UNIVERSITÉ CATHOLIQUE DE LOUVAIN
ÉCOLE POLYTECHNIQUE DE LOUVAIN

Structural trust inference for social recommendation

Promoteur :
Peter Van Roy (EPL)

Lecteurs :
Salim Jouili (EURA NOVA)
Sabri Skhiri (EURA NOVA)

Mémoire présenté en vue de
l'obtention du grade de
Master 120 en Sciences informatiques
(option *Networking and Security*)
par Daire O'Doherty

Louvain-la-Neuve, Belgique
Année académique 2011-2012

ACKNOWLEDGEMENTS

This master's thesis would have not been possible without the help and support of many people.

First and foremost, I would like thank Salim Jouili, for his constant advice, guidance, and for pushing me to always strive for better.

Secondly, I would like to thank my advisor Peter Van Roy, for his wise and useful suggestions which allowed to shore up any gaping holes in my explanations.

Finally, I would like to thank my parents and anybody else who supported and encouraged me throughout the course of this thesis, as well as the last two years.

ABSTRACT

The emergence of trust as a key link between users in social networks has been shown to provide an effective means of enhancing the personalization of on-line user content. However, the availability of such trust information remains a challenge to the algorithms that use it, as the majority of social networks do not provide a means of explicit trust feedback. The first part of this master thesis presents an investigation into the inference of trust relations between actor pairs of a social network, based solely on the structural information of the bipartite graph typical of many on-line social networks. Using intuition inspired from real life observations, this work argues that the popularity of an item in a social graph is inversely related to the level of trust between actor pairs who have rated it. From an existing bipartite social graph, this method computes a new social trust graph, linking actors together by means of symmetric weighted trust relations. Through a set of experiments performed on a real social network dataset, this method is trained, validated and compared to a naive structural trust inference approach producing statistically significant results, and showing strong trust prediction accuracy.

Further to this, the use of trust in recommender systems has been shown to improve the accuracy of rating predictions, especially in the case of "controversial" items, where a user's rating significantly differs from the average. Many different techniques have been used to incorporate trust into recommender systems, each showing encouraging results. However, the lack of trust information available in public datasets has limited the empirical analysis of these techniques and trust-based recommendation in general, with most analysis only being performed on a single dataset. The second part of this work provides a more complete empirical analysis of trust-based recommendation and a further test of the trust inference formula developed in part one. By making use of the trust inference formula developed in the first part of this work, we are able to apply trust-based recommendation techniques to three separate

datasets by incorporating the resulting trust metrics computed between users by the trust inference formula into various trust-enhanced recommendation techniques. These techniques are then analysed and compared to the standard collaborative filtering techniques widely in use today. For this analysis and comparison, we measure the overall accuracy of each technique in terms of the Mean Absolute Error (MAE), the Root Mean Square Error (RMSE) as well as measuring the prediction coverage of each technique (i.e percentage of predictions made). We thus provide a comparison and analysis of each technique on all three datasets, showing the applicability and performance of each technique under differing circumstances, such as a sparse dataset, a well connected dataset, a dataset with ratings of uniform distribution, and finally a dataset containing large differences of opinion between users.

CONTENTS

INTRODUCTION

The exponential growth and development of Web 2.0 has brought about a rapid increase in the availability of on-line user content, as well as creating a fundamental shift in the way people use and share knowledge. The popularity and increased usage of blogs and wikis have given rise to new means of on-line collaboration and information sharing, and have created a virtual platform in which users can explicitly express their preferences and opinions. Furthermore, the emergence of social networks has allowed users to connect themselves to any number of people they know, or who share these preferences and perspectives, forming vast on-line communities of similar, like-minded users. The Internet, as such, has itself become a large social network, linking "people, organizations and knowledge" [6]. With such a vast and ever increasing availability of knowledge and content, these developments have pushed researchers to develop techniques to handle this *information overload*, and to provide certain forms of personalization of the information and content, which would be of the most interest to each individual user. One ongoing area of research attempting to fulfill these needs is that of the incorporation of *trust* into on-line systems.

The emergence of trust [41, 24, 58, 37] as a key link between users in social networks is a growing area of research, where trust has been used for the improvement and enhancement of the individual personalization of many on-line activities. In particular, many studies [25, 16, 51, 58] have shown trust to be greatly effective in improving the prediction accuracy and coverage performance of traditional on-line recommendation techniques, as well as providing a more robust solution to reputation systems for on-line peer-to-peer file sharing [28]. Such research is generally based on the sociological idea that users of on-line networks are more inclined to have similar opinions to people that they know and trust (i.e trusted neighbours), and thus are more likely to appreciate recommendations made by such neighbours. However, trust has different interpretations depending on the domain of research in which the term is used,

from the domains of information security and authentication, to content personalization and recommendation. For the purposes of this master thesis, the notion of trust is seen as an indication of similarity or commonality between two users in a social network. More formally, a trust metric from user u to user v in a social network can be seen as the subjective probability that the truster, u will have the same preferences and tastes as the trustee v.

However, although trust has been shown to improve content personalization and the clustering of similar users [56, 60, 59, 20], it is necessary to be able to effectively and efficiently obtain accurate trust information for subsequent use in such systems. Some previous studies [19, 36] have made use of trust assertions provided explicitly by users in some social networks, such as *Epinions.com* as a means for providing such enhancements. Potential drawbacks of this reliance on explicit user feedback include its lack of availability as well as its potential unreliability.

The unreliability of this feedback may be caused by the potential reluctance of users to publicly provide such feedback, as well as potential user indifference to the system (e.g users providing arbitrary trust ratings, or neglecting it completely), presenting inconsistencies in the provided trust metrics, which may subsequently have a negative impact on the success and appropriateness of the systems using them.

The unavailability of such explicit feedback is also due to the fact that many social networks do not provide the means for this kind of explicit feedback. Taking a real world example of the popular social network Youtube.com, an on-line medium for the distribution of videos, users are able to individually contribute and watch videos, as well state personal preferences by either rating videos or subscribing to different groups. In essence, this network represents a bipartite graph with two distinct sets of vertices, namely users and videos. The edges in the graph represent explicit user preferences for videos in the form of ratings or subscriptions to particular groups. However, this site, like many of its type, does not provide any mechanism for explicit user to user connections, such as a trust connection that may be used for the enhancement of the individual user experience.

The first part of this master thesis is thus motivated by the need for a method for the automatic inference of trust information between users in social networks, for subsequent use in the trust-enhanced algorithms proposed to improve personalized recommendations. Based on real life observations, the first part of this work presents an investigation into how trust connections can be automatically inferred between users in a social network. The basis for this investigation is the correlation of user similarity and trust as shown by Ziegler and Lausen in [65, 21, 64].

In order to provide a generic methodology, that may be applicable to numerous social graphs, the proposed trust inference formula uses only the information contained in the topology and structure of the bipartite graph typical of many social networks, namely the directed edges from user to item vertices, and not to use any of the content of the graph, as such content may differ for each social network. Using this information, the proposed formula focuses on the items for which a pair of users both have a directed edge.

Trust-based Recommendation: A Further Empirical Analysis

Previous work on trust-enhanced recommendation systems has yielded a number of different techniques used to incorporate trust information into the recommendation process. Many of these techniques have been shown to perform significantly better than traditional recommendation techniques in terms of overall accuracy and coverage of rating predictions, particularly where a user's rating of an item differs significantly to the average rating for this item.

Although there have been some previous empirical comparisons of these different trust-enhanced techniques [57], due to the unavailability of suitable datasets, which provide explicit trust information between users, such analysis has usually been restricted to just a single dataset.

The second part of this master thesis thus aims to provide a further empirical analysis and comparison of some of these different trust-enhanced recommendation techniques proposed, as well as a comparison to traditional collaborative filtering algorithms. By making use of the trust inference formula proposed in the first part of this work to automatically infer trust relations between actor pairs, these techniques are performed on three separate datasets, thus allowing a further analysis of the performance of these techniques under varying conditions of different datasets.

Publications

During the course of this work, in co-operation with EURA NOVA, two papers were written and submitted to two workshops:

1. The first paper, based on the first part of this thesis: "Towards trust inference from bipartite social networks" [39] was accepted for publication to the "Second ACM SIGMOD Workshop on Databases and Social Networks: DBSocial", and was presented in Scottsdale (Arizona, USA), on the 20th May 2012.

2. The second paper, based on the second part of this thesis: "Trust-Based Recommendation: an Empirical Analysis" [40] was submitted to the

"Sixth ACM SIGKDD Workshop on Social Network Mining and Analysis: SNA-KDD" which will take place in Beijing, China. As of the writing of this work, the reviewer's decision for accepted paper for this workshop are is to be made public within the next week.

Outline

This thesis is split into three main chapters. The first chapter will present an overview of the state-of-the-art from the two main domains from which this master thesis is based. I will begin by explaining the first domain of on-line content recommender systems, including the two main approaches to on-line content recommendation used today, their methods of gathering information to be used in the recommendation process as well as some of the drawbacks to each approach and the difficulties each of them face. Further to this, I will provide an overview of the second domain of trust in on-line systems in which this master thesis is also heavily based. For this, I will briefly discuss some of the different interpretations of trust in on-line domains of research, and then explain some of the approaches developed to incorporate trust into on-line recommendation and reputation systems.

The second chapter presents the investigation into the automatic inference of trust connections between users in on-line social networks. This chapter will begin with an explanation of the motivation for this part of the thesis, followed by an explanation of the methodology and intuition used, before outlining and explaining the formula developed and used to infer this trust information. Following this, the results of the experiments carried out to validate and train this formula are presented. This chapter represents the main focus and contribution of this master thesis.

The third chapter presents the empirical analysis of trust-based recommendation algorithms performed on three datasets using the trust inference formula developed in the previous chapter. This chapter begins again with an explanation of the motivation for this analysis, before presenting the recommendation techniques performed for the analysis. The datasets, as well as the performance measures used to analyse and compare the accuracy and coverage of each of the algorithms are then presented followed by an explanation of the experiments carried out for the purpose of the analysis. Finally, this chapter presents an analysis section, showing and analysing the results of the experiments performed.

Finally, the conclusion will present a short overview of the main points of this work, as well as a short discussion on the possible directions and future steps which can be taken as part of any future work.

CHAPTER 1

STATE-OF-THE-ART

This chapter provides some background into the state-of-the-art of the two main research domains upon which the thesis is based. Among a number of other domains, such as graph similarity and nearest neighbour discovery, this work can be seen to be based primarily in two research domains: firstly the field of recommender systems and content personalization, and secondly the domain dealing with the concept of "trust" in social networks and more particularly in social recommender systems.

I begin by introducing the domain of recommender systems, giving a formal definition of the recommendation problem, as well as the two primary methods of extracting information to be used in the recommendation process. I will then provide an outline of two of the main categories of recommender systems that are most widely used today, that of Content-based recommendation and Collaborative Filtering recommendation, providing explanations of the techniques and the information of the social graph they use, as well as the advantages and disadvantages of both. The material from this recommendation systems relies heavily on the article: [1].

I then introduce the domain of trust in on-line systems, briefly outlining the differing interpretations of trust in certain domains of research. before providing the interpretation of trust used in trust-based recommender systems, which will be the main focus of this work. Finally, some of the main studies proposed in the area of trust-based recommendation are then presented, grouped according to the approach and techniques used in each.

1.1 Recommender Systems

The primary objective of any recommendation system [48, 47, 13, 50, 14] is to propose or suggest items (or people) to users of an on-line system that would be of the most interest to them. Many examples of recommendation applications can be found all over the Internet, from the friend recommendation systems of Facebook.com and the "who to follow" recommendation system of Twitter.com to the book and movie recommendation systems of Amazon.com and IMDb.com to name just a few. Each recommendation application generally relies on one way or another to extract information regarding items or people that each user may want to be recommended. In essence, there are two categories of methods of gathering such information:

a) **Intrusive**
These kind of systems can be seen as intrusive in the sense that they require explicit item feedback from the users of the system to indicate which items or types of items that the user tends to like or dislike. The most common basis of recommender systems is based on explicit feedback of items by users in the form of ratings. The general idea is simple; a system provides a ratings scale (e.g. 1-10 out 10, 1-5 stars, etc) from which a user can provide a rating for an item to indicate their general preference for a particular item.

b) **Non-intrusive**
Unlike the intrusive systems, non-intrusive systems do not require direct item feedback from the user in to indicate the items that they like or dislike, but more implicit methods are used to gather such information. Methods like this could be based on many aspects, such as the items that a user has interacted with within the system, and how many times the user has interacted with it. Examples could be the type of news articles that a user reads in a news forum, or the books that a user actually purchases, like in the case of Amazon.com.

A common form of information extraction for recommendation systems is that which relies on explicit ratings from the users, as explained above. The idea of item recommendation for these systems is to use the ratings of the items that a user has previously interacted with (i.e their ratings history) to estimate the ratings of the other items in the system for which the user has not yet provided a rating or interacted with (i.e unseen items). Having estimated ratings for unseen items, a common strategy is for the system to then recommend the top n items of this estimation process to the user, or in other words the items which were estimated to have the best ratings for each individual user.

More formally the recommendation problem can be written as follows, as given in [1]:

Let C be the set of all users and let S be the set of all possible items that can be recommended (e.g books, movies, etc). The space S of possible items and the space C of users can both be very large. Let u be the utility function that measures the usefulness of item s to user c. Then for each user $c \in C$, we want to choose such item $s' \in S$ which maximizes the user's utility. This utility is mostly represented by a rating, given by the user to indicate the user's preference for a particular item.

A common problem for recommendations systems however is that the utility function u is usually not defined on the entire search space $C \times S$. As stated earlier, for explicit user ratings, these ratings are usually only defined for the items that a user has already explicitly rated, and naturally the number of items rated by an individual user u is usually only a very small subset of the total number of items S in the system. Such *sparseness* makes it more difficult for a recommendation algorithm to provide accurate estimates for all of the unseen items for which u has not yet given a rating, as the system has limited information of the user's preferences on which to base an estimation. This is also the case for users generally termed *cold start users*, who have either just joined the network, or have simply not rated many items in the graph. For users like this, there is of course less information for the algorithm to use to accurately predict if an item would be appreciated by this user or not.

1.1.1 Categories

Depending on the method used, as well as the type of information from the network that is used to generate estimations for unseen items, recommendation systems can generally been seen to fall into two main categories, namely Content-Based systems, and Automatic Collaborative Filtering (ACF) systems.

1.1.1.1 Content Based Recommendations

The first main category of recommender systems is that of content-based systems. The main idea of content based approaches [29, 45, 38, 7] is that users will be recommended items that are similar to the items previously preferred by the user. The utility $u(c, s)$ of item s for user c is estimated based on the previous utilities (e.g ratings) assigned by the user c to items $s_i \in S$ that are *similar* to item s. The heuristics used to measure this similarity of items are often based on techniques used in the domain of *information retrieval*, particularly those keyword analysis

[49], and information retrieval [4].

Profiling

One technique for such content-based systems is to use user and item *profiling*, as done in [44]. The profile of an item can contain keywords and other information that may broadly characterize the nature of the item, or the interests and preferences of the user. For example, if we take a system that recommends films in a social network:

a) The profile of individual users could contain preference information, such as the types of films that the user prefers, or favorite actors, directors, genres, favorite films etc. These profiles can be created by either explicitly asking the user to fill out a questionnaire or by using keyword analysis techniques to analyze the content of the items for which the user has previously had interaction (e.g buying, rating, watching, etc). Using this information, the recommendation system can recommend unseen items that are similar to previously seen items, or matching the profile preferences of the user.

b) As well the user profile information, the content of the item profiles in the network can contain information and attribute of the item itself (genre, main actor, director, etc) likewise this information can be used in the process to estimate the utility of items for individual users based on the similarities of the content of items profiles using to the items previous preferred by the user, as discussed. So for example, if the user has previously given a high utility (e.g rating) to a film with Tom Cruise as the main actor, the system could perhaps recommend more films with Tom Cruise to the user.

Text-based items

Content-based systems are most commonly used for text-based items such as news articles [29] and web pages [5] and the techniques used for the recommendation, as stated above, are largely based in the domains of information and text retrieval [49, 4]. For example, the two techniques given above used profiles for items, but for text-based items such as news articles or web pages, the system could associate weights to certain *keywords* for individual users, based on some term frequency measures calculated on articles or web pages that have already been read by the user. So if a user has a passion for modern Chemistry, they may read many Chemistry articles, and the system would thus associate high preference weights to more chemistry orientated terms than say sport related terms, and could estimate the utility of new articles based on the frequency of such words and terms that would be contained in these

articles.

Drawbacks of content-based systems
The main drawback to content-based approaches is the fact that the resulting recommendations are very much limited by the features that are explicitly associated with the objects in the systems. That is, the content of the profile of a non-textual item is the only information available to relate this item to any other item in the graph, or indeed to the profile of any user. If this content does not contain a sufficient set of features of data, or perhaps misrepresents the item (e.g the genre of a film is debatable) then the accuracy and effectiveness of the resulting recommendations can decrease. As well as this, keyword frequency techniques only work on text based items, such as books or news articles. For movies and music for example these techniques can again only be applied to the descriptions of the items.

On top of this, another drawback for systems that can only recommend items relating to a user profile's is that the user is always limited to being recommended items that are similar to those already rated by the user, or genres specifically mentioned in the user's profile. This is quite a limited characteristic, where the user may never discover anything new beyond what is already known, and so the diversity of the recommendation can be limited. One possible method of mitigating this problem could be the introduction of some form of randomness into the recommendation process. For example, one might introduce one random recommendation for every ten previous recommendations.

1.1.1.2 Collaborative Filtering Recommendations

The second main type of recommender system is that of Collaborative Filtering systems. The main idea behind collaborative filtering approaches [15, 46, 8, 26] to recommending items to user is to go beyond using solely the ratings history and profile information of the individual user in question, and estimate the utility of items for individual users based on the interaction and ratings history of *other* users deemed to be *similar* to the user. A formal definition of collaborative filtering is given in [1]:

The utility $u(c, s)$ of item s for user c is estimated based on the utilities $u(c_j, s)$ assigned to item s by those users $c_j \in C$ who are deemed to be *"similar"* to user c. Taking the example of the movie recommender system as above, the system would try to find the neighbour of "peers" of

user c that have similar tastes in movies as c, and thus recommend the movies most liked by these similar peers to c. Each implementation of a collaborative filtering system differs slightly in the way that they use the ratings of similar users to derive an overall rating estimation for a particular user. The most common of these methods are grouped together as memory-based algorithms as explained in [1].

