Spring 2005

Evaluating online trust using machine learning methods

WeiHua Song
Louisiana Tech University

EVALUATING ONLINE TRUST USING MACHINE LEARNING METHODS

By

WeiHua Song, MSc.

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

COLLEGE OF ENGINEERING AND SCIENCE
LOUISIANA TECH UNIVERSITY

May 2005

LOUISIANA TECH UNIVERSITY

THE GRADUATE SCHOOL

04/26/2005
Date

We hereby recommend that the dissertation prepared under our supervision by Weihua Song entitled Evaluating Online Trust by Machine Learning Methods

be accepted in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Computational Analysis and Modeling

Supervisor of Dissertation Research

Head of Department

CAM
Department

Recommendation concurred in:

Advisory Committee

Approved:
Director of Graduate Studies

Approved:
Dean of the Graduate School

Dean of the College

GS Form 13
(5/03)

ABSTRACT

Trust plays an important role in e-commerce, P2P networks, and information filtering. Current challenges in trust evaluations include: (1) finding trustworthy recommenders, (2) aggregating heterogeneous trust recommendations of different trust standards based on correlated observations and different evaluation processes, and (3) managing efficiently large trust systems where users may be sparsely connected and have multiple local reputations. The purpose of this dissertation is to provide solutions to these three challenges by applying ordered depth-first search, neural network, and hidden Markov model techniques. It designs an opinion filtered recommendation trust model to derive personal trust from heterogeneous recommendations; develops a reputation model to evaluate recommenders' trustworthiness and expertise; and constructs a distributed trust system and a global reputation model to achieve efficient trust computing and management. The experimental results show that the proposed three trust models are reliable. The contributions lie in: (1) novel application of neural networks in recommendation trust evaluation and distributed trust management; (2) adaptivity of the proposed neural network-based trust models to accommodate dynamic and multifacet properties of trust; (3) robustness of the neural network-based trust models to the noise in training data, such as deceptive recommendations; (4) efficiency and parallelism of computation and load balance in distributed trust evaluations; and (5) novel application of Hidden Markov Models in recommenders' reputation evaluation.

APPROVAL FOR SCHOLARLY DISSEMINATION

The author grants to the Prescott Memorial Library of Louisiana Tech University the right to reproduce, by appropriate methods, upon request, any or all portions of this Thesis. It is understood that "proper request" consists of the agreement, on the part of the requesting party, that said reproduction is for his personal use and that subsequent reproduction will not occur without written approval of the author of this Thesis. Further, any portions of the Thesis used in books, papers, and other works must be appropriately referenced to this Thesis.

Finally, the author of this Thesis reserves the right to publish freely, in the literature, at any time, any or all portions of this Thesis.

Author Weihua Song

Date 05/13/2005

GS Form 14
(5/03)

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENTS

I am indebted to the following for their help and support:

Dr. Vir V. Phoha, My PhD advisor, for his invaluable guidance, advice, and generous support throughout my PhD study in Louisiana Tech University.

Dr. Richard Greechie, for his continuous support, encouragement and guidance during the course of my research.

Dr. Galen Turner, for his generous help, advice and instruction with graph theory and number theory.

Dr. Sumeet Dua, for being on my advisory committee and his valuable instructions.

Dr. Weizhong Dai and Dr. Raja Nassar, for their valuable instructions, help and encouragement.

College of engineering and science and CEnIT, for awarding me assistantship.

Ms. Xin Xu, for her helpful discussions.

I would especially thank my husband Dr. Yi Su, my parents and my son for their continuous support and encouragement.

CHAPTER 1

INTRODUCTION

Online consumer trust is crucial for e-commerce (Cole 1998), because the online environment exposes consumers to the threat of possible inappropriate opportunistic behaviors by online vendors, such as masquerading, misuse and unauthorized distribution of personal information, and even credit card fraud. Online consumer trust is important also because it helps consumers build appropriate favorable expectations of what to expect of the vendor (Gefen 2000). Empirical research shows the significant role of trust in eCommerce (Jarvenpaa and Tractinsky 1999).

1.1 Trust and Reputation

Trust is classified into three types according to Ratnasingham and Kumar (2000). The three types are competence trust, predictability trust, and good will trust. *Competence trust* develops in an economic foundation and is concerned with whom to trust and under what circumstances. *Predictability trust* emphasizes the trading partners' consistent behavior so that the traders can make predictions and judgments due to past experiments. This theory is widely accepted by researchers in establishing quantitative trust evaluation models.

