58.
- [24] SARWAR, B. M., KARYPIS, G., KONSTAN, J. A., AND RIEDL, J. T. Application of dimensionality reduction in recommender system - a case study. In *ACM WebKDD Workshop* (2000).
- [25] SARWAR, B. M., KONSTAN, J. A., BORCHERS, A., HERLOCKER, J., MILLER, B., AND RIEDL, J. Using filtering agents to improve prediction quality in the grouplens research collaborative filtering system. In *CSCW '98: Proceedings of the 1998 ACM conference on Computer supported cooperative work* (New York, NY, USA, 1998), ACM, pp. 345–354.
- [26] SCHAFER, J. B., KONSTAN, J., AND RIEDL, J. Recommender systems in e-commerce. In *EC '99: Proceedings of the 1st ACM conference on Electronic commerce* (New York, NY, USA, 1999), ACM, pp. 158–166.
- [27] SHARDANAND, U., AND MAES, P. Social information filtering: Algorithms for automating "word of mouth", 1995.
- [28] SINHA, R., AND SWEARINGEN, K. Comparing recommendations made by online systems and friends. In *In Proceedings of the DELOS-NSF Workshop on Personalization and Recommender Systems in Digital Libraries* (2001).
- [29] VERYKIOS, V. S., BERTINO, E., FOVINO, I. N., PROVENZA, L. P., SAYGIN, Y., AND THEODORIDIS, Y. State-of-the-art in privacy preserving data mining. *SIGMOD Record* 33 (2004), 2004.

- [30] YAHALOM, R., KLEIN, B., BETH, T., AND BETH, D. T. Trust relationships in secure systems - a distributed authentication perspective. In *In Proceedings, IEEE Symposium on Research in Security and Privacy* (1993), pp. 150–164.
- [31] ZIEGLER, C.-N., AND LAUSEN, G. Analyzing correlation between trust and user similarity in online communities. In *Proceedings of Second International Conference on Trust Management* (2004), Springer-Verlag, pp. 251–265.
- [32] ZIEGLER, C.-N., AND LAUSEN, G. Spreading activation models for trust propagation. In *In Proceedings of the IEEE International Conference on e-Technology, e-Commerce, and e-Service* (2004), IEEE Computer Society Press.