Memory Based Algorithms
For memory based algorithms, the value of the estimated rating $r_{c,s}$ for user c of item s is usually computed as an aggregate of the ratings of the N most similar users to c who have previously rated item s. Some common techniques used for this aggregation are:

$$r_{c,s} = \frac{1}{N} \sum_{c \in \hat{C}} r_{c',s} \tag{1.1}$$

$$r_{c,s} = k \sum_{c \in \hat{C}} sim(c, c') \times r_{c',s} \tag{1.2}$$

$$r_{c,s} = \bar{r}_c + k \sum_{c' \in \hat{C}} sim(c, c') \times (r_{c',s} - \bar{r}_{c'}) \tag{1.3}$$

where multiplier k serves as a normalizing factor, usually selected as:

$$k = 1 / \textstyle\sum_{c' \in \hat{C}} |sim(c, c')|$$

and where the average rating of a user c: $\bar{r_c}$ in the *Adjusted Weighted Mean* technique 1.3 is defined as:

$$\bar{r}_c = (1/|S_c|) \textstyle\sum_{s \in S_c} r_{c,s}, where\, S_c = \{s \in S | r_{c,s} \neq\}.$$

Technique 1.1 shows a simple arithmetic mean of all of the ratings that have previously been given to this item. This is quite a common technique, and gives the global average rating of this item. Although this technique does not provide any form of personalization, we will see in Chapter 3 how this technique can actually outperform the more sophisticated and personalized techniques in terms of prediction accuracy, depending on the dataset in question and the distribution of the ratings in the dataset.

Technique 1.2 shows a *weighted* mean of the ratings given to the item s, weighted with the similarity score $sim(c, c')$ computed between the rater

c' and the user for whom the prediction is for c. Thus using this technique, the higher the similarity measure computed between two users, the more contribution the rater's rating for item s will have to the estimated rating $r_{c,s}$ for user c.

Finally, technique 1.3 shows an *adjusted* weighted mean, whereby the way in which the rater c' usually assigns ratings to items (i.e the average rating $\bar{r}_{c'}$) is taken into account. Instead of using the absolute value of the rating assigned to item s by user c' as was the case for the weighted mean technique (Equation: 1.2) this technique uses the deviation of the rating $r_{c',s}$ given by user c' to item s from the average rating $\bar{r}_{c'}$ of the user c'. Furthermore, this technique also weights this deviation using the similarity measure computed between users c and c' in the same way that was described for the weighted mean technique (Equation: 1.2).

As shown above, the similarity measure $sim(c, c')$ computed between two users c and c', is used as a weight in the aggregation of the ratings previously assigned to a target item s (i.e the more similar users are, the more contribution their opinion (rating) will have on the outcome of the estimated rating of item s for user c). To calculate this similarity, the most common approach is to base the similarity of two users on the ratings they have previous assigned to the items that they have *both* rated (shared items). This similarity measure is most commonly computed using an adjusted version of Pearson's correlation coefficient [46] as shown in 1.1.1.2 between two users x and y.

$$sim(x,y) = \frac{\sum_{s \in S_{xy}} (r_{x,s} - \bar{r}_x)\,(r_{y,s} - \bar{r}_y)}{\sqrt{\sum_{s \in S_{xy}} (r_{x,y} - \bar{r}_x)^2 \sum_{s \in S_{xy}} (r_{y,s} - \bar{r}_y)^2}}$$

Similarity Measure: Pearson's Correlation Coefficient

where S_{xy} is the set of all items co-rated by both users x and y, i.e $S_{x,y} = \{s \in S | r_{x,s} \neq \& \, r_{y,s} \neq\}$

1.2 Trust in On-line Systems

The second research domain in which this master thesis is based is that of the incorporation of trust into online systems and more particularly, its usage in social networks. The concept of trust has been used in various on-line contexts, each with differing usages and interpretations of trust. Two of the main domains for which the notion of trust has become prominent are those of of on-line security and authentication as well as that of content personalization and recommendation and reputation systems.

For each domain, trust has been used to enhance the effectiveness of the systems by using the knowledge of *"trusted peers"*.

In the domain of recommender systems, as we will see in further detail, trust has been used to enhance the prediction accuracy of rating estimations by filtering the information available for items previously unseen by a user by considering only the information provided by the user's trusted neighbours.

In the domain of authentication and security, on the other hand, trust can be seen to constitute a reliance on and a belief in another party, without which the system which relies on the trusted party cannot operate correctly (e.g PKI systems, chain of certificates [53, 52, 61]).

As we can see from these brief descriptions, and as introduced earlier, the definition of trust and its usage differs significantly for each domain in which it is used. For the purposes of this work, I will concentrate solely on the interpretation of trust from the domain of content personalization and recommendation, which sees trust as metric that can be used to improve the accuracy of content personalization and item recommendation. I will begin by introducing the two main types of trust used in this main, local trust, and global trust.

1.2.1 Local & Global Trust Metrics

As mentioned in [63], one of the main distinguishing characteristics of all of the trust inference algorithms is whether they propose the use of a *local* trust metric, or a *global* trust metric.

- **Local Trust Metrics** are personalized for every user in the network. They take into account the personal opinions and bias of the individual user. Algorithms that use local trust inference compute different trust values for a node, depending on who the source the trust is. These sort of algorithms can be seen to be more appropriate for opinion based applications, where there is no shared opinion of what is good or bad, but everybody has their own different opinions.
- **Global Trust Metrics** on the other hand take into account the opinions of all peers and trust edges in the network connecting them. Algorithms using this metric compute one single trust value for every node in the system, so that the trust a node a has for another node b will be the exact same trust value that another node c has for b. Algorithms using global trust inference can be seen to be more appropriate to applications where trust reflects behavior that is universally considered good or bad..trustworthy or non trustworthy.

For the reasons given above, local trust metrics generally tend to be involved in the proposals for trust-based recommender systems for items such as books or movies, which are more based on the opinion or taste of individual users, like that proposed by [65]. On the other hand, global trust metrics are more generally proposed for systems such as peer to peer, file sharing applications, where the measure of trust can be more difinitive, and does not depend on the individual tastes of users. For the file sharing site for example, it can be seen that a node can be either good and reliable at providing files or not reliable and malicious.

1.2.2 Social trust graph

As described in [25], social trust graphs are weighted directed graphs, whose nodes represent agents or users of a system, and whose directed edges represent trust relations between these agents. The weights of the directed links between nodes represent the trust values from one node to the other. So, an edge from user u to user v with trust value t means that u trusts v to the extent of t.

This model of social graph can be extended to model almost all existing social networks. Apart from the social graph which we use for our experiments, one real life example of such a network is Twitter.com, with each vertex representing users, and the directed edges being the relationships "*follows*" whereby one user u may follow another user v , thus building a *follows* social graph.

1.3 Trust-based Recommender Systems

The area of trust-based recommender systems has been a growing area of research for the past number of years. The incorporation of trust into recommendation algorithms has been shown to provide significant improvements to the classical collaborative filtering techniques [46, 1]. The main difference between most of these trust-enhanced methods is the acquisition of the trust values themselves between actor pairs. Many algorithms have been developed to acquire trust in social graphs, of these here we present a few.

1.3.1 Obtaining Global Trust

One example of a method that has been proposed to infer global trust metrics users in a peer to peer file-sharing system is the EigenTrust algorithm presented by Kamvar et al in [28]. This algorithm presents a

method for the inference of global trust values where trust is seen as an overall measure of reliability and performance. According to the authors, EigenTrust aims to decrease the number of downloads of inauthentic files in a peer to peer file sharing application and reduce the effectiveness of malcious users in the network by assigning each peer in the network a unique global trust value. This global reputation value is based on the peer's history of file uploads, and the experiences that the other peers in the system have had with this peer, and the files uploaded by this peer. As stated in the paper, the algorithm was designed for the following:

a) Minimal Overhead: In terms of computation, infrastructure, storage, and message complexity

b) Not Profit Newcomers: Reputation is only obtained through consistent good behavior

c) Distributed and Self Policing

d) Robust to Malicious Collectives who try to collectively subvert the system

Each peer thus maintains a level of trust for each of the other peers with which it has interacted. This local trust score, assigned by individual peers, is based on the proportion of good files that the assigning user has received from the other peer in the network for which it is providing a local trust value. As well as this local trust value, each peer in the network is also associated with a global reputation score, assigned to each peer by the EigenTrust algorithm. In its essence, the global reputation score assigned by EigenTrust to a peer i in the system is computed using the local trust values that have been assigned to peer i by the other peers in the network, weighted by the global reputations of the assigning peers. I will now give a brief overview of the EigenTrust algorithm.

To begin with, the algorithm normalizes the local trust values in order to stop malicious users assigning arbitrarily high local trustvalues to other malicious users, as well as arbitrarily low values to honest users. For this, they define a *normalized local trust value:* $c_{i,j}$ as:

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

thus ensuring that all values will be between 0 and 1. The algorithm then aggregates these local trust values: peer i asks its neighbors about their opinions about other peers, and then weights these opinions by the trust peer i has in them:

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

where t_{ik}represents the trust that peer i places in k based on the opinions of its trusted neighbors.

In its simplest form, the algorithm uses a matrix representation of these trust values, where C is the local trust adjacency matrix $[c_{ij}]$ and $\vec{t_i}$is the vector containing the values t_{ik}, then $\vec{t_i} = C^T\vec{c_i}$, and over a series of iterations: $t = (C^T)^n c_i$,it converges to a globally accepted trust rating for each peer. If n is large, vector $\vec{t_i}$ will converge to the left principle eigenvector of C, that it quantifies how much trust the system as a whole places in peer j.

1.3.2 Propagating Local Trust using explicit trust

1.3.2.1 Using Propagation Rules

One of the most common ways to infer trust relationships between users in a social graph who have not yet interacted with each other or provided trust values for each other is to use a method of propagating of trust through the network based on a set of trust propagation rules. The basis for these propagation rules is to view trust values to hold some form of transitivity property.

One example of a set of propagation rules for trust values in a network is that present in by Guha et al in [24]. Although other methods of trust inference use slightly differing forms of trust propagation rules, the set of propagation rules proposed here was one of the first to make a major contribution to this field, and was the stepping stone for many of the path inference algorithms that were to follow, some of which I will present in the next section.

In this paper, trust relationships between two users are represented as values between the interval of 0 and 1 in a matrix of trust C, where C_{ij} represents the current inference of $i's$ trust for j. As well as this, they also introduce the idea of distrust to be modeled as negative trust. From these matrices, they then use a "basis set" of techniques, as shown in Table 1.1, by which the system may infer that one user should trust or distrust another.

Direct Propagation	M	A trusts B, so $trust(A)$ propagates to B
Co-citation	M^TM	A trusts B,C, so $trust(B)$ propagates to C
Transpose Trust	M^T	A trusts B, so $trust(B)$ propagates to A
Trust Coupling	MM^T	A,B trust C, so $trust(A)$ propagates to B

Table 1.1: Basis Set of Inference Techniques

Each technique from the basis set of Table 1.1 thus extends a conclusion of trust through a constant length sequence of forward and backward steps through the trust graph. Using these techniques, at each step of the propagation, the trust matrix C is replaced by a new matrix $C.M$ representing one step of propagation. Thus, M is an operator that encodes one of the propagation techniques in the matrix.

Then, using the vector $\alpha = (\alpha_1, \alpha_2, \alpha_3, \alpha_4)$, representing weights for combining the atomic propagation techniques above, $R_{M,\alpha} = \alpha_1 M + \alpha_2 M + \alpha_3 M + \alpha_4 M$ captures all atomic propagations into one matrix.

Then, trust is propagated through a series of applications of the basis set, before arriving at a final belief matrix $P^{(k)}$ whose ijth entry represents the propagation from i to j after k applications.

1.3.2.2 Path Inference

Using trust propagation rules similar to those presented above, one of the more common methods of inferring trust through a social trust graph has been that of path inference. To infer such trust values between two users a and b who have not yet interacted with each other in the graph , the main focus of path inference algorithms is firstly to discover a trust path, consisting of trust assertions between intermediate users, linking user a to user b. This path would thus link the user a to user b through the users that have previously been trusted by a.

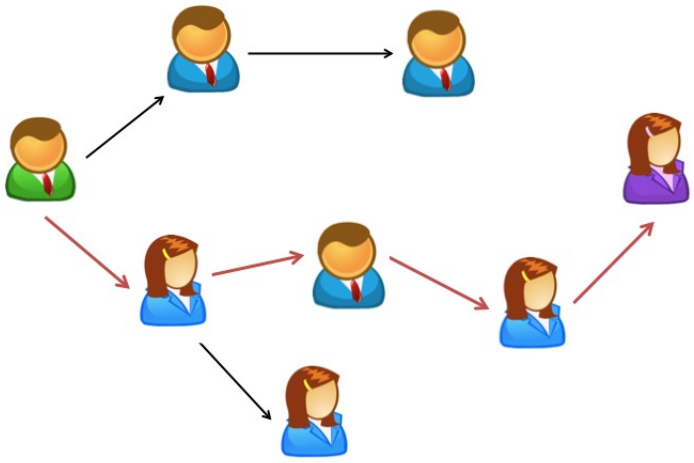


Figure 1.1: Trust Path

If a path is found between these two users (users a and b), the trust value to be inferred between them is generally inferred through some sort of aggregation of the trust values that exist between the intermediate users in the path. For example, one possible way which has been used to aggregate these values, is simply to weight the trust value received by a from a neighbour n for a target user b by the amount of trust a has attributed to n, this process is also carried out by each intermediate user along the discovered trust path. Thus, using this aggregation, trust values are propagated and discounted through the graph.

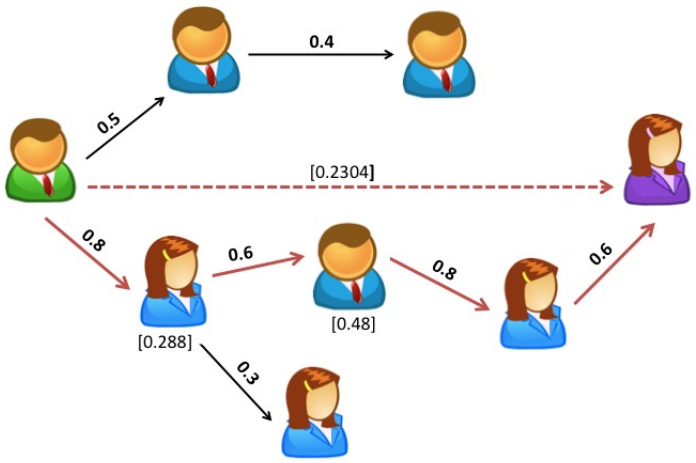


Figure 1.2: Trust Aggregation

Many algorithms of this nature have been proposed [58, 35, 37, 19], each of which relies on the assumption that trust is in some way *transitive*, and therefore can be propagated through a trust graph.

In [19], Golbeck introduces an algorithm called TidalTrust to estimate trust values between actor pairs in a social network. This algorithm uses trust values that are explicitly provided by the users of the network, and trust estimates are thus propagated through the network using a modified breadth-first search and propagation rules similar to those of [24] presented above. The resulting estimate of the trust value from user u to user v is thus the weighted average of the trust scores attributed to v by users which u has already trusted (i.e $u's$ trusted neighbours) along a trust path.

The basic concept of the algorithm is that if a user u wishes to obtain an estimated value of trust for another unseen user in the network v, the user issues a request for a trust estimate (a trust query) to all of its

trusted neighbours. If one of u's trusted neighbours has a direct trust value for v, this value is returned directly to u, if they do not have a direct trust assertion for v, each neighbour forwards the trust request received from u to all of *their* trusted neighbours. Each trust value returned to u by a trusted neighbour n is then weighted according to the trust value that u has attributed to n. This same weighting process occurs when trusted neighbours receive answers from their neighbours to trust queries they have forwarded along the trust path. An additional feature to this general aggregation process is that TidalTrust eventually uses the strongest trust path from the requesting user u to the target user v.

Massa et al [35, 37] developed a very similar approach to that of Golbeck's TidalTrust, which they call MoleTrust. MoleTrust also incorporates the use of explicit user trust assertions and uses this information to propagate trust through the network in two phases. Firstly, all cycles in the graph are removed, thus turning the trust graph into a directed acyclic graph. Then, similar to the propagation process used by TidalTrust, trust values are propagated to a source user u for another user v using a trust-based weighted mean of the trust values attributed to v by the trusted neighbours of u. As well as this, MoleTrust incorporates an extra parameter into the propagation phase called the *trust propagation horizon*. This parameter specifies the maximum hop distance in the graph from the source user u for which trust is propagated, meaning that the length of trust path used to infer trust between two users is limited to the specified trust propagation horizon. This parameter is used for two reasons, firstly to reduce computational cost, but also, as they say, because the reliability of the propagated trust decreases with every new trust propagation hop.

1.3.3 Automatically Inferring Local Trust

In this section, I present some of the methods that have been proposed to automatically infer trust between users.

1.3.3.1 Trust Clustering

Drawing upon the conclusions of [65], which shows a correlation between the trust one user has in another and the similarity between the two, as well as the results from such papers as [51], that the use of trust can have a positive impact on the accuracy of recommendation systems, [16] propose a method of improving the accuracy of of recommendation systems by using a trust clustering algorithm, to cluster the users in a social network according according to trust.

The solution proposed is based upon a probabilistic trust inference algorithm used on a network of trust as explained above, and based on Bayesian chains. From this inference of trust, they then create a *trust distance metric space*, where the more trust that exists between a pair of users, the closer they are in the space. Thus grouping users in this metric space according to their trust metrics. To illustrate their idea they use the example:

a) Alice knows Bob and thinks he has a $P_{a,b}$ chance of being trustworthy

b) Bob knows Eve and think that she has a $P_{b,e}$ chance of being trustworthy, and he tells this to Alice if he is trustworthy. If he is not trustworthy, he may lie and give any value to Alice

c) Alice reasons that Eve is trustworthy if Bob is trustworthy and gives her the correct value $P_{b,e}$ and Eve is trustworthy with respect to Bob

d) This combination happens with probability $P_{a,b}P_{b,e}$ if Bob's and Eve's trustworthiness are independent

They then illustrate that if trust is a proxy for similarity, then Alice and Bob's mutual trust can be a measure of similar tastes for any sort of item, like movies. Therefore, if trust is interpreted as the probability of liking the same film, then Alice will agree with Eve about a movie if Alice and Bob each agree on it and Bob and Eve agree too.