Reputation is a concept highly related with trust. In fact, they are often used interchangeably. However, there are differences between the two. According to Wang

and Vassileva (2003), "*trust is an agent's belief in another agent's capabilities, honesty and reliability based on its own direct experience. Reputation is based on recommendations received from other agents.*" In this context, an *agent* is a buyer/seller/vendor (collections of sellers and buyers), or a node with trust systems in the network. Reputation can be centralized, computed by a trust third party, or decentralized.

We treat trust as an agent's personalized belief in another agent's trustworthiness. It can be based on direct experiences. It can also be opinion filtered recommendations from other agents, i.e., a personalized trust function is applied to one or more recommendations to form an agent's own trust belief. In this dissertation, *reputation* is defined as a trust belief from authorized rating agents, for example, from an authorized central agent, from distributed trust agents or from other authorized rating agents.

The common characteristics of trust and reputation are:

- Context Dependent: a user's trust or reputation may differ with contexts. For example, a user may have a good reputation as a TV retailer, but has a less desirable reputation as a computer retailer. Trust models may also vary with contexts. In other words, retailers of expensive products have their reputations updated more frequently than retailers of low-priced products. Also, response speeds of trust models may differ since users' sensitivity to the providers' reputations is different. In all the following discussions, I only consider trust or reputation under one single context.
- Multi-facet: a user's trust or reputation can be measured in different aspects, for example, service (or product) quality and in-time delivery. An overall reputation

is a function of the individual reputation in each aspect. The function varies with users' preferences and reputation standards. Details are discussed in Chapter 2.

- Dynamic: a user's trust or reputation can change. New users experience reputation built-up stage. Their reputation may converge at a certain level, or shift from one convergent level to another. See details in Appendix B.

1.2 Central Reputation, Personalized Trust, and Global Reputation

Reputations can be classified as central reputations and global reputations. In a system where no central reputation is available, an alternative is reliance upon recommendation trust. Recommendation trust is also referenced to as personalized trust.

1.2.1 Central Reputation

Examples of representative central reputation systems in E-commerce are eBay, amazon.com and eOpinion. Other central reputation systems are Sporas (Zacharia and Mae 1999) and REGRET (Sabater and Sierra 2001; Sabater and Sierra 2002) etc. In a central reputation system, transaction ratings of form *{rater, ratee, rate}* are sent to a secured central agent. The central agent periodically updates the traders' reputations based on the transaction ratings received. In general, we have $V = F(\vec{R})$, where V is the central reputation of a user (ratee), and $\vec{R}$ is the collection of the user's transaction ratings. F varies with trust context and trust model M. F may be public or private to the online users. For example, reputation in eBay is a function of the cumulative positive and non-positive ratings for a seller or buyer over several recent periods (a week, a month, or 6 months). Resnick and Zeckhauser (2000) have empirically analyzed eBay central reputation system and concluded that the system does encourage transactions. Figure 1.1

shows how central reputations are obtained. Details of SPORAS is analyzed in Appendix B (on page 99).

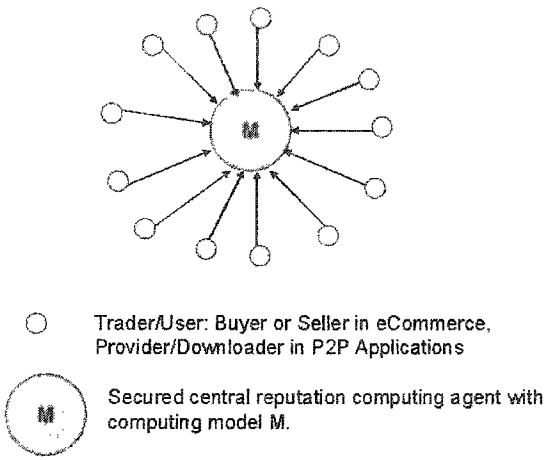


Figure 1.1: A central reputation system: all users send their transaction ratings to a secured central agent. The secured central agent computes the users' central reputations based on model *M*, which is context dependent.