Using this inference they claim to easily estimate trust between individuals in a highly complicated graph, one with exponentially many, highly correlated paths, by viewing every trust path as a Bayesian chain. Such that, if there exists a path from Alive to Eve in a random network constructed from trust values, then this path is a chain of people from Alice to Eve who each trust their successor, and Alice can thus trust Eve. Therefore, Alice trusts Eve with the probability that there is a path from Alice to Eve in the random graph. From this edge assigned between two pairs based on some probability, they choose a mapping a function f from trust value to probabilities and create a random graph where each edge (u, v) exists with some probability $f(t_{u,v})$ where $t_{u,v}$ defined to be the direct trust between u and v. They then use this graph to generate inferred trust values $T_{u,v}$ such that $f(T_{u,v})$ is the probability that there is a path from u to v in the random graph.

Using this graph created from the probabilistic trust inference, they then performed a *correlation clustering* algorithm over the graph, grouping people together who have more trust for one another. The goal of a correlation clustering algorithm is to find clusters that maximize agreement within, and minimize agreement between clusters. For this application

however, the trust value from one node to the other is used as the measure of similarity, with high trust indicating agreement, and low trust indicating disagreement.

From the resultant clusters, they then produce a series of tests using different recommendation algorithms, using higher weights for ratings from nodes that are in the same cluster as the requesting node. Their results show a statistically significant improvements between non clustering techniques and alternatives, using the Mean Absolute Error as well as the Root Mean Squared Error as comparisons.

1.3.3.2 Trust based on recommendation history

O'Donovan and Smith [51], developed another method to automatically generate trust between users based on ratings history between the two users. The resulting trust metric can be seen as a reliability metric between two users. In this work, the authors distinguish between two types of trust, item-level trust and profile-level trust. Trust is then built up between users u and v, by measuring the reliability of $v's$ past recommendations for u, which is seen as the percentage of predicted ratings v has made for u which have been within a certain threshold of $u's$ actual rating for the recommended item. This trust is then seen in two levels, as a general reliability score for v dubbed the *profile level* or at a finer grained *item level* which measures reliability of v to recommend item i.

1.3.3.3 Trust based on user tastes

Wang et al [59], develop a method to generate trust between users based on their common *tastes*. By first grouping items into different classifications, the authors use a frequency measure of the number of ratings each user asserts to different classifications of items, and thus build up a personalized taste set for user and infer trust based on the common taste sets between users.

1.4 Empirical Analysis of Trust in Recommender Systems

All of these methods presented above have been used to generate trust estimates between users, and subsequently use these trust relationships in various collaborative filtering techniques to generate personalized predictions of items to users. Each study has shown that the incorporation

of trust into recommendation techniques improves the accuracy of recommender systems in comparison to basic collaborative filtering, especially where the user's rating for an item differs from the average rating for this item.

For the second part of this master thesis, I compare some of these different techniques proposed to incorporate trust information into recommender systems, in order to analyse and compare these techniques, as well as to attempt to validate the claims of the trust-based recommender system community. Previously some papers have also dealt with such analysis.

An empirical comparison of some of the more used methods of incorporating trust into collaborative filtering techniques was made by Victor et al in [57]. This analysis compares a number of different trust-enhanced recommendation techniques in terms of prediction coverage, and average error rate. These techniques were applied to a set of controversial reviews from the Epinions.com dataset, for which they develop an algorithm to identify controversial reviews based on the level of disagreement between the ratings in this set.

In a similar work, Victor et al [55] also compared the different algorithms for generating and propagating trust values, TidalTrust, MoleTrust and O'Donovan et al. Again, this comparison was based on the application of these methods again to controversial items in the network. However, this comparison was performed on only one dataset, that of the Epinions.com dataset.

CHAPTER 2

STRUCTURAL TRUST INFERENCE

2.1 Introduction

This chapter presents an investigation into the development of a method to infer trust between user pairs of a social network graph. This represents the main focus and contribution of this master thesis. As we will see, the trust inference method developed herin represents a reusable and generic method using only the structural and topological information of the graph typical of many online social networks, and incorporates a novel methodology based on the items shared between user pairs in the graph.

I begin by introducing the motivation for this trust trust inference formula, before giving a formal definition of the bipartite structure typical of many social graphs, which will be the basis for the inference formula. I then present the main methodology of the popularity of shared items that is used in the trust inference formula. For this, I provide some real life observations that inspire the intuition behind this methodology giving examples from real life social networks. I then introduce and explain the final formula used, as well as the structural features of the bipartite graph that it uses before presenting the method followed and the experiments undertaken to firstly train and then validate the formula using a social dataset taken from an on-line social network.

Following this training and validation phase, I then compare the proposed approach to a naive approach which also uses similar structural features of the network as the proposed method. For this, I provide a comparison of results based on the prediction rate for both trust edges between users in the graph, and the prediction rate of distrust edges between users. As

well as this, I plot the structural properties of the new graph computed from our formula to check their validity against those properties typical of social networks by comparing these properties to those of the real graph of the dataset used.

2.2 Problem Formulation

Previous studies [57, 20, 34, 35, 60] have shown that the use of trust in the context of social networks and content recommendation can provide a number of benefits. Trust has been shown to improve the accuracy performance for content recommendation [37, 55, 51, 59, 33], as well as file-sharing peer-to-peer systems [28]. With such advantages, it is clear that the availability of such connections is invaluable to the enhancement of on-line social recommendation systems and on-line user experience in general.

As stated in the introduction of this document, trust information between users is most often provided through explicit feedback from the actors in a social network themselves. But of course, users are only able to provide such explicit feedback in just a few on-line social networks, with the large majority of social networks not explicitly providing such functionality. Previous studies [32, 18, 62, 17] have been made to predict such trust relations based on probabilistic models. However, to the best of our collective knowledge, no algorithm yet exists that allows for the automated inference of such trust based solely on the topological information of a bipartite graph. This investigation is thus motivated by the need for a generic method for the automatic inference of trust connections between actor pairs in a social graph. These trust metrics should be fit to be used by trust aware algorithms that have been designed to enhance on-line user experience, such as trust aware social recommendation systems, thus allowing their application without the need for explicit user feedback, nor any knowledge of the content of the graph.

2.2.1 Structure of social networks

Many on-line social networks and recommendation systems [1, 46] can be seen to deal with a bipartite graph representing a set of actors (e.g. users) connected to a set of items (e.g. books), as shown in figure: 2.1. Each connection corresponds to an act through which an actor performs an operation on an item (rating, buying, commenting, etc).

Formally, let $G = (A \cup I, E)$ be a bipartite graph where A and I are two disjoint sets, the set of *actor* and the set of *item* vertices respectively,

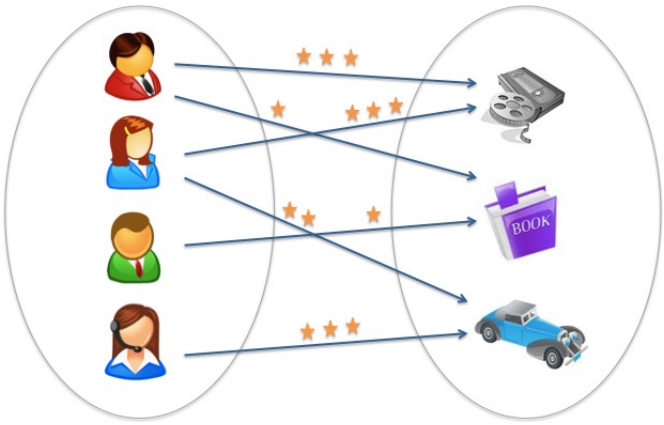


Figure 2.1: Bipartite Social Graph

and $E \subseteq A \times I$ is the set of edges (i.e. interactions between actors and items). The difference with a classical graph lies in the fact that edges only exist between actor vertices and item vertices. Recommendation algorithms aim to predict a set of edges $e \in E$ that are *relevant* to the individual actors.

Trust aware social recommendation provides a prediction by means of the trust information between actors, consisting of a set of relations between actors within the set A (Figure: 2.2).

As stated in the introduction, this investigation follows the notion of trust as an indication of similarity or commonality between two users in a network. Based on this, this work aims to provide a measure that computes the trust between actors based on the whole bipartite graph.

Formally speaking, from a bipartite graph $G = (A \cup I, E)$ describing interactions between actors and items, we want to create a graph $\mathbf{g} = (A, T)$ where A is the set actor vertices and $T \subseteq A \times A$ is the set of edges representing the trust relations between actors (Figure: 2.2).

2.3 Trust inference

This section presents the methodology, based on intuition inspired from real life observation, used in this work for the inference of trust relations between users in a social graph. I will then introduce and explain the formula, based on the described methodology, which was chosen for the implementation of this trust inference.

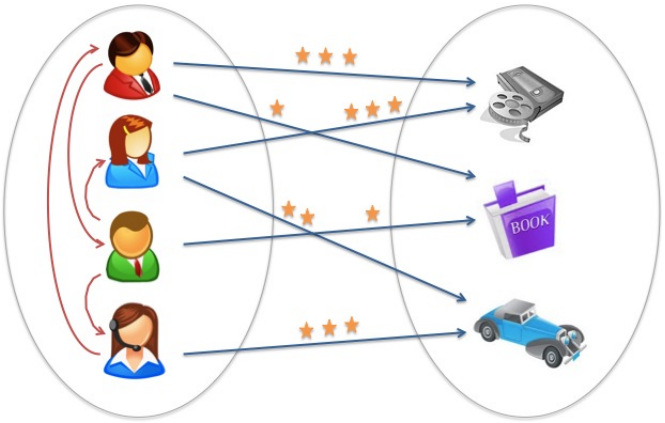


Figure 2.2: Inferred Trust Relations

2.3.1 A methodology for inference of trust

From the structural information of the social bipartite graph presented above in section 2.2, to infer trust connections between users this investigation focuses on the common relationships between vertices in set A to those in set I. For a social bipartite graph consisting of two distinct sets of vertices, set A and I (section 2.2), a *shared item* between two vertices in A is defined to be any vertex in set I for which both vertices in set A have a directed edge. This work distinguishes these *shared items* according to their relative *popularity* in the graph, defined as the indegree of each item, or the number of directed edges from vertices in set A to this vertex in I. Based on real life observations, this work follows the intuition that the higher the indegree of a vertex in set I, the *less* that can be deduced about the similarity between two vertices in A who both have a directed edge to this vertex, and thus, the less we can say about the potential trust relationship between them. This intuition is inspired from real life observations of the popularity of items and people in a social context. Considering a real life example of the book "Harry Potter", which has been the subject of widespread popularity and attention for more than a decade, if two users of a social network such as *Amazon.com*, were to provide a positive rating for "Harry Potter" (figure: 2.3), we argue that there is little that we can deduce about their relative similarity, as the approval of such a popular and widely known item may well be partially due to the popularity and widespread appeal of the item in general and does not constitute a distinguishing character trait.

However, this intuition is not limited solely to books or items. If we

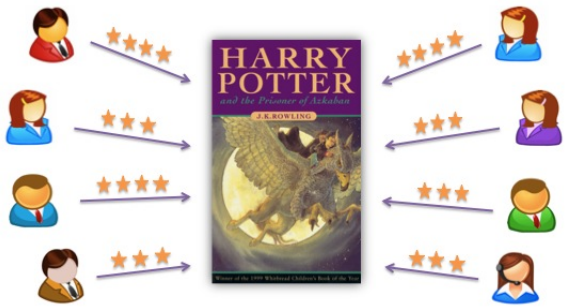


Figure 2.3: Intuition of Popular Items

were to take the "follows" graph of the social network *Twitter.com*, we can consider a famous singer or actor and apply the same intuition. For example, the English actor Stephen Fry is a popular and well known public figure, who happens to be an avid user of Twitter.com. At the time of the writing of this document, Mr. Fry is followed by 4,348,328 users on Twitter.com. Using the same intuition, we argue that there is little we can deduce about the similarity or potential trust connection between two users of Twitter.com who both "happen" to follow Stephen Fry. As before, such a connection may well be more likely based on the popularity of the public figure more than a strong similarity or character trait between the two followers.

Following the same methodology, and based on the proposed correlation of user similarity and trust [65, 21, 64], this work states that the lower the indegree of a vertex in I, meaning the less popular a particular item is in the graph (figure: 2.4), the *more* that can be deduced about the similarity between two vertices of set A who have an edge to this vertex, and thus, the probability that they will have similar tastes will be greater. As such, a connection is more likely to be based on a genuine interest in such an item and not on coincidence or on the popularity of the item itself.

Further to this intuition, we also take into account the concept that trust can be built through other means within social networks. To take a concrete example common to most social networks, users may access publicly available comments from other users in the network and may agree or disagree, thus a user may subsequently trust the user who issued this comment directly through such means with a certain probability. This is also taken into account in our inference of trust, as will be seen in the following section.

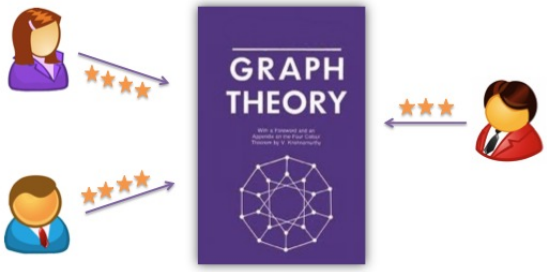


Figure 2.4: Intuition of Rare Items

2.3.2 Deriving a formula for trust inference

Building upon the methodology of *shared items* presented above, we believe that there are two main structural factors that need to be taken into account in order to infer trust connections between two users in our social bipartite graph. Firstly, we believe that it is necessary to take into account the *Relative Diversity* between the two users, which we define as the number of neighbours that both users can reach through two hops in the graph. Using only the topological information of the graph, we follow the approach of *structural similarity* as presented in [30], using the *Jaccard Index* to compute a distance measure between vertices in set A based on the neighbourhood of each vertex. As we are dealing with a bipartite graph, each vertex does not have a direct connection to a vertex in the same set.

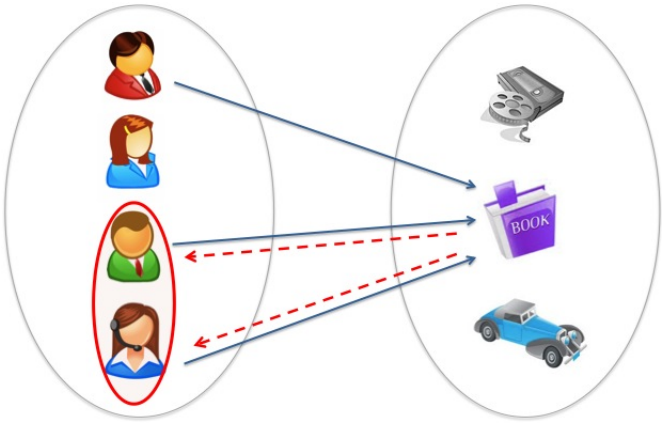


Figure 2.5: Two-hop Neighbourhood

Thus, this work considers the neighbourhood of a vertex $u \in A$ as the

set of vertices $S \in A$ through which vertex u has an indirect connection in the graph through the vertices in set I for which u has a directed edge. We define this as the *two-hop neighbourhood* of u, connecting u to vertices in the same set, through $u's$ interaction with vertices in set I. To compute this relative diversity between two vertices u and v, we thus consider both of their two-hop neighbourhoods as two sets, and we apply the *Jaccard Index* between these sets. The Jaccard index [27] is a well known statistic, widely used to compare the similarity and diversity of sample sets and perfectly suits our need to compute the relative diversity between two vertices in our bipartite graph. This formula is presented below in equation 2.1, where N_u represents the neighbourhood of vertex u and N_v represents the neighbourhood of vertex v.

$$J(u,v) = \frac{|N_u \cap N_v|}{|N_u \cup N_v|} \tag{2.1}$$

The second structural factor we believe to contribute to social trust is that of the intuition of shared items presented above in section 2.3.1. Based on this intuition, we need to provide the distance between two vertices in set A in relation to the *popularity* of the vertices in set I for which they both have a directed edge. The formula used to compute this distance value based of shared vertices is presented in equation 2.2, where $deg(i)$ represents the indegree of the shared item i. The more highly connected a shared vertex, the higher the resulting distance value will be, and consequently, the less connected a shared vertex is, the lower the distance value will be. Thus, this equation rewards low connected shared items, and penalizes highly connected shared items.

$$D(i) = \left(\frac{2}{1+e^{(-deg(i)^{\sigma}+2^{\sigma})}} - 1\right) \tag{2.2}$$

Figure 2.6 shows the behavior of this formula as a function of the degree of an item and the constant parameter σ. As we can see from this curve, as the degree of the item i increases, the output value $D(i)$ also increases exponentially. This perfectly fits our methodology of rewarding low connected items while penalizing highly connected items. The resulting values $D(i)$ are normalized in the interval $[0, 1]$. Moreover, the parameter $0 < \sigma < 1$ is incorporated into the equation in order to provide a way to adjust the slope of the curve. Concretely, this parameter

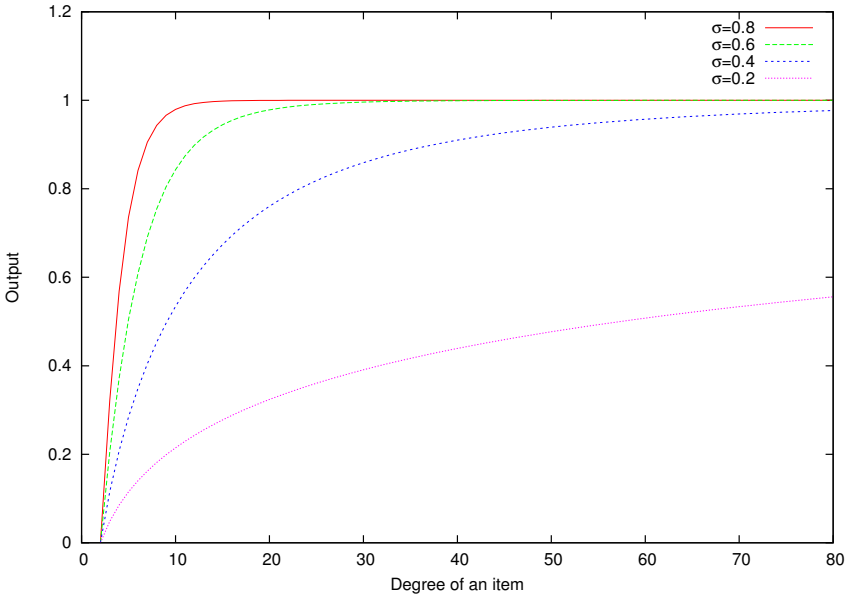


Figure 2.6: Formula for Shared Items

allows to modify the distribution of the values $D(i)$ over $[0, 1]$. In practical terms, the increase of σ causes $D(i)$ to rapidly reach high values, meaning that in the case of $\sigma = 0.8$, when the degree of the item is near to 10, the computed $D(i)$ values will be very near to 1. However, in the case of $\sigma = 0.2$, the $D(i)$ values reach 1 with items having degrees ≥ 1000. In other words, this parameter is used to define from which degree value an item is considered as popular. This will depend on the data sets involved. In addition, the minimum value 0 of $D(i)$ is obtained when the degree of the involved item is set to two, whatever the value of σ. In practical terms, this corresponds to the case where an item is only rated by the users involved in the computation themselves.

By combining these two aspects, both the relative diversity and the distance based on shared vertices, the proposed trust inference formula is presented as a whole in the below equation 2.3.

$$Trust(u, v) = \alpha + \beta J(u, v) + \gamma(1 - \frac{\sum_i^{i \in SI} D(i)}{|SI|}) \quad (2.3)$$

where SI is the set of shared items between the users involved, as well as the parameters $\alpha + \beta + \gamma = 1$. The constant α defines the probability that a pair of users trust each other through any other form of external information (i.e. recommendation, search engine, etc). This parameter

is inspired by the "teleportation" parameter used in the PageRank algorithm [9, 43] which defines the probability of the direct access to a web page (without following hyperlinks). The constants β and γ define the contribution of each proposed factor to the computation of the trust between a pair of users.

2.4 Experimental Evaluation

2.4.1 Data set

In order to test the proposed methodology, it was necessary to be able to compare the results of the tests performed on the trust inference formula against meaningful real life trust assertions. For this, we needed real life test data consisting of:

a) A real life social bipartite graph containing two sets of vertices, A and I, representing a set of actors, and a set of items respectively. This graph should also contain directed edges from actor vertices in set A to item vertices in set I, representing explicit ratings of items by users. This graph, hereon referred to as the *ratings graph*, will be used for the application of our trust inference formula.

b) A corresponding real life social trust graph, containing explicit trust assertions between the user vertices in set A. This trust graph must belong to the same social network and be complementary to the ratings graph.

For the purpose of these experiments, I used the *Epinions* dataset available from trustlet.org. Epinions.com is an on-line social network where users contribute reviews and share their opinions on any number of items or topics, from books and DVDs to holidays and restaurants. Users can also provide ratings on a scale of 1 to 5 for these items. In addition to this rating system, epinions also provides a "*Web Of Trust*" facility, whereby users can explicitly provide "*trust*" assertions, indicating their individual trust for other users in the network. These ratings, as well as the web of trust service are used to provide recommendations for item reviews deemed to be most applicable to individual users. Importantly, these ratings are also used to designate the top ranked reviews for each item. The more highly rated a review is, especially if these reviews are provided by highly trusted users, the more prominent position this review will take.

This dataset provides all aspects of a dataset necessary to evaluate our methodology. *Epinions* is also a well known dataset and has been used in

numerous previous studies [24, 37] using trust. One drawback of the use of the trust graph of this dataset, is that the trust assertions it contains do not provide any weight of trust between users on any scale, and indeed do not provide any information of possible "distrust" between users, whereby a particular user may explicitly not like or not trust another user in the network. The trust assertions can be seen as simple directed edges between users, where the existence of an edge indicates a trust assertion from one user to another, while the absence of an edge does not indicate whether these users explicitly do not trust each other, or if these users have not yet come into contact in the graph, or indeed if these users have just failed to provide any explicit feedback to the system.

2.4.2 Setup

For the remainder of this chapter, the ratings graph of the Epinions dataset is considered to be a bipartite graph $G = (A \cup I, E)$, with vertices in set A representing the users of epinions and the vertices in set I representing the items of the dataset, and $E \subseteq A \times I$ representing the edges of ratings of items in I from users in A. To set up the test data, I first removed all users in the trust graph that had not rated any items in the corresponding ratings graph. These users are not essential to this experiment and they go against the intuition of shared items behind our methodology. This elimination however had almost no effect on the experiments, as the number of users who had not rated any items was insignificant. I then split the experiments into two separate phases: a training and validation phase, and a comparison phase whereby I compare the proposed trust inference approach to a naive approach based loosely on the structural methodology in order to provide a further validation of the formula.

Firstly, for the training and validation phase, I began by following a classical *holdout* style validation and applied the proposed formula to an independent training set of the Epinions dataset to create a new trust graph, which consisted of trust links between the users computed by the trust inference formula.

By comparing the resulting weighted edges of the generated trust graphs computed from the formula with the corresponding real trust assertions provided in the real epinions trust graph I was able to determine local optimal values of each of the parameters of the formula as well as to validate it along two key axes as will be explained later in section 2.4.3.

For the comparison phase, I first compare our method of trust inference against a similar but naive trust inference method which is loosely based on the same methodology. Following this comparison, I then analyze the

structural properties [2, 31] of the computed trust graph and compare these properties to the corresponding properties of the real trust graph of the epinions dataset. I thus base this comparison on the following structural properties typical of on-line social networks:

a) degree distribution

b) hop plot

c) clustering coefficient properties of the graphs.

2.4.3 Validation & Training

The aim of the validation phase was to both validate the methodology proposed in section 2.3.1, as well as to optimize the parameters α, β, and γ which weight the different features of trust inference formula.

To do this, I needed to discover the relative contribution of each aspect of the formula (Eq. 2.3) to the overall accuracy of the computed trust edges of the generated trust graph. Firstly, I chose the value of parameter $\sigma = \frac{1}{3}$, as this value gave a balanced distribution of the resulting metrics for this dataset, making sure that the resulting trust metrics computed by the formula did not increase too quickly for items with an average indegree, but also so that they are also in line with the proposed methodology, as presented in section 2.3.1 of penalizing highly connected items. This, of course, would depend on the dataset in question, and may indeed change over time, as the network evolves and the items in the network become more and more connected as users continue to rate more items.

Following a classical *k-fold method* of cross-validation, I split the Epinions ratings dataset into two independent subsets:

- Training set
- Test set

I subsequently took the training set and split it further into subsets of 1000 users. For each subset, I then applied the trust inference formula as proposed in equation 2.3 which computed the trust links between each user. Taking this resulting computed trust graph I thus observed the effect on the resulting computed trust metrics in the generated trust graph by comparing these edges to the corresponding edges in the real trust graph provided with the Epinions dataset.

As the edges of the Epinions trust graph only indicated the existence of trust or not as discussed in section 2.4.1, and did not provide any scale of the level of trust (e.g 0.7), we chose a threshold for this comparison

to indicate whether the computed trust metrics were to be considered as a trust of not. If the weight of a computed trust edge was less than this threshold, this edge was considered to indicate no trust, and thus if the weight was greater than or equal to the threshold, this edge was considered to be a trust edge. The threshold that was chosen for these experiments was the arithmetic mean of all of the trust values computed by the formula in the computed trust graph, thus edges with values above the mean value were taken to be trust edges, and those with values below were taken to be no trust edges, as above.

By comparing each edge in the computed graph to the corresponding one in the real graph we validated our formula by computing the following metrics:

- **True positives** (TP)
  The number of edges in the computed trust graph considered to indicate trust (i.e with value above the mean value), corresponding to an existing trust assertion edge in the real trust graph
- **False positives** (FP)
  The number of edges in the computed trust graph considered to indicate trust (i.e with value above the mean value), but where no corresponding trust edge existed in the real trust graph
- **True negatives** (TN)
  The number of edges in the computed trust graph considered to indicate no trust (i.e with value below the mean value) for which no corresponding trust edge was present in the real graph
- **False negatives** (FN)
  The number of edges in the computed trust graph considered to indicate no trust (i.e with value below the mean value), but where a corresponding trust assertion existed in the real trust graph

From these four metrics, in order to find the correct measures allowing us to quantitatively validate our formula we focused on four key questions:

a) How many real trust assertions contained in the real graph can we predict?

b) How many real no trusts or absence of trust in the real graph can we successfully predict?

c) What is the ratio of the number correct trust assertions prediction compared to the number of trust assertions predicted that do not correspond to a real trust assertion?

d) What is the ratio of the number of correct trust assertions predicted to the real number of trust assertions in the real trust graph?

Taking measurements inspired from the domain of *information retrieval* [42, 54], we thus calculated the following metrics:

- **Trust prediction rate**: The fraction of the real trust assertions in the graph correctly identified, as the number of true positives divided by the sum of the number of the true positives and the number of false negatives:

$$\left(\frac{TP}{TP+FN}\right) \tag{2.4}$$

- **Distrust prediction rate**: The fraction of correctly predicted distrust assertions, or the number of metrics computed to indicate the lack of trust in the real graph, as the number of true negatives (TN) divided by the sum of the number of true negatives and the number of false positives:

$$\left(\frac{TN}{TN+FP}\right) \tag{2.5}$$

As part of the training of the formula was to find good values for parameters α, β, γ, these measurements were calculated for each of the subsets of the training set as described above, with different combinations of values for each parameter. The final results for each combination of values was computed using a simple arithmetic mean of the results computed for each of the subsets. By repeating this step and applying different weights to the different aspects of the formula we were able to retrieve local optimal values of the parameters α, β and γ.

2.4.3.1 Training & Choosing local optimum parameter values

Figure 2.7 shows partial results for the k-fold cross-validation training phase performed on the subsets of the training set. As we can see from the figure, the combinations of values for each parameter were validated using the Trust Prediction Rate measurement (Eq: 2.4) which is shown in blue in figure 2.7, as well as the Distrust Prediction Rate measurement (Eq: 2.5) shown in red in figure 2.7.

As opposed to making a choice on the combination of parameter values based solely on one of these two measurements, I also added the results of these two measurements together to take into account the performance in relation to the combined results shown in green in figure 2.7. This was done in order to make sure that the chosen combination of parameter values did not overly emphasize either one of the measurements, and gave a balance between the two. More plainly, we did not want to choose a combination of values which may have a high number of True Positives and therefore have a high Trust Prediction Rate, but also give

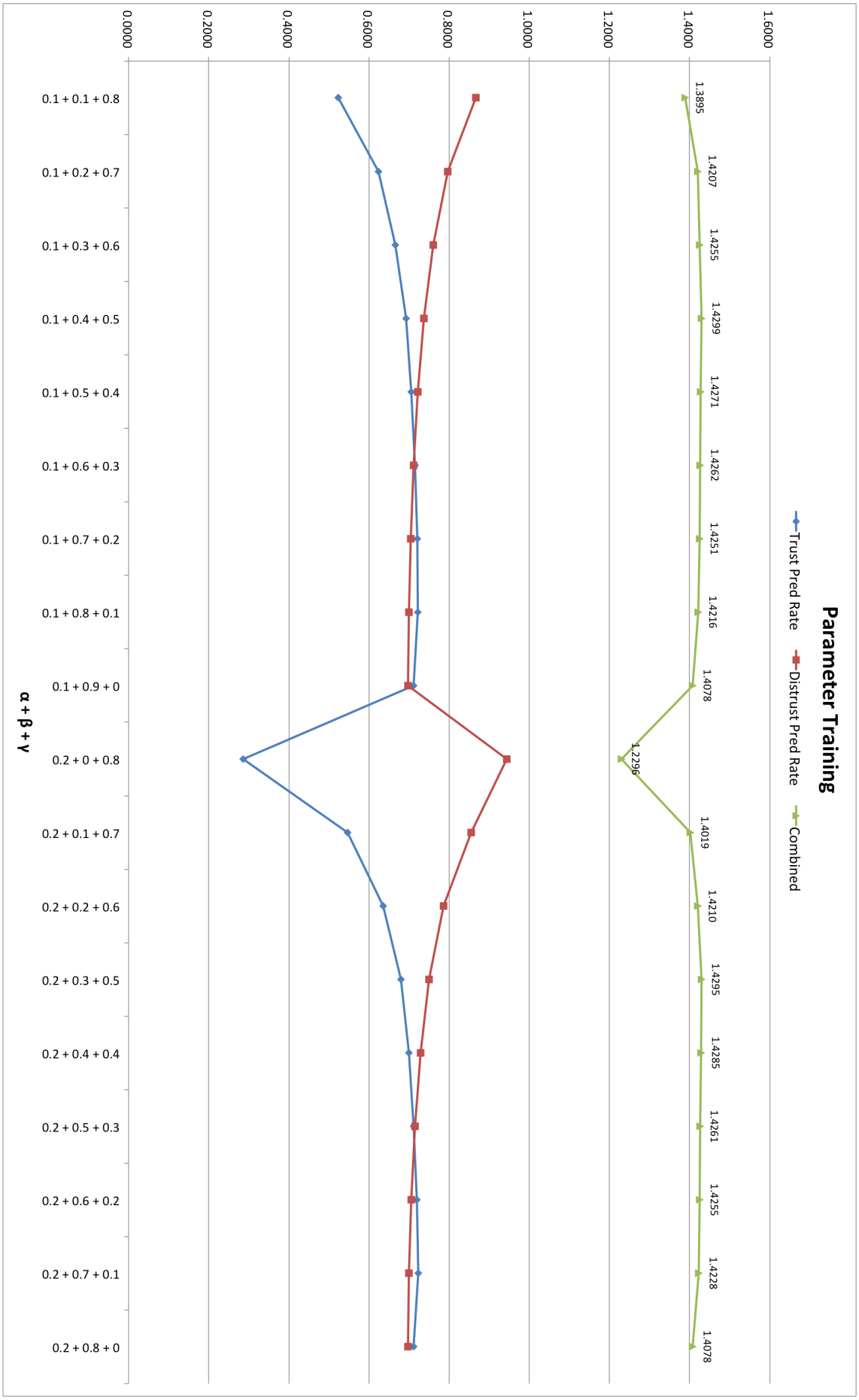


Figure 2.7: Training of Parameters for Trust Inference Formula

α	β	γ	σ
0.1	0.4	0.5	$\frac{1}{3}$

Table 2.1: Selected parameters

too many False Positives and therefore having a low Distrust Prediction Rate. Taking into consideration all three of these measurements resulted in the choice of the combination of values for the parameters that gave the best balance between the number of True Positives predicted by the formula as well as True Negatives.

Table 3.3.2 illustrates the selected parameters resulting from the validation and training phase described above. As we can see from figure 2.7, the chosen combination of parameter values has both a high trust prediction rate, and a higher distrust prediction rate. As well as this however, it is the top performing combination of values in terms of the combination result.

Here we remark that the $D(i)$ term in Eq. 2.3, weighted by the parameter γ, corresponding to the aspect of the formula related to the methodology of the popularity of shared items as indicated in section 2.3.1, has the most important contribution to the computation of trust in these experiments. This validates the proposed intuition based on the fact that the popularity of the shared items has an important impact on the trust between the users involved in the trust calculation.

2.4.4 Comparison to a naive approach

To the best of our collective knowledge, no other method exists for the inference of trust based solely on the structural information of a social bipartite graph. As a result, to compare the performance of the proposed formula to a similar methodology, I compare the trust inference method proposed to a naive trust inference method.

2.4.4.1 A naive approach to structural trust inference

This naive method also uses only the structural information of the graph, as well as the concept of shared items, and which is also based on the correlation of trust and user similarity.

Based on the underlying nature of many current collaborative filtering recommendation systems, as well as the correlation of similarity and trust proposed in [65, 21], the corresponding naive methodology states that:

- users with similar tastes will have rated the same items
- thus users with shared items should trust each other

Using this methodology, the method infers trust between user pairs if they have both rated at least one of the same items.

	Trust relation	Distrust relation
Our method	**65.00** %	**80.50** %
Naive method	35.62 %	80.05 %

Table 2.2: Prediction rate

To compare these two approaches, another experiment was carried out in a k-fold cross-validation style very similar to that performed for the training and validation phase above. For this experiment however, the *test set* of the Epinions dataset was used, and split into subsets of 1000 randomly chosen users. Both formulae were then applied to each of these subsets, and the trust prediction and distrust prediction measurements described above were computed for each. The final results for each were taken as the arithmetic mean of the results for all subsets performed by each approach.

Table 2.2 shows a comparison between the mean of the results of the trust inference method proposed in section 2.3.2 against those of the naive method performed on the subsets of 1000 randomly chosen users from the *test set* in a similar k-fold cross-validation style as described above.

From these results, we remark that the trust inference method proposed in this work outperforms the naive method for both the prediction of trust relations as well as the prediction of distrust relations. With mean values of 0.65, and 0.356, and standard deviations of 0.0829, and 0.06979 for the proposed method and the naive method respectively, a standard two tail, paired t-test of these results returns a P-value of less than 0.0001, showing the difference in results of the two methods to be statistically extremely significant to the 99% *confidence level*.

Given the potentially suboptimal nature of the parameters used for this comparison, due to the incomplete nature of the training phase at this

time as mentioned above, the likely discovery of a more optimal set of values for the parameters can only result in a further improvement to these results.

Indeed, one can conclude that the existence of shared items between prospective users alone is not necessarily a discriminant feature to infer a trust between them. On the other hand however, I remark that the two methods perform very similar in terms of the the rates of prediction of distrust relations, with a slight superiority of on the part of the proposed trust inference method. This can be explained by two possible facts;

a) Firstly, the fact that the existence of a distrust relation is not explicitly provided by the Epinions dataset.

b) Secondly, due to the shrinking diameter aspect of social networks, the data used will evolve over time and with it, new trust relations will be created.

This means that the results of our formula, especially the *FP*, can be improved if we take a more evolved version of the graph.

2.4.4.2 Secondary Comparison and Validation

As a secondary validation of our trust inference formula, we have analyzed the structural properties of the graph computed with our formula, to check its validity to the structural properties typical of social networks. To do this, we compared the properties of the computed graph to those of the real trust graph of the epinions dataset. For this comparison, we have plotted the following three structural properties.

- degree distribution (Figure: 2.8)
- the hop plot (Figure: 2.9)
- clustering coefficient of both graphs (Figure: 2.10)

All three of these structural properties were plotted for the same subset of 1000 users, considering only the edges in the computed graph which correspond to a trust assertion according to the chosen threshold.

As we can see, all three structural properties are very similar for both the real trust graph and the computed trust graph. Figure 2.8 shows the degree distribution plots of both the constructed trust graph 2.8(a), computed from the proposed trust inference formula, and the ground truth graph 2.8(b) from the real trust graph of the Epinions dataset. From these plots we can see that both distributions hold very similar properties to those typical of most on-line social networks as shown in [2, 31]. We can also see that the degree distribution of the constructed

graph 2.8(a) is more dense than that of the ground truth graph 2.8(b), meaning that there are more users who are highly connected and highly trusted in the constructed graph than in the ground trust graph. The explanation for this is quite straightforward. As mentioned above in section 2.4.4, the trust graph computed from our formula computes more trust edges between users than those that are currently existing in the real trust graph. This means that the computed trust graph can be seen in fact as a more evolved version of the real trust graph, containing trust edges that have not yet been asserted by the users in the real graph. These trust edges can thus potentially be used to recommend to users other user who they may trust.