1.2.2 Personalized Trust

Personalized trust refers to an agent's personal trust opinion of others. Personalized trust is based on direct experiences or indirect experiences, i.e., recommendations. It is computed as: $V = f(\vec{R})$, where V is the personalized trust, and $\vec{R} = \{r_1, r_2, \ldots, r_n\}$ represents either the agent's trust ratings from his/her direct experience or the n recommenders' trust ratings, coming from their direct or indirect experience. Function f varies with agents and may be known only to agents themselves. However, through observations, we can learn function f through approximation or machine learning techniques. Chapter 2 approximates function f using graph searching algorithms and neural network techniques.

Mostly, personalized trust is known as recommendation trust, or word-of-mouth trust, where $\vec{R} = \{r_1, r_2, \ldots, r_n\}$ is a set of n recommendations. Representative recommendation trust models include Bayesian model of Mui, Mohtashemi, and Halberstadt (2002), Beyesian network model of Wang and Vassileva (2003), Dempster-Shafer model of Yu and Singh (2001), and HISTOS model of Zacharia and Mae (1999).

1.2.3 Global Reputation

This dissertation differentiates global reputation from central reputation. Global reputation refers to an aggregated reputation from multiple distributed local reputation systems. A user may have different reputations at multiple local systems. Differences in local reputations may result from different local reputation models or trust behavior observations by the distributed local trust agents. The representative distributed reputation models are proposed by Yu and Singh (2002), and by Song and Phoha (2004a). A major difference of the two models is that the former model assumes that a user does not have multiple local reputations. It relies on the social network of the users in a distributed system to get chained recommendations. The later model uses neural network techniques to evaluate the users of multiple local reputations. It distributes the load of memory and computation and monitors the system's performance. Figure 1.2 shows the structure of distributed reputation system designed by Song and Phoha (2004a). Details are analyzed in Chapter 4.

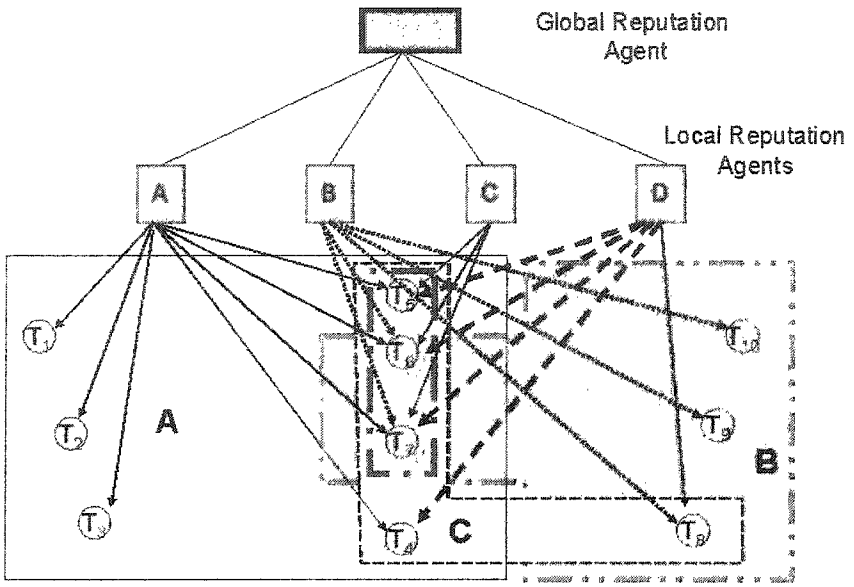


Figure 1.2: A distributed reputation management structure: The entire system is divided into several highly connected local communities managed by local agents - for example, A, B, C, and D. Local agents evaluate users' local reputations using local reputation models. The global agent uses a global reputation model to derive a user's (for instance, user T_6 and T_7) global reputation from his multiple local reputations.

1.3 Objectives

Trust and reputation were first used in e-commerce systems (Schafer, Konstan, and Riedl 1999; Vassileva, Breban, and Horsch 2002) to encourage transactions between strangers. The use of trust and reputation has extended to areas of distributed computing (Azzedin and Maheswaran 2002), file sharing P2P system (Cornelli, Damiani, and Samarati 2002) and information filtering (Montaner, L´opez, and Rosa 2002) etc. However, there are some challenges. The first challenge is given heterogeneous recommendations, how to filter trust opinions. The second challenge is rather than aggregating heterogeneous trust opinions, how to select a trustworthy recommender. The question by nature is how to evaluate recommenders. With the development of trust systems, the third challenge comes. That is, in a very large system how should one evaluate users' trust efficiently? If a distributed reputation system is in use, how should

the system derive a user's global reputation? The objectives of this dissertation are to provide solutions to these questions. The objectives are:

(1) to build an opinion filtered recommendation trust model to form an agent's trust opinion from heterogeneous trust recommendations in a multiagent reputation system;

(2) to build a rating model to evaluate the reputation of recommenders; and

(3) to adopt efficient distribute trust management and build a global reputation model to evaluate users of multiple local reputations.