Figure 2.9 shows both hop plot properties again of the computed trust graph 2.9(a) and the ground truth graph from the Epinions dataset 2.9(b). Once again, these plots show hop plot properties typical of social networks. As well as this, both of these plots are almost identical, without almost all of the users in the graph reachable through 4 hops of the graph.

Figure 2.10 shows the clustering coefficient plots for the computed trust graph 2.10(a) and the ground trust graph 2.10(b). Again both of these plots show properties similar to those typical of social networks. It is also worth noting the much increased clustering of the users in the computed trust graph 2.10(a) to that of the actual trust graph in the Epinions dataset 2.10(b). This can once again can be attributed to the fact that the trust inference formula computes trust edges that do not yet exist in the real trust graph. The clustering coefficient plot resulting from this formula is very typical of social and small-world networks [3], and adds worth weight to the idea that the computed trust graph can be seen as a more evolved version of the real trust graph.

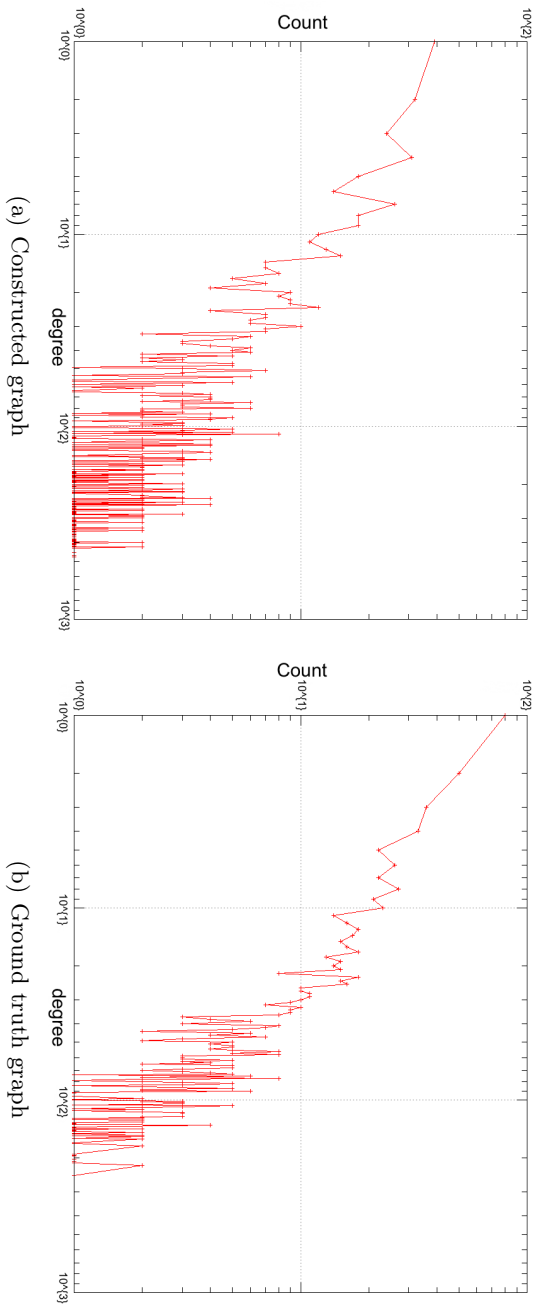
From this comparison, we can conclude that our trust inference formula contains the principle properties typical of social networks [2, 31, 3], and thus, as intended, our formula computes a new social trust graph connecting users by means of weighted trust edges.

2.5 Conclusion

Based on the correlation of trust and similarity, as empirically shown in [65], this chapter presents a methodology focusing on the popularity of shared items between users in a social graph for the inference of trust between users in a social network. Using this methodology, a formula is developed to infer trust information between users using only the structural information available from the bipartite graph typical of many

on-line social networks. From a real-life dataset, this formula computes a new trust graph, linking users together by means of weighted trust links. By comparing these computed trust edges to the corresponding edges in the real trust graph, and using measurements inspired from the domains of information retrieval and data mining [42, 54] the proposed formula is validated and its parameters, weighting each structural aspect of the formula, are trained following a k-fold cross-validation process.

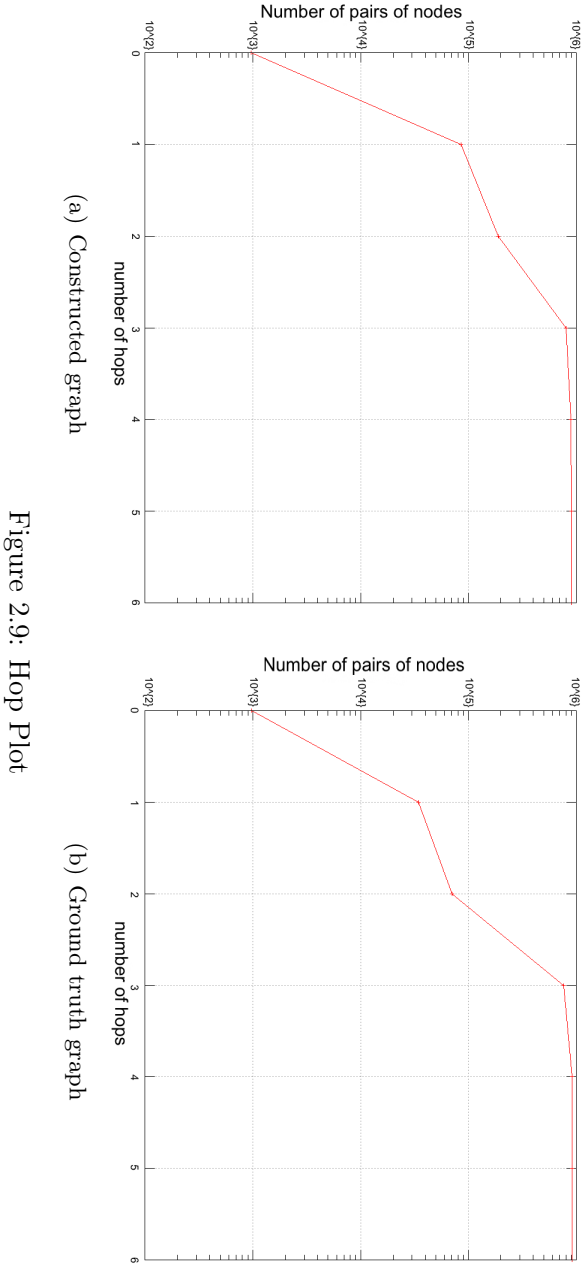
Following a similar cross-validation approach on a separate test set, the formula is then compared to a similar, naive structural trust inference method showing the proposed method to have a statistically extremely significant superiority. Finally, as a further validation of the proposed formula the degree distribution, hop plot and clustering coefficient structural properties of the trust graph resulting from this formula are plotted and compared to those of the real trust graph. These tests show the formula to have a strong trust prediction rate, as well as having structural properties typical of most social networks.



(a) Constructed graph

(b) Ground truth graph

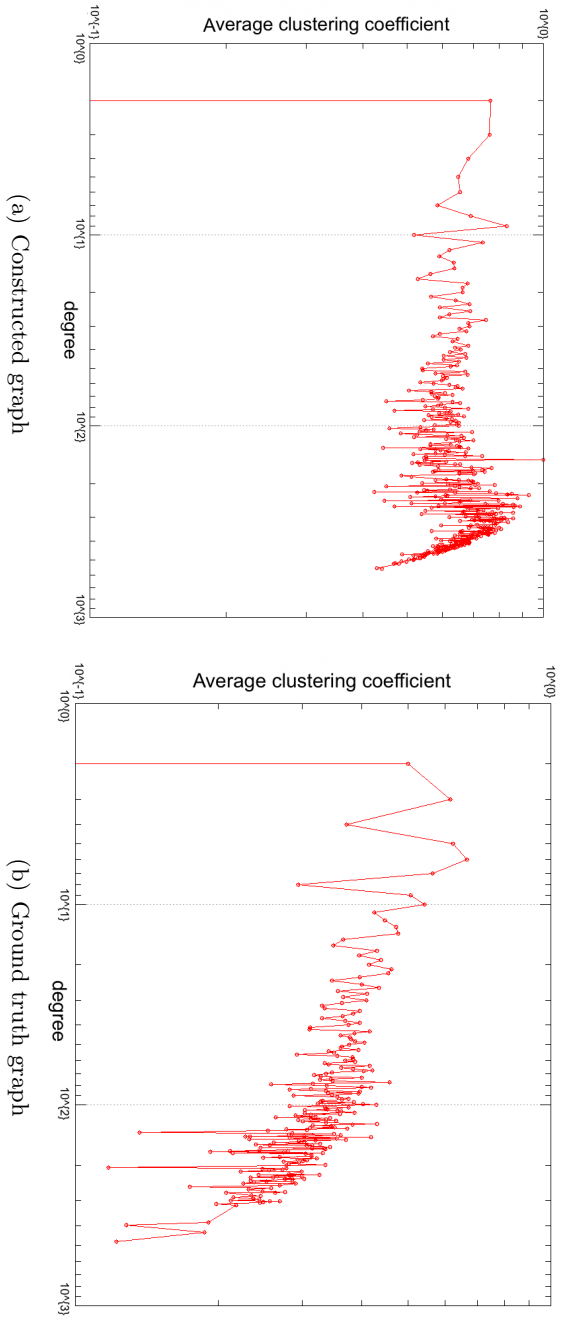
Figure 2.8: Degree Distribution



(a) Constructed graph

(b) Ground truth graph

Figure 2.9: Hop Plot



(a) Constructed graph

(b) Ground truth graph

Figure 2.10: Clustering Coefficient

CHAPTER 3

EMPIRICAL ANALYSIS

3.1 Introduction

This chapter presents a further validation of the trust inference formula developed in section 2.3.2, as well as a more complete empirical analysis of trust-based recommendation techniques. The trust inference formula is applied to the ratings information of three separate datasets, each of which have previously been used for the purpose of testing recommender systems. Each of these datasets differ in their respective connectivity properties, the distribution of ratings in the graph, as well as their overall use. The trust metrics computed between user pairs by the trust inference formula are subsequently used by a number of different techniques which have previously been designed and proposed for the incorporation of trust into recommendation algorithms, and the accuracy of each of these techniques is analysed and compared.

I begin by outlining the motivation for the second part of this work, before presenting the different recommendation techniques used in this analysis. These techniques include those proposed in previous work to incorporate trust into recommendation algorithms, as well as two of the standard collaborative filtering recommendation techniques commonly used in many social networks.

The three datasets chosen for this analysis are then presented, before the process undertaken to carry out the experiments for the analysis is outlined. Finally, detailed results performed by each algorithm on each of the datasets are presented, followed by an in-depth analysis and comparison of all recommendation techniques on all three datasets.

3.2 Recommendation Techniques

Each of the trust-enhanced recommendation algorithms described in section 1.2 above all use slightly different techniques to incorporate the trust information existing between users into the final recommendation process used to predict the utilities (e.g ratings) of unseen items to users. In this section I will present some of these techniques as well as some of the standard collaborative filtering techniques based on Pearson Correlation Coefficient (PCC) which will then be the subject of the analysis and comparison section later in this chapter.

The trust-enhanced techniques presented and analysed are mainly enhancements of the memory-based recommendation algorithms for item recommendation presented in section 1.1 and as discussed in [1]. As such, the main focus of this chapter is to perform an analysis and comparison of these trust-enhanced techniques, as well as comparing them to the standard memory-based recommendation techniques.

3.2.1 Standard Techniques

In this section we present two of the well-known collaborative filtering techniques widely used in many recommender systems: Resnick's [46] algorithm for collaborative filtering, and a standard weighted mean. To generate a recommendation for an item i to user u both of these algorithms use a similarity measure between u and a user v who has already rated i to weight the rating that v gave to i in the computation. This similarity measure is generally given by Pearson's Correlation Coefficient. As well as these two algorithms, we also present a very simple and naive method of item recommendation called a simple mean.

$$r(u,i) = \frac{1}{|R|} \sum_{v}^{v \in R} r_{v,i} \tag{3.1}$$

Equation 3.1 shows a simple mean of all of the ratings for item i, where N is the total number of raters for this item. This is a simple method that is still used in some recommender or popularity systems.

$$r(u,i) = \frac{\sum_{v}^{v \in R^+} w_{u,v} r_{v,i}}{\sum_{v}^{v \in R^+} w_{u,v}} \tag{3.2}$$

Equation 3.2 shows the standard *weighted mean* algorithm for rating prediction. Like the simple average formula above, this formula uses the

the average of the ratings of other users for item i but personalizes the prediction for a target user u by weighting the ratings according to how similar the rating user v is to the target user u. This weighting is usually calculated using Pearson's Correlation Coefficient.

$$r(u,i) = \bar{r_u} + \frac{\sum_v^{v \in R^+} w_{u,v}(r_{v,i} - \bar{r_v})}{\sum_v^{v \in R^+} w_{u,v}} \tag{3.3}$$

Equation 3.3 shows Resnick's formula [46] for correlation based Collaborative Filtering, where $w_{u,v}$ represents the weight of similarity between users u and v. This similarity weight is again usually calculated by using Pearson's Correlation Coefficient. A prediction for an item i for user u is based on the mean rating $\bar{r_u}$ of user u and the sum of the ratings of other users for item i. As described in [1], this algorithm is a form of *adjusted* weighted sum, which takes into account the fact that different users may use the ratings scale in different ways, by using the deviations of a user's rating for item i from the user's average rating overall. These ratings are thus weighted using the results of the similarity measure $w_{u,v}$ calculated using Pearson's Correlation Coefficient. As stated in [57], in practice, only the ratings of users with a positive correlation to u are considered in the prediction process. This is represented in 3.3 by the set R^+.

3.2.2 Trust-enhanced techniques

In this section we present some of the techniques used to incorporate trust into recommender systems. These algorithms are generally enhancements of the standard techniques seen above, and others have been developed to follow the natural idea that users are more inclined to appreciate recommendations based solely on the ratings of their trusted peers.

$$r(u,i) = \frac{\sum_v^{v \in R^T} t_{u,v} r_{v,i}}{\sum_v^{v \in R^T} t_{u,v}} \tag{3.4}$$

Equation 3.4 shows an enhanced version of the Weighted Mean formula shown in equation 3.2. This formula uses the trust value present between users u and v as a weight for the ratings of other users in place of the similarity measure. This is the technique used by the TidalTrust algorithm [19] presented above to incorporate their trust metrics into the recommendation process.

$$r(u,i) = \bar{r_u} + \frac{\sum_v^{v \in R^T} t_{u,v}(r_{v,i} - \bar{r_v})}{\sum_v^{v \in R^T} t_{u,v}} \tag{3.5}$$

Equation 3.5 shows a refined version of Resnick's formula which incorporates trust. In this method, the similarity weight $w_{u,v}$ attributed to ratings by user v for user u, calculated using Pearson's Correlation Coefficient is replaced with the value of trust $t_{u,v}$ present between u and v. This is technique used by the MoleTrust algorithm [33] presented above to incorporate their trust metrics into the recommendation process.

$$r(u,i) = \frac{1}{|R^T|} \sum_v^{v \in R^T} r_{v,i} \tag{3.6}$$

Equation 3.6 shows a trust filtering method, whereby the raters of item i are filtered according to their trust values, where only the raters who are trusted above a certain threshold are used in the computation of the predicted rating. Using these raters, we then take a simple mean of their ratings for item i. This method provides results according to the simple idea that "users are more likely to accept recommendations from their most trusted friends". We use this method in our comparison to evaluate the applicability of these claims.

$$r(u,i) = \bar{r_u} + \frac{\sum_v^{v \in R^{T+}} w_{u,v}(r_{v,i} - \bar{r_v})}{\sum_v^{v \in R^{T+}} w_{u,v}} \tag{3.7}$$

Equation 3.7 shows the Resnick collaborative filtering technique refined to include the trust value $t_{u,v}$ as a further filtering mechanism to choose only the item raters who are trusted above a certain threshold of trust. In this formula the weight $w_{u,v}$ is still used in the weighting process of the rating values, and is calculated using Pearson's Correlation Coefficient. This technique is that used by O'Donovan and Smith [51].

$$r(u,i) = \bar{r_u} + \frac{\sum_v^{v \in R^{+}} w_{u,v}(r_{v,i} - \bar{r_v})}{\sum_v^{v \in R^{+}} w_{u,v}} \tag{3.8}$$

$$w_{u,v} = \frac{sim_{u,v} + t_{u,v}}{2} \tag{3.9}$$

Equation 3.8 shows another refined use of the Resnick collaborative filtering technique, which combines both the similarity of users and the respective trust values between. In this formula, the weight $w_{u,v}$ assigned is in fact a combination of the similarity score $simu, v$ which is computed between users u and v by Pearson's Correlation Coefficient, and the trust score $t_{u,v}$ that exists between them. This technique is

again inspired by the first technique used by O'Donovan and Smith in [51]. However, the technique shown here differs slightly from that of [51] in the fact that here the combination of similarity and trust is made up using an arithmetic mean as shown in equation 3.9 as opposed to the harmonic mean of the two results as used by O'Donovan and Smith.

$$r(u,i) = \bar{r_u} + \frac{\sum_v^{v \in R^T} t_{u,v}(r_{v,i} - \bar{r_v})}{\sum_v^{v \in R^T} t_{u,v} + \sum_v^{v \in R^+ \setminus R^T} w_{u,v}} + \frac{\sum_v^{v \in R+ \setminus R^T} w_{u,v}(r_{v,i} - \bar{r_v})}{\sum_v^{v \in R^T} t_{u,v} + \sum_v^{v \in R+ \setminus R^T} wu, v} \tag{3.10}$$

Finally, equation 3.10 shows the *Ensemble Trust* collaborative filtering, as proposed by Victor et al [57]. In this work, the authors explain that this technique is designed to be able to take into account all possible ways to obtain a positive weight for a user who has rated an item. Using this rationale, trust relations are favored over similarity measures. This method aims to increase the percentage of predictions that the recommender system can make, which they call the coverage, a term which we follow in this work.

3.3 Experiments

3.3.1 Datasets

The purpose of this work is to provide an analysis and comparison of the recommendation techniques presented in section 3.2, on a number of different datasets. To this end, I have selected three publicly available datasets previously used for the study of recommendation systems. Each dataset differs in their intended purpose and properties, such as the connectivity and size of the graph, as well as the sparsity and distribution of ratings. These datasets thus provide the perfect basis for the analysis of the performance and behavior of the different techniques in different circumstances.

3.3.1.1 MovieLens

MovieLens is an on-line social network where users can rate and discover movies. The MovieLens dataset [46] is a widely used dataset for the study of recommender systems, and was publicly distributed by the

GroupLens Research at the University of Minnesota. [1] The dataset used is a commonly used version of the network and is available directly from the GroupLens website. It contains 6,040 users, 3,900 different movies, and 1,000,209 ratings collected from registered users as of the year 2000. Movie ratings are on a discrete scale from 1 to 5 stars, and this version of the dataset is well connected with each user in the set having at least 20 ratings encoded. The distribution of these ratings can be seen in Figure: 3.1(a).

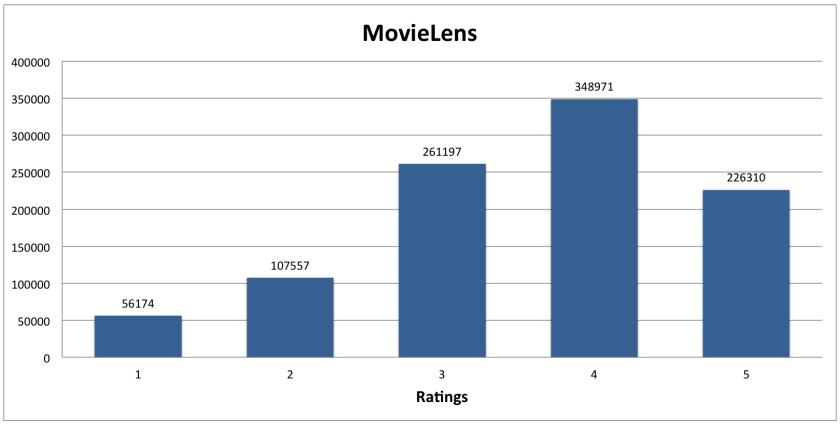
3.3.1.2 LibimSeTi

LibimSeTi [2] is an on-line dating service, where users are able to store personal profiles, as well as rate the profiles of other users. The dataset [3][10] used in this study contains 17,359,346 anonymous 4 ratings, on a discrete scale from 1 to 10, with 168,791 profiles made by 135,359 LibimSeTi users as of April 4, 2006. From this dataset, we used 20,000 randomly chosen profiles as will be explained. Figure: 3.1(c) shows the distribution of the ratings for this dataset. As can be seen, this dataset contains a high number of extreme ratings, with a large percentage of the ratings in the graph being either the lowest possible value of 1 (2,315,546 ratings), or the highest possible value 10 (3,540,342 ratings).

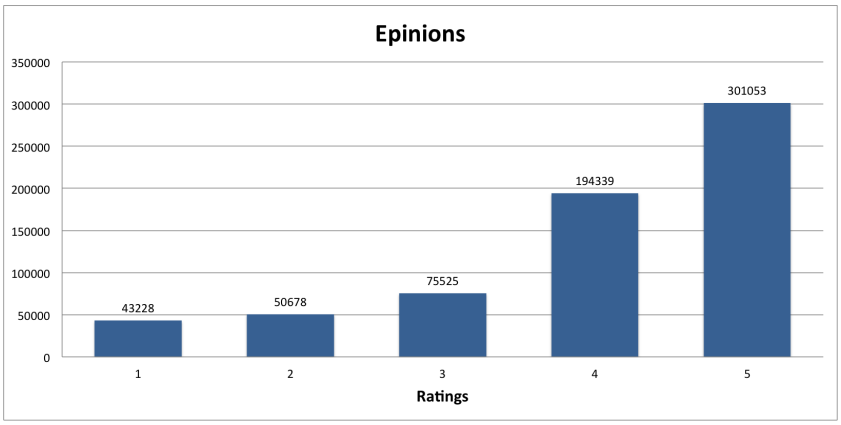
3.3.1.3 Epinions

The Epinions dataset used for these experiments is the same dataset presented and used in section 2.4.1. The availability of the explicit trust information in this dataset has made it very applicable and widely used dataset in the study of trust-enhanced recommendation systems. However, for the purposes of this comparison, we do not use this explicit trust information as we are interested in providing an equal comparison of trust in recommender algorithms across all datasets, and thus we only use the ratings information to automatically generate a web of trust, in the same manner as for the other two datasets. Figure 3.1(b) shows the distribution of the ratings in this dataset. As can be seen, 74% of the ratings in the graph are given as either the top value of 5 (45%) or just below it at 4 (29%).

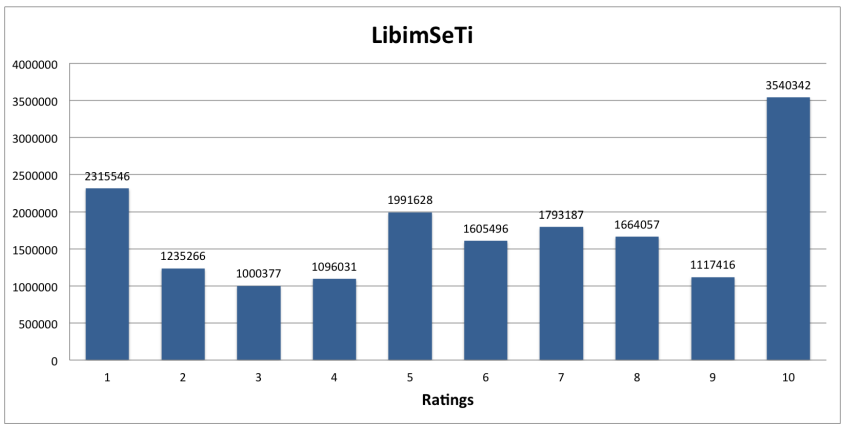
[1] http://www.grouplens.org/
[2] http://www.libimseti.cz/
[3] http://www.ksi.ms.mff.cuni.cz/ petricek/data/



(a) MovieLens Ratings Distribution



(b) Epinions Ratings Distribution



(c) LibimSeTi Ratings Distribution

Figure 3.1: Ratings Distributions

3.3.2 Choosing Parameter Values

For the purposes of this empirical analysis, I wanted to be able to get a clear comparison of the each recommendation technique, and how they perform on each dataset. For this, I decided to use the same values for the parameters of the trust inference formula: α, β, γ and σ and not to tailor the values of the parameters specifically to any of the datasets to find values would be most applicable to each specific dataset, but instead to use the same set of values that would return comparable results for all.

First of all, taking into consideration the connectivity properties of the MovieLens, as well as the LibimSeTi datasets, I decided to reduce the value of parameter σ in the formula, which controls the distribution of trust values in relation to the indegree of the items, as detailed in section 2.3.2. As the connectivity properties of these datasets was much greater than those of the Epinions dataset (e.g items in the MovieLens dataset have an indegree of at least 20 ratings), which was used for the experiments in the previous chapter, it was decided to reduce the value of σ to 0.2 as opposed to 0.33 as used previously so as not to overly punish the items in the well connected MovieLens dataset, but at the same time taking into consideration the sparseness and much lower connectivity properties of the Epinions dataset also used (52.82% of the population only expressed less than 5 ratings as shown in figure: 3.1).

As well as this, I decided that I would eliminate the usage of the parameter α used in the previous section which represents the probability that users trust each other based on any form of other external information in section 2.3.2. Therefore, I wished to concentrate only on the two structural features of the graph accounted for in the previously developed trust inference formula and to weight these features accordingly using only two parameters. The parameter β in this chapter, as before, is used to weight the first structural feature of the Relative Diversity of the users according to their two-hop neighbourhoods, and the parameter γ is used in this chapter to weight the second structural feature representing our methodology of the popularity of shared items as described in section 2.3.1.

The choice of these values for the parameters was based on the fact that having had the time to run more tests before commencing the experiments of this empirical analysis chapter, we were able to train the trust inference formula more and more using different combinations of values for the parameters than we initially had time for, and thus we were able to further train the formula. From these extended training tests using the Epinions dataset, as well as deciding to no longer use the α in the

α	β	γ	σ
0	0.4	0.6	0.2

Table 3.1: Selected parameters

computation of trust, we were able to see that these values for β and γ performed the best following the three criteria.

Figure 3.2 shows partial results for the secondary training phase of the formula without considering the parameter α. As before in section 2.4.3 in chapter 2, this training phase was performed using a k-fold cross validation approach using subsets of the training set of the Epinions dataset. In blue, again we can see the trust prediction rate performed by each of the combinations of values of the β and γ. In red, as before is the distrust prediction, and in green the combination of the two.

Table 3.3.2 shows the chosen values for the parameters α, β and σ used for the experiments and analysis in this chapter.

3.3.3 Performance Measures

To measure the performance of each of the recommendation algorithms for the purpose of comparison, we follow the common technique of hiding a certain number of ratings from the graph, and applying each algorithm in turn to predict the value of this rating. From these predictions, we measure the accuracy of each algorithm according to two commonly used accuracy measures in the field of recommender systems [26].

Firstly, we use the Mean Absolute Error (MAE) as shown in Equation 3.11, where N is the total number of ratings to predict, $P_{u,i}$ is the predicted value of the rating of item i by user u and $R_{u,i}$ is the real value of the rating of item i from user u. The MAE measures the average absolute deviation between the predicted rating $P_{u,i}$ of the algorithm and the user's true rating $R_{u,i}$.

$$MAE = \frac{\sum_{i=1}^{N} |P_{u,i} - R_{u,i}|}{N} \tag{3.11}$$

Secondly, we use the Root Mean Square Error (RMSE) shown in Equation 3.12. We use the RMSE as well as the MAE as it gives us a broader view of the performance of each algorithm. As discussed in [26, 37],

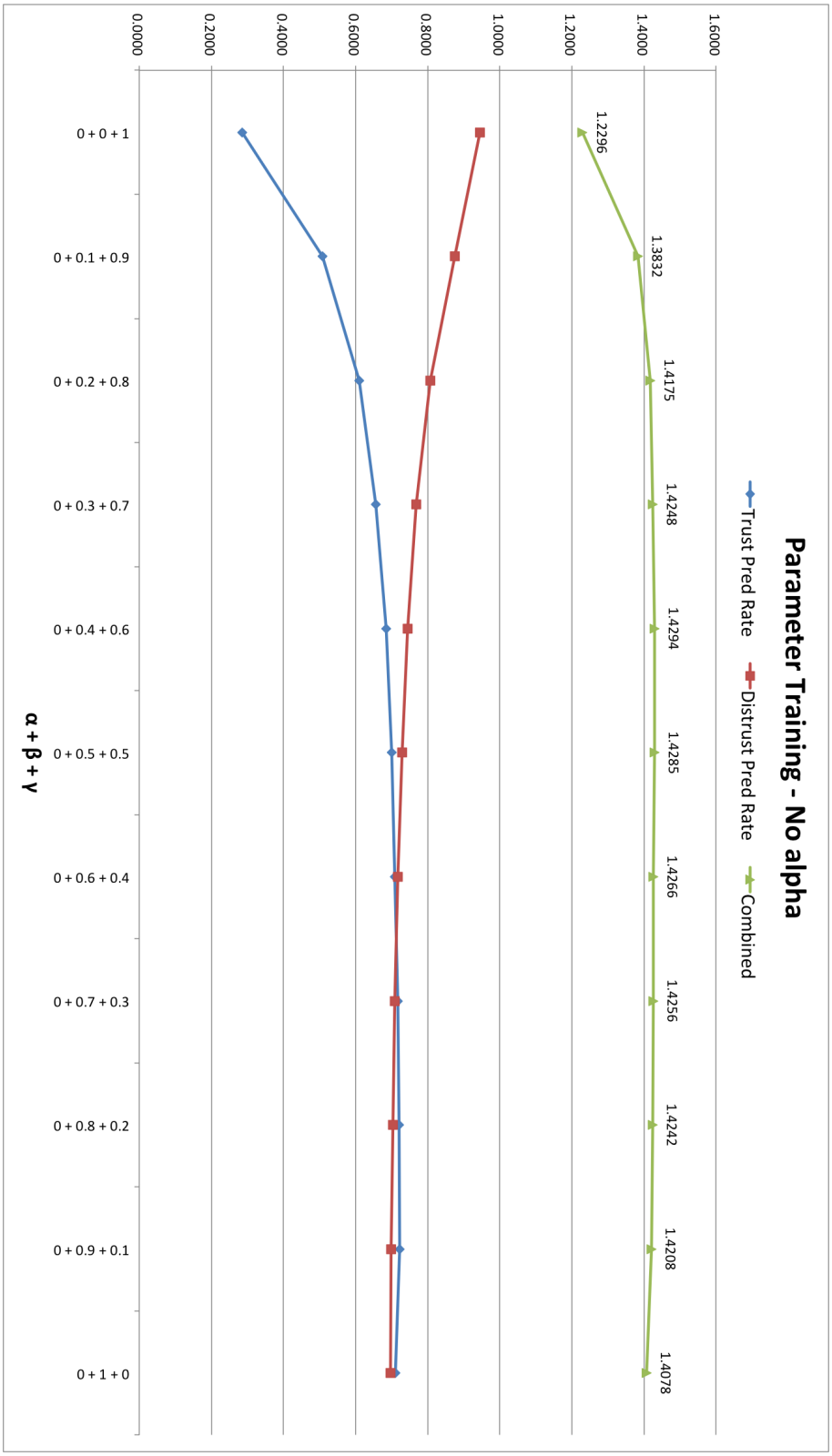


Figure 3.2: Further Parameter Training with No Alpha

while using only the MAE, each error in prediction is treated with equal value, whereas with the RMSE larger errors in prediction are punished, as the difference between the predicted rating $P_{u,i}$ from the real rating $R_{u,i}$ is squared. This means that the RMSE punishes predicted values that deviate further from the real rating value. As a result, lower RMSE values indicate that an algorithm is more accurate in the sense that its predicted values do not deviate very far from the true rating.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(R_{u,i} - P_{u,i})^2}{N}} \tag{3.12}$$

As well as these accuracy measures, we follow the analysis of performance as discussed in [26, 57, 59, 37], and measure the *coverage* of the algorithms. The coverage is simply the percentage of the hidden user to item ratings for which the algorithm was able to produce some sort of prediction. We compute this by taking the number of ratings for which the algorithm was able to make a prediction and divided it by the total number of hidden ratings that were to be predicted, as shown in Equation 3.13, where P is the total number of predictions that were made by the algorithm, and N is the total number of user to item ratings that were requested of the algorithm to predict. The coverage of predictions has been shown to be an important measure of performance for recommendation algorithms, where some techniques have been shown to be unable to produce predictions for a significant percentage of ratings. This can happen for example with Resnick's collaborative filtering formula (Eq: 3.3) when there are no users who have rated an item i who have a positive pearson correlation with the target user u.

$$Coverage = \frac{P}{N} \tag{3.13}$$

3.3.4 Validation Process

Before beginning the experiments, a sample subset of $20,000$ randomly chosen users was extracted from both the LibimSeTi and the Epinions dataset. For the MovieLens dataset, with just over $6,000$ users we used the entire dataset. In previous studies [19, 34, 57, 51], the performance of recommender systems is typically assessed using a *leave-one-out* method, which entails hiding each rating in the graph one by one, and thus predicting the value of this rating using the recommendation algorithm. This process is then repeated a certain number of times to build up a

measure of accuracy of the algorithm. However, in this work we wish to measure the performance of each algorithm when there are different percentages of ratings missing from the ratings graph and thus less information available. In order to do this, and to compare each of the different recommendation techniques on each of the data samples, we followed a standard *k-fold* cross validation process, which we will explain in this section. The following process described was repeated in an identical fashion for all three datasets.

To begin, the data samples were first split into randomly chosen subsets of $1,000$ users. For each subset, a certain percentage of the ratings was then randomly removed from the graph, which were then subsequently used as the ratings for which each recommendation algorithm would provide a prediction. Once these ratings had been removed, the trust inference formula was then applied to the remaining ratings subset, as detailed in section 2.3.2. Doing this thus generated a new trust graph computed by our formula, linking users of the subset together by means of weighted symmetric trust links as was the case in section 2.4.3. Following this, each recommendation algorithm described in 3.2 was then applied to the resulting subset to predict the ratings that had previously been randomly selected to be removed.

The trust-enhanced algorithms thus used the trust information contained in the computed trust graph in the process of computing their predictions. By comparing the resulting predicted ratings of each algorithm to the corresponding actual rating, it was then possible to compute the accuracy and coverage measures as detailed in section 3.3.3, and measure the performance of each recommendation algorithm for each subset.

The final performance results of each algorithm presented in Tables 3.3, 3.4, and 3.2 are thus the mean of the results computed for each subset. This process was then repeated five times for 10%, 20%, 30%, and 50% of the ratings randomly removed. As stated above, this procedure was repeated for each dataset.

3.4 Analysis & Discussion

In this section I present the results of the experiments performed on the selected datasets. Firstly, I give an overview of the results as a whole before going a little bit further in depth for the analysis of each dataset. Tables 3.2, 3.3, and 3.4 report the performance results for each algorithm performed on the MovieLens, Epinions, and LibimSeTi datasets respectively. Each table presents the performance results (MAE, RMSE and Coverage) achieved by the algorithms tested, performed four times with 10%, 20%, 30% and 50% of the total ratings hidden from the test set.

3.4.1 Overview

Overall, we can remark that the trust-based techniques, on average, outperform their standard counterparts for all three datasets. This follows the results of previous studies [20, 34, 57] and thus adds further weight to the idea that trust can enhance the performance and accuracy of recommendation algorithms. In terms of the accuracy, we also remark that the performance of each algorithm differs depending on the dataset on which it is used.

For example, if we consider the figures for 10% of the ratings hidden in the results for the Epinions dataset shown in Table 3.3, we can see that the algorithms using a weighted mean strategy: Trust-based WM (Eq: 3.4) and Pearson WM (Eq: 3.2), with MAE of 0.919, and 0.951 respectively, slightly outperform the techniques based on Resnick's formula: Trust CF (Eq. 3.5) with a MAE of with 0.927, and Pearson CF (Eq. 3.3) with a MAE of 0.966.

However, the opposite is true for the LibimSeTi dataset (Table 3.4) and the MovieLens dataset (Table 3.2) where the Resnick-based techniques significantly outperform those using a weighted mean. This is demonstrated by a superiority of 0.56 for the Trust CF algorithm over the Trust WM algorithm, and a superiority of 0.57 for the Pearson CF algorithm over the Pearson WM algorithm both in the MovieLens dataset for the results with 10% of the ratings hidden.

One possible cause of this swing in fortunes could be the different connectivity properties as well as the distribution of the item ratings of each of the datasets. As discussed by Massa et al in [34] using the same Epinions dataset, and as shown in figure 3.1(b), the vast majority of ratings (74%) in the Epinions dataset are given a weight of either 4 or 5 stars. Of these, 45% are given the highest rating of 5, meaning that there is very little variance in rating values. As the Resnick-based algorithms uses the variance of a user's rating for an item to the user's average rating as part of their computations, such lack in variance may indeed be hindering these algorithms.

On the other hand, as shown in figure 3.1(a), the MovieLens dataset is quite highly connected with all users having rated a minimum of 20 items each, and as shown in [15] these ratings follow a normal distribution with a significant degree of variance. This variance thus allows the Resnick-based algorithms to use more information and outperform the algorithms based on a weighted mean technique.

	10% hidden			20% hidden			30% hidden			50% hidden		
Algorithm	MAE	RMSE	Cov %	MAE	RMSE	Cov %	MAE	RMSE	Cov %	MAE	RMSE	Cov %
Pearson CF	0.707	0.99	98.74	0.714	0.9988	98.08	0.721	1.008	97.50	0.741	1.030	96.08
Pearson WM	0.764	1.041	98.74	0.769	1.049	98.08	0.775	1.055	97.50	0.792	1.078	96.08
Simple Mean	0.755	1.024	100	0.756	1.027	100	0.756	1.028	100	0.762	1.034	100
Trust CF	0.700	0.980	99.96	0.702	0.982	99.96	0.704	0.985	99.95	0.712	0.993	99.93
Trust Filtered Mean	0.755	1.025	99.96	0.756	1.027	99.96	0.756	1.029	99.95	0.762	1.035	99.93
Trust Filtered CF	0.78	0.993	98.10	0.790	1.004	97.25	0.795	1.011	96.48	0.813	1.033	94.62
Combined CF	0.707	0.989	98.10	0.714	0.997	97.25	0.719	1.003	96.48	0.734	1.022	94.62
Trust WM	0.756	1.026	99.96	0.757	1.029	99.96	0.757	1.030	99.95	0.764	1.037	99.93
Ensemble Trust	0.700	0.979	99.99	0.702	0.982	99.99	0.704	0.984	99.99	0.712	0.992	99.97

Table 3.2: Results for MovieLens

3.4.1.1 Trust Filtered Mean

In terms of the Trust Filtered Mean algorithm (Eq: 3.6), which takes a simple average of all of the ratings given by only trusted users, and thus filtering out all other ratings, the results are not as straightforward. Indeed, for the results performed on the MovieLens dataset, the Trust Filtering Mean algorithm performs identically to the Simple Mean algorithm (Eq: 3.1) in terms of accuracy, with a slightly worse RMSE performance for some of the tests.

A possible explanation for such similarity in scores could be the high connectivity of the items in the MovieLens dataset, meaning that even when the trust filtered algorithm filters out all of the users who have rated an item there is perhaps still a relatively high number of raters left, and an average of their rating values might still give a result very close to the average rating as computed by the simple mean algorithm. This doesn't exactly follow the original motivation laid out for this technique that users would be more likely to appreciate recommendations from those whom they trust. Again, for the Epinions dataset, the Trust Filtered Mean actually performs worse than the simple mean, performing quite similar accuracy results with only 10% of the ratings hidden but falling to an inferiority of 0.1 for the test performed with 50% of the ratings hidden. This may possibly be attributed to both the sparsity of the Epinions dataset, meaning that the algorithm has less information to be able to infer trust between users, as well as this as a result of the majority of the ratings being close to the top of the ratings scale, it seems that filtering only the trusted information can still not outperform a simple mean. Again, even though it performs the second best out of all of the other algorithms it still doesn't quite fit our methodology.

However, for the LibimSeTi dataset, we can see that the trust filtered mean algorithm consistently outperforms the simple mean in terms of both the MAE and the RMSE performance measures, and increases this superiority as the number of ratings that are hidden increases, with a superiority of up to 0.37 for the MAE with 50% of the ratings hidden. As will be discussed in further detail, this increased performance may be attributed to the distribution of the ratings within the LibimSeTi dataset as discussed in section 3.1. As can be seen in figure 3.1(c), this dataset contains a large percentage of extreme ratings of either the lowest rating of 1 or the highest possible rating of 10.

As will be discussed, previous studies [20, 35] have shown that with such difference in opinion between the ratings of items (i.e controversial items), trust can be used to provide a more personalized recommendation to the individual users. From these results, we can see that this seems to hold true for this dataset, and indeed seems to follow the motivation of the

trust filtered mean algorithm, whereby only using the ratings information provided by trusted users provides better accuracy results when a user's rating for an item differs significantly from the average rating of this item.

3.4.1.2 Combined CF

For the performance of the Combined CF algorithm (Eq: 3.8), again we see some very interesting results showing how the introduction of trust into recommendation algorithms can improve the accuracy in relation to standard similarity techniques. By comparing this algorithm to the two other algorithms most similar to it, that of the Pearson CF (Eq: 3.3) which uses the only the Pearson Correlation Coefficient (PCC), and that of the Trust CF algorithm (Eq: 3.5), we can see that the combination of trust with the PCC similarity measure improves the accuracy of recommendations in relation to the pure PCC based algorithm: Pearson CF, however, this improvement still falls short of the MAE performance when only trust is used, as is the case with the Trust CF algorithm. This is demonstrated by the MAE accuracy performance of the Combination CF of 1.439 with 10% of the ratings hidden for the LibimSeTi dataset, in comparison to that of 1.444 of the Pearson CF and 1.410 of the Trust CF algorithm for this same test set. This comparison is also reflected in the performance the RMSE accuracy measure of the algorithms.

However, for the MovieLens dataset, we see that the Combination CF algorithm performs exactly the same MAE result as the Pearson CF algorithm for both 10% (MAE of 0.707 and 20% (MAE of 0.714) of the ratings hidden. But, by examining the performance results of the RMSE (Eq: 3.12) accuracy measure, we see a different story. As discussed in section 3.3.3, while the MAE gives the mean of the absolute deviation of the prediction values from the actual value, the RMSE accuracy measure squares this deviation and thus the more a predicted value deviates from the real value the more this prediction is punished, and thus lower RMSE measures show an algorithm to be more accurate. From these results therefore, although the Combined CF algorithm performs the same MAE accuracy as the Pearson CF for 10% and 20% of the ratings missing in the MovieLens dataset, we can see that in fact it performs better and gives predicted values that are closer to those of the Pearson CF formula.

Such results for the accuracy performance underline the notion laid out by previous works that the incorporation of trust into the recommendation process, even in tandem with previous similarity measures can improve the overall accuracy of the recommendation algorithms using it.

3.4.1.3 Trust Filtered Collaborative Filtering

One surprise result in terms of accuracy performance was that of the Trust Filtered collaborative filtering technique (equation 3.7) proposed by O'Donovan and Smith in [51]. In comparison to the other algorithms, this technique performed quite poorly in the experiments, and in fact it was consistently the worst performer in terms of MAE. This result goes against previous intuition that users are more inclined to appreciate more recommendations from their trusted neighbours, as well as the improvement of results as shown in [51].

However, if we take a closer look at the RMSE (Eq: 3.12) performance results for this technique, we can see that it does not perform as badly in the more connected datasets: MovieLens and LibimSeTi in comparison to the standard techniques. One explanation for this performance could be related to the way in which the technique filters users. From the users with a positive similarity measure calculated using the PCC, this technique filters out the users who are also not trusted by the user and therefore does not include them in the recommendation process. This constrains the amount of users that can be used in the recommendation process and may indeed eliminate users from the recommendation process who are trusted by the user, but who are not positivity correlated with the user according to PCC. This consequently constrains the amount of information that the recommendation algorithm can use to make a prediction, thus leading to a lower number of predictions that are correct. However, from the performance based on the RMSE, we can deduce that although the technique gives the correct prediction less often than the other techniques, the error in prediction is never too high, as the RMSE punishes large errors in prediction.

In terms of coverage, on the whole we can also see that the trust-enhanced techniques vastly improve the percentage of prediction coverage of each of the recommender algorithms, particularly when the dataset itself is very sparse, as is the case with the Epinions dataset. But as well as this, in the case of the MovieLens and LibimSeTi datasets which are more connected, we notice that when the percentage of ratings that are hidden is increased, the coverage of the standard collaborative filtering techniques using Pearson's Correlation Coefficient (PCC) drops considerably, whereas the rate of coverage of the trust-enhanced techniques stays reasonably stable, and drops at a much slower rate than the standard techniques. Taking an example from the LibimSeTi dataset, when 50% of the ratings are removed, the trust-enhanced techniques outperform the standard techniques in terms of coverage by up to 20%, with the trust-based algorithms having a coverage rate of 96.69% while the standard techniques having a coverage rate of only 76.32%. A possible explana-

	10% hidden			20% hidden			30% hidden			50% hidden		
Algorithm	MAE	RMSE	Cov %	MAE	RMSE	Cov %	MAE	RMSE	Cov %	MAE	RMSE	Cov %
Pearson CF	0.966	1.356	49.17	0.991	1.391	41.74	1.012	1.413	34.69	1.048	1.462	20.94
Pearson WM	0.951	1.367	49.17	0.971	1.399	41.74	0.988	1.420	34.69	1.015	1.465	20.94
Simple Mean	0.898	1.29	96.99	0.904	1.303	93.88	0.923	1.328	89.73	0.935	1.349	79.87
Trust CF	0.927	1.308	83.37	0.946	1.329	79.28	0.965	1.356	73.76	1.004	1.403	59.86
Trust Filtered Mean	0.898	1.297	83.37	0.911	1.308	79.28	0.924	1.331	73.76	0.945	1.361	59.86
Trust Filtered CF	1.029	1.360	49.17	1.0503	1.394	41.17	1.074	1.4215	34.69	1.110	1.470	20.94
Combined CF	0.962	1.35	49.17	0.989	1.387	41.17	1.008	1.409	34.69	1.048	1.461	20.94
Trust WM	0.919	1.323	83.37	0.941	1.345	79.28	0.952	1.368	73.76	0.981	1.41	59.86
Ensemble Trust	0.927	1.308	83.37	0.946	1.33	79.28	0.965	1.356	0.737	1.004	1.403	59.86

Table 3.3: Results for Epinions

tion for this increase in accuracy in this case could be the potential of our trust inference formula to find trusted neighbours beyond those whom a user shares an item. As discussed in section 2.3.2, our formula uses two structural features to infer trust information, and does not only concentrate on shared items. The use of the *two-hop neighbourhood* in this formula allows our formula to infer trust between users who have not yet rated one of the same items. Bearing in mind that Pearson's Correlation Coefficient (PCC) is primarily based on shared items, this means that the set of trusted users for a user is more likely to be larger than that of the set of users for which PCC returns a positive result, especially when 50% of the ratings have been deleted.

We can see that this difference in coverage remains the same for the Epinions dataset with an improvement of almost 40% when 50% of the ratings are removed, but it does not hold for the MovieLens dataset, where the trust-based techniques only deliver around 0.8% improvement. This suggests that as both the LibimSeTi dataset and more particularly the Epinions datasets are more sparse than the highly connected MovieLens dataset, the likelihood is higher that the increased removal of ratings will remove all shared items remaining between users. When considering the fact, as discussed in [34] that more than 50% of the users in the Epinions dataset have provided less than 5 ratings, this likelihood of separation of users becomes more evident. However, with each user having a minimum of 20 rated items in the MovieLens dataset this is less likely to be the case, as users have rated much more items and thus are thus more likely to have shared items with other users in the network, even after a high number of ratings removed.

Apart from the Simple Mean algorithm (Eq. 3.1) which has has the best coverage performance as it does not filter out any ratings information but take a simple average of all ratings given to each item, we remark that Ensemble Trust algorithm (Eq. 3.10) proposed by Victor et al in [57] leads to the best performance in terms of rating prediction coverage, providing almost optimal coverage in every dataset regardless of the number of ratings that are hidden. This again follows the observations made by the authors themselves. As stated in their work [57], the motivation for this algorithm was to account all possible ways to obtain a positive weight for a user to include them in the recommendation process, while favoring a trust weight over a similarity weight computed by PCC. It is worth noting here also, that again this benefit in coverage does not come at the expense of accuracy, with the Ensemble Trust algorithm performing equally well in terms of accuracy as the Trust-enhanced collaborative filtering (Eq. 3.5, indeed in the results for the MovieLens dataset in table 3.2 we see that Ensemble Trust returns a slightly better MAE.

	10% hidden			20% hidden			30% hidden			50% hidden		
Algorithm	MAE	RMSE	Cov %	MAE	RMSE	Cov %	MAE	RMSE	Cov %	MAE	RMSE	Cov %
Pearson CF	1.444	2.148	89.43	1.453	2.167	85.85	1.461	2.176	83.05	1.472	2.196	76.32
Pearson WM	1.452	2.275	89.43	1.454	2.298	85.85	1.455	2.304	83.05	1.453	2.327	76.32
Simple Mean	1.459	2.223	100	1.464	2.247	100	1.475	2.266	100	1.491	2.308	100
Trust CF	1.410	2.077	99.02	1.418	2.094	98.73	1.433	2.116	98.33	1.456	2.153	96.69
Trust Filtered Mean	1.437	2.206	99.02	1.438	2.224	98.73	1.447	2.242	98.33	1.454	2.275	96.69
Trust Filtered CF	1.488	2.149	89.21	1.498	2.170	84.9	1.504	2.178	81.85	1.516	2.199	72.72
Combined CF	1.439	2.137	89.21	1.449	2.157	84.9	1.456	2.168	81.85	1.469	2.189	72.72
Trust WM	1.43	2.202	99.02	1.432	2.220	98.73	1.441	2.239	98.33	1.454	2.279	96.69
Ensemble Trust	1.410	2.077	99.02	1.418	2.095	98.73	1.433	2.116	98.33	1.456	2.153	96.69

Table 3.4: Results for LibimSeTi

3.4.2 MovieLens Results

Table 3.2 presents the results of each algorithm performed on the MovieLens dataset. Further to that discussed in section 3.4.1, on close inspection of the results, we see that the standard formulae perform quite well with the Pearson CF (Eq. 3.3) returning a MAE of 0.707 with 10% of the ratings hidden. On the other hand, we also notice that the Trust CF algorithm (Eq. 3.5) performs slightly but not significantly better with a MAE of 0.70. This is more than likely due to the high connectivity of the dataset, with many ratings available for the PCC similarity measure to use.

However, we notice that as the percent of ratings that are removed from the graph increases, the accuracy measures returned by the trust-based formulae do not drop as quickly those of the standard formulae. Again this may be attributed to the fact that our trust inference formula is able to connect users to a wider range of users, and even though it does not use the ratings information like the PCC measure does, it seems that it is still able to gather enough information to perform well, and indeed slightly better than the standard formulae.

We also remark here that the Ensemble Trust algorithm (Eq. 3.10) actually performs the best for this dataset, slightly outperforming the TrustCF technique for both MAE and RMSE accuracy measures. The increased accuracy performance of the RMSE measure shows, as stated in section 3.3.3, that even though Ensemble Trust maximizes the coverage of rating predictions as much as possible it doesn't seem to compromise the accuracy of the predictions it returns and also doesn't seem to suffer from the previously proposed coverage-accuracy trade-off. As discussed in section 3.4.1, this can be attributed to the fact that the trust inference formula developed in this work, unlike previously proposed methods to infer trust, does not use any form of trust propagation between users along a trust path but it infers trust directly between pairs of users. These results show that by using both the PCC information and the trust information Ensemble Trust is able to maximize coverage as well as accuracy.

3.4.3 Epinions Results

Table 3.3 shows the results performed by each of the recommendation techniques on the subset of the Epinions dataset. First of all, from these results we remark that, counter-intuitively, the naive Simple Mean algorithm (Eq 3.1) actually outperforms all of the other algorithms. Of course, this is not the case for both MovieLens and LibimSeTi datasets

and certainly goes against intuition. However, these results are not completely a surprise and are similar to those observed in [34] for the same dataset. The main explanation for this result can be attributed to the fact introduced in section 3.4.1, that the vast majority of ratings (74%) in this Epinions dataset have a value of either 4 or 5, with (45%) of these having the highest possible rating of 5. With such little variance in the rating values, the simple unweighted mean of all the ratings of users will usually return values close to the real rating value. This is also reflected in the performance of the Trust Filtering Algorithm (equation 3.6), which performs equally well as the Simple Mean. This algorithm likewise uses simply a non-weighted average of the ratings for an item, but this time only from trusted users. But with such little variance between ratings we noticed that filtering through trusted users does not improve on the simple mean, and in fact in terms of coverage performs worse.

However, if we concentrate only on the performance of the trust-based algorithms in respect to the performance of the standard algorithms, we can see that the trust-enhanced algorithms again outperform their standard similarity based counterparts in terms of accuracy, as is the case for the other datasets.

In terms of coverage, the percentage of rating predictions the trust-based algorithms were able to make was almost twice as much as those that are based on the PCC similarity measure. Again, this may be attributed to the sparsity of this Epinions dataset. As discussed, the calculation of the PCC measure uses the deviation of ratings for shared items from the average rating of a user, and with the sparsity and low level of rating distribution of this dataset as discussed in section 3.4.1, it appears that the algorithm finds it difficult to compute positive similarity measures between a sufficient number of users. On the other hand, as discussed in section 3.4.1, the use of the *two-hop neighbourhood* structural aspect of the proposed trust inference formula (Eq. 2.3) allows users to connect to a larger range of other users in the graph than would be possible if the existence of shared items is only taken into account. By using this two-hop neighbourhood, the set of trusted users who have rated an item (for which a rating value is to be predicted) is more likely to be larger than the set of users that have a positive similarity measure from the PCC formula (as the PCC only uses information from shared items), thus giving much greater coverage to the trust enhanced techniques. The results from this Epinions dataset show the advantage of this feature of the proposed formula in very sparse datasets.

As well as this, as was the case for the MovieLens dataset, we remark that trust-enhanced techniques do not seem to suffer from the so-called "accuracy-coverage trade-off" described in [57], where a rise in coverage

is usually at the expense of accuracy, as previously experienced by some trust-enhanced measures. Again, as stated in the previous analysis of the MovieLens dataset, one possible explanation for this may be that previous comparisons have used trust propagation techniques [24] to increase the coverage of their trust enhanced formulas, and have experienced the issue whereby shorter propagation paths deliver more accurate results, and thus increasing the number of hops of propagation delivers less accurate trust estimations, and thus negatively impacting the accuracy of subsequent rating predictions. The method of inference proposed in this work however, does not seem to suffer from this drop in accuracy as it does not use any sort of trust propagation between users, but trust is inferred directly between users and consequently our resulting trust metrics seem to avoid the accuracy trade off experienced as a result of trust propagation.

3.4.4 LibimSeTi Results

Table 3.4 presents the results for each of the recommendation techniques performed on the LibimSeTi dataset.

In comparison to the improvement in the recommendation accuracy shown in the MovieLens dataset, on a close inspection of the results for the LibimSeTi dataset, we can a see particularly larger scale of improvement in the accuracy of predictions when trust is incorporated into the recommendation process. This is demonstrated by a superiority of 0.34 for performance of the the Trust CF algorithm in relation to the MAE performance for 10% of the ratings hidden over its standard counterpart Pearson CF, with both having a MAE of 1.410 and 1.444 respectively. Whereas the superiority between these two algorithms for the same test on the MovieLens dataset was only 0.07.

One possible explanation for this increased superiority over different datasets may be the increased number of extreme ratings present in the LibimSeTi dataset, As discussed in [15] and shown in figure: 3.1(c), this LibimSeTi dataset contains a large proportion of extreme ratings, with a large percentage of the ratings in the dataset having either the lowest possible value, 1, or the highest possible value 10. In many of the previous studies that have incorporated trust into the recommendation process [20, 35], it has been shown that difference in opinion between the users in a dataset is where the trust-enhanced techniques perform best in relation to the standard techniques, where a user's rating for a particular item differs significantly from the average rating for this item. Thus considering these extreme opinions present in the LibimSeTi, the use of trust in the recommendation techniques, as previous studies have shown,

seems to provide more personalization to the predicted recommendations which seems to reflect these differences opinions.

Further to what we have discussed in section 3.4.1, we notice that the Ensemble Trust (Eq. 3.10) and the Trust CF formulae are almost identical throughout the results. One possible explanation for this is that the set of users who have a positive pearson correlation (PCC) to the target user is similar to the set of users who are trusted by the user.

As for the rest of the results, we see common behavior to that of the datasets above, as the percentage of ratings that are hidden increases, the accuracy and the coverage of the algorithms drop, but the trust formulas do not drop as quickly, particularly in relation to coverage, where the coverage of the trust based algorithms remains reasonably optimal.

3.5 Conclusion

This chapter presents an empirical analysis of a number of the different techniques previously proposed to incorporate trust information into recommender systems, as well as standard recommendation algorithms commonly used in recommender systems.

Using the trust inference formula developed in the previous part of this thesis (section: 2.3.2) to infer trust information between users using only the structural information of the social graph, these techniques are applied to three different datasets without the need for explicit trust assertions. Each recommendation technique is then performed on each dataset to predict the values of ratings with 10%, 20%, 30% and 50% of the ratings hidden from each graph.

From these experiments, the accuracy of each algorithm is measured using the Mean Absolute Error (MAE), and the Root Mean Square Error (RMSE) measurements, and the prediction coverage of each algorithm is also measured. These measures are thus used to analyse the performance of each algorithm on each of the three different datasets with increasing numbers of ratings removed. From these results, we remark that the more ratings that are hidden in the graph, the accuracy of the standard algorithms drop at a quicker rate than those of the trust-based algorithms. We also remark that the prediction coverage of trust-based techniques remains relatively high even with 50% of the original ratings from the graph.

It is clear from these results that the incorporation of trust into recommendation algorithms can increase both the accuracy and the coverage of personalized recommendations. However, it is important to note the affect on each algorithm of the connectivity of the network in question.

Particularly in the case of a sparse dataset, the use of trust can significantly improve both the coverage and the accuracy of recommendations. On the other hand, when used in a highly connected network where the ratings of items are reasonably distributed, such as MovieLens the improvement is not distinct, with trust showing signs of improvement but not very significant improvement. However, with a well connected network but where the ratings are more extreme and show disagreement between users such as the LibimSeTi dataset, we can see that trust can distinctly improve accuracy and personalization of recommendations.

CONCLUSION

Split into two main parts, this thesis presents a further investigation and analysis of the domain of trust-based recommender systems.

Using a methodology inspired from real observations, the first part of this work proposes a new method for the automatic inference of trust information between users in bipartite social graph using only the structural information present in the graph. Through a set of experiments performed on a real on-line social network, the proposed method shows a high degree of accuracy for the prediction of true trust assertions between users, performing statistically significant results in comparison to a similar approach, showing that the existence of highly rated shared items between two users in a social network does not present a significant discriminant feature for the prediction of trust. The resulting trust graph generated from the proposed trust inference method also shows structural properties typical of on-line social network, further validating the proposed method.

The second part of this master thesis presents an empirical analysis of a number of different techniques designed to incorporate trust information into the process of content recommendation. Due to the unavailability of explicit trust information in the majority of on-line social networks, previous studies presenting a similar empirical analysis have been restricted to just a single dataset. By using the trust inference method developed as part of the first part of this thesis, we were able to perform these techniques on three separate datasets.

The analysis performed shows that the incorporation of trust into the recommendation process can indeed improve the accuracy of rating predictions, as well as vastly improve the prediction coverage of standard recommendation algorithms, as suspected. Notably, we can see that

trust is most effective in situations of extreme rating, where a user's rating from an item may significantly differ from the average rating, again as indicated in previous studies. Further to this, the experiments in this work also show a new interesting behaviour of trust-enhanced algorithms, that even as the number of ratings that are hidden from the graph increases, the accuracy and coverage of the trust-based recommendation techniques does not drop as fast as those of the standard techniques, underlining the advantage of incorporating trust into the recommendation process. However, most importantly, from this analysis we remark that the performance of each technique highly depends on the properties of the dataset on which it is applied. Such properties include the distribution of the ratings in the graph, the overall sparsity of the graph, as well as the amount of extreme ratings contained in the graph.

From this work, we can conclude that the method proposed in the first part of this thesis constitutes a re-usable and generic trust inference formula, capable of being applied to many different and varying social graphs containing a bipartite structure. This work also adds further weight to the benefits of using trust in the recommendation process, by showing its advantages over standard techniques on multiple and varying datasets.

3.6 Future Work

The work performed in this master thesis represents just a small step into this domain. The trust formula developed in the first part of this work still has much room for much improvement. An interesting direction for future development and improvement of this formula could be the consideration of content information of the vertices and edges. One example of this would be the incorporation of the rating value given by a user to a shared item. Presently, the formula only considers the structure of the graph, and infers trust between users using the existence of shared items. However, if one user has given a high rating to a shared item, and the other has given a low rating, then this will of course affect the trust inferred between them. The current form of the formula does not take this difference in ratings into account.

As well as this, another interesting possible direction which was briefly touched on in this work, but not included in the final version of the formula presented here, would be to take into consideration how users have previously interacted with the graph. For instance, a rating of 3 out of 5 stars for one user may be an average score for one user, but for a more conservative user, this might be above average, and thus indicating a preference for this item. As well as this, if one user has

rated many items and another user has only rated a few this may also indicate a distinguishing character trait which may be interesting to take into consideration. If one user likes lots of things, and another only like a few things, how much can we say about the similarity between them?

In addition to the possible enhancements of the trust inference formula, this work can be extended by applying it to a much larger scale. In a computational context, a future step would be to provide a large scale implementation of the method on top of large scale graph processing engines such as Apache Giraph.

BIBLIOGRAPHY

[1] Gedimindas. Adomavicius and Alexander. Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. In *Knowledge and Data Engineering*, volume 17 of *IEEE Transactions*, pages 734–749, 2005.

[2] Leman Akoglu and Christos Faloutsos. Rtg: A recursive realistic graph generator using random typing. In Buntine et al. [12], pages 13–28.

[3] L. A. N. Amaral, A. Scala, M. Barthélémy, and H. E. Stanley. Classes of small-world networks. *Proceedings of the National Academy of Sciences*, 97(21):11149–11152, October 2000.

[4] Ricardo A. Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[5] Marko Balabanović and Yoav Shoham. Fab: content-based, collaborative recommendation. *Commun. ACM*, 40(3):66–72, March 1997.

[6] Wellman Barry. Computer networks as social networks. In *Science*, volume 293, pages 2031–2034, 2001.

[7] Chumki Basu, Haym Hirsh, and William Cohen. Recommendation as classification: Using social and content-based information in recommendation. In *In Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 714–720. AAAI Press, 1998.

[8] John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithm for collaborative filtering. In *Proceedings of the 14 th Conference on Uncertainty in Artificial Intelligence*, pages 43–52, 1998.

[9] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks*, 30(1-7):107–117, 1998.

[10] Lukas Brozovsky and Vaclav Petricek. Recommender system for online dating service. In *Proceedings of Znalosti 2007 Conference*, Ostrava, 2007. VSB.

[11] Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl, editors. *The Adaptive Web, Methods and Strategies of Web Personalization*, volume 4321 of *Lecture Notes in Computer Science*. Springer, 2007.

[12] Wray L. Buntine, Marko Grobelnik, Dunja Mladenic, and John Shawe-Taylor, editors. *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML PKDD 2009, Bled, Slovenia, September 7-11, 2009, Proceedings, Part I*, volume 5781 of *LNCS*. Springer, 2009.

[13] Robin Burke. Knowledge-based recommender systems, 2000.

[14] Robin Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12:331–370, 2002. 10.1023/A:1021240730564.

[15] Nicolas Delannay and Michel Verleysen. Collaborative filtering with interlaced generalized linear models. *Neurocomputing*, 71(7-9):1300–1310, March 2008.

[16] T. DuBois, J. Golbeck, J. Kleint, and A. Srinivasan. Improving recommendation accuracy by clustering social networks with trust. In *ACM RecSys'09 Workshop on Recommender Systems & the Social Web*, October 2009.

[17] Thomas DuBois, Jennifer Golbeck, and Aravind Srinivasan. Rigorous probabilistic trust-inference with applications to clustering. In *International Joint Conference on Web Intelligence and Intelligent Agent Technology*, WI-IAT'09, pages 655–658, Washington, DC, USA, 2009. IEEE Computer Society.

[18] Thomas DuBois, Jennifer Golbeck, and Aravind Srinivasan. Predicting Trust and Distrust in Social Networks. In *2011 IEEE Third Int'l Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third Int'l Conference on Social Computing*, pages 418–424. IEEE, October 2011.

[19] J. A. Golbeck. *Computing and applying trust in web-based social networks*. PhD thesis, University of Maryland, 2005.

[20] Jennifer Golbeck. Generating predictive movie recommendations from trust in social networks. In *Trust Management*, volume 3986 of *iTrust'06*, pages 93–104, Berlin, Heidelberg, 2006. Springer Berlin / Heidelberg.

[21] Jennifer Golbeck. Trust and nuanced profile similarity in online social networks. In *ACM Trans. Web*, volume 3, pages 1–33, 2009.

[22] Jennifer Golbeck and Ugur Kuter. The Ripple Effect: Change in Trust and Its Impact Over a Social Network. In Jennifer Golbeck, editor, *Computing with Social Trust*, Human-Computer Interaction Series, chapter 7, pages 169–181. Springer, 2009.

[23] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35(12):61–70, December 1992.

[24] R. Guha, Ravi Kumar, Prabhakar. Raghavan, and Andrew Tomkins. Progation of trust and distrust. In *13th international conference on World Wide Web*, WWW'04, pages 403–412, 2004.

[25] Chung-Wei Hang and Munindar P. Singh. Trust-based recommendation based on graph similarity. In *13th AAMAS Workshop on Trust in Agent Societies*, 2010.

[26] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, January 2004.

[27] Paul Jaccard. The Distribution of the Flora in the Alpine Zone. *New Phytologist*, 11(2):37–50, 1912.

[28] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *12th international conference on World Wide Web*, WWW'03, pages 640–651, New York, NY, USA, 2003. ACM.

[29] Ken Lang. Newsweeder: Learning to filter netnews. In *in Proceedings of the 12th International Machine Learning Conference (ML95*, 1995.

[30] E. A. Leicht, Petter Holme, and M. E. J. Newman. Vertex similarity in networks, October 2005.

[31] Jure Leskovec, Deepayan Chakrabarti, Jon Kleinberg, Christos Faloutsos, and Zoubin Ghahramani. Kronecker graphs: An approach to modeling networks. *J. Mach. Learn. Res.*, 11:985–1042, March 2010.

[32] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. Predicting Positive and Negative Links in Online Social Networks. March 2010.

[33] Paolo Massa and Paolo Avesani. Trust-Aware Collaborative Filtering for Recommender Systems. In Robert Meersman and Zahir Tari, editors, *On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE*, volume 3290 of *Lecture Notes in Computer Science*, chapter 31, pages 492–508. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2004.

[34] Paolo Massa and Paolo Avesani. Trust-aware recommender systems. In *Proceedings of the 2007 ACM conference on Recommender systems*, RecSys '07, pages 17–24, New York, NY, USA, 2007. ACM.

[35] Paolo Massa and Paolo Avesani. Trust metrics on controversial users: balancing between tyranny of the majority and echo chambers. *International Journal on Semantic Web and Information Systems*, 2007.

[36] Paolo Massa and Paolo Avesani. Trust Metrics in Recommender Systems. In John Karat, Jean Vanderdonckt, and Jennifer Golbeck, editors, *Computing with Social Trust*, Human-Computer Interaction Series, chapter 10, pages 259–285. Springer London, London, 2009.

[37] Paolo Massa and Bobby Bhattacharjee. Using Trust in Recommender Systems: An Experimental Analysis Trust Management. In Christian Jensen, Stefan Poslad, and Theo Dimitrakos, editors, *Trust Management*, volume

2995 of *LNCS*, chapter 17, pages 221–235. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2004.

[38] Raymond J. Mooney and Loriene Roy. Content-based book recommending using learning for text categorization. In *Proceedings of the SIGIR-99 Workshop on Recommender Systems: Algorithms and Evaluation*, Berkeley, CA, August 1999.

[39] Daire O'Doherty, Salim Jouili, and Peter Van Roy. Towards trust inference in bipartite social networks. In *Proceedings of the Second ACM SIGMOD Workshop on Databases and Social Networks, DBSocial 2012, Scottsdale, USA*, ACM, 2012.

[40] Daire O'Doherty, Salim Jouili, and Peter Van Roy. Trust-based recommendation: an empirical analysis. In *Submitted to: Proceedings of the Sixth ACM SIGKDD Workshop on Social Network Mining and Analysis SNA-KDD, Beijing, China*, ACM, 2012.

[41] John O'Donovan. Capturing trust in social web applications. In Golbeck Jennifer, editor, *Computing with Social Trust*, volume 3, pages 213–257, 2009.

[42] David L. Olson and Dursun Delen. *Advanced Data Mining Techniques*. Springer Publishing Company, Incorporated, 1st edition, 2008.

[43] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.

[44] Michael J. Pazzani and Daniel Billsus. Learning and Revising User Profiles: The Identification of Interesting Web Sites. *Machine Learning*, 27(3):313–331, 1997.

[45] Michael J. Pazzani and Daniel Billsus. Content-based recommendation systems. In Brusilovsky et al. [11], pages 325–341.

[46] P. Resnick, N. Iacovou, M. Sushak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *1994 ACM Conference on Computer Supported Collaborative Work Conference*, pages 175–186, Chapel Hill, NC, 10/1994 1994. Association of Computing Machinery, Association of Computing Machinery.

[47] Paul Resnick and Hal R. Varian. Recommender systems. *Commun. ACM*, 40(3):56–58, March 1997.

[48] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors. *Recommender Systems Handbook*. Springer, 2011.

[49] Gerald Salton, editor. *Automatic text processing*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1988.

[50] J. Ben Schafer, Joseph Konstan, and John Riedi. Recommender systems in e-commerce. In *Proceedings of the 1st ACM conference on Electronic commerce*, EC '99, pages 158–166, New York, NY, USA, 1999. ACM.

[51] Barry Smyth and John O'Donovan. Trust in recommender systems. In *10th international conference on Intelligent user interfaces*, pages 167–174, 2005.

[52] John R. Vacca. *Public Key Infrastructure: Building Trusted Applications and Web Services*. CRC Press, Inc., Boca Raton, FL, USA, 2004.

[53] John R. Vacca. *Computer and Information Security Handbook*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2009.

[54] C. J. van Rijsbergen and Ph. Information Retrieval, 1979.

[55] Patricia Victor, Martine De Cock, and Chris Cornelis. Trust and recommendations. In Ricci et al. [48], pages 645–675.

[56] Patricia Victor, Chris Cornelis, Martine De Cock, and Ankur Teredesai. Trust- and distrust-based recommendations for controversial reviews. *IEEE Intelligent Systems*, 26(1):48–55, 2011.

[57] Patricia Victor, Chris Cornelis, Martine De Cock, and Ankur M. Teredesai. Trust- and distrust-based recommendations for controversial reviews. *IEEE Intelligent Systems*, 26:48–55, 2011.

[58] Frank Edward Walter, Stefano Battiston, and Frank Schweitzer. A model of a trust-based recommendation system on a social network. *Autonomous Agents and Multi-Agent Systems*, 16(1):57–74, February 2008.

[59] Jing Wang, Jian Yin, Yuzhang Liu, and Chuangguang Huang. Trust-based collaborative filtering. In *FSKD*, pages 2650–2654. IEEE, 2011.

[60] Jianshu Weng, Chunyan Miao, and Angela Goh. Improving collaborative filtering with trust-based metrics. In Hisham Haddad and Hisham Haddad, editors, *SAC*, pages 1860–1864, New York, NY, USA, 2006. ACM.

[61] John Yen, Robert Popp, George Cybenko, K.A. Taipale, Latanya Sweeney, and Paul Rosenzweig. The trusted systems problem: Security envelopes, statistical threat analysis, and the presumption of innocence. *IEEE Intelligent Systems*, 20:80–83, 2005.

[62] Elena Zheleva, Lise Getoor, Jennifer Golbeck, and Ugur Kuter. Using friendship ties and family circles for link prediction. In *2nd ACM SIGKDD Workshop on Social Network Mining and Analysis (SNA-KDD)*, 2008.

[63] C.-N. Lausen G. Ziegler. Propagation models for trust and distrust in social networks. In *Information Systems Frontiers*, volume 7, pages 337–358, 2005.

[64] Cai-Nicolas Ziegler and Jennifer Golbeck. Investigating interactions of trust and interest similarity. *Decision Support Systems*, In Press, Corrected Proof.

[65] Cai-Nicolas Ziegler and Georg Lausen. Analyzing correlation between trust and user similarity in online communities. In Christian Jensen, Stefan Poslad, and Theo Dimitrakos, editors, *Trust Management: Second International Conference, iTrust 2004*, LNCS, pages 251–265. Springer, 2004.