1.3.1 Filtering Heterogeneous Trust Recommendations

In a system where no global or central reputation mechanism is available, an alternative is to aggregate recommendations. Current recommendation trust models provide various mechanisms to select recommenders, and to aggregate the selected recommenders' trust evaluations. The HISTO model (Zacharia and Mae 1999) is based on the assumption that a requester trusts some agents more than others. The recommended trust is an aggregation of the selected recommenders' reputations, their recommendations and the deviations among the recommendations. Similar to the HISTO model, Riggs and Wilensky (2001) developed another quality filtering trust model. The model rates reviewers and applies the quality of the reviewers into merits of the reviewed papers. The idea is that a reviewer is reliable if he/she consistently ranks papers near their ultimate average. Thus, a reliable reviewer's rate is the actual rate of the paper. Mui, Mohtashemi, and Halberstadt (2002) proposed a recommendation trust model based on Beyesian probability theory. This model studies a parallel referral network between a requester and a party of interest. Yu and Singh (2001) presented an evidence model to

evaluate recommendation trust. It applies Dempster-Shafer theory (Shafer G. 1976) to multiple witnesses. In order to avoid continuous and explicit ratings of references, Pujol, Sang, and Halberstadt (2002) proposed a recommendation trust model based on social network topology. Their model applies a noderanking algorithm to infer a node's reputation. Other collabrative trust models are developed by Azzedin and Maheswaran 2002; Daniani, Vimercati, Paraboschi, Samarati, and Violante 2002; Gupta, Judge, and Ammar 2003; Kamvar, Schlosser, and Garcia-Molina 2003; Schafer, Konstan, and J.Riedl 2002; Wang and Vassileva 2003; Yu and Singh 2003.

However, there are a few complexities that need to be built into the recommendation trust model. First, different recommenders may vary significantly in the estimation of the performance of the same service provider. Second, different recommenders may observe different instances of the performance of a given service provider. Third, deceptive recommendations may exist. To address the complexities, I develop an opinion-filtered recommendation trust model. The model applies graph searching algorithms and neural network techniques, and derives the recommendation trust based on a requester's own trust standard and trust model.

1.3.2 Evaluating Recommenders' Reputations

Instead of integrating heterogeneous trust opinions, we might want to get the recommendations from trustworthy recommenders. However, there is limited research on evaluating an agent's reputation as a recommender. A key challenge is that a recommender's reputation is affected by both the recommender's trustworthiness and the recommender's expertise, including the recommender's trust knowledge of others and the reliability of the recommender's trust evaluation models. In this setting, I develop a

Hidden Markov Model (HMM) based approach to measure an agent's reputation as a recommender. The approach models chained recommendation events as an HMM. Based on the assumption that agents learn to choose reliable recommenders, the transition probability matrix of the HMM actually measures the recommenders' reputations. There are four attractive features of the approach. First, it does not require explicit reputation evaluations of chained recommendations. Second, it integrates a recommender's expertise as well as his/her trustworthiness into his/her reputation evaluations. Third, it is applicable to any possible recommendation network including those with loops and unreachable nodes. Fourth, the approach quantifies the learning speed of a recommender's reputation. The learning speed can be used as the reliability measurement when recommendation events are sparse. The model can be applied to identify optimal recommendation paths and to locate reliable file servers in P2P networks.

1.3.3 Deriving Global Reputation in a Distributed Trust System

With the development of trust systems, an efficient trust management is in demand especially when the system becomes very large. Unfortunately, current centralized trust models are inappropriate to apply in large distributed multi-agent systems. This is mainly because, in a very large distributed trust system, local reputation management may use different trust evaluation models based on partially overlapped observations. Other related issues are computation complexity and efficiency, memory usage, scalability and availability. Under this situation, I aim to develop a global reputation model such that it has the capability (1) to distribute the load of computation and memory among the global reputation management agent and local agents, (2) to allow local or distributed communities using different trust models, (3) to allow

overlapping observations of the same user by different local communities, (4) to adjust to various changes, such as system restructuring, changes of trust models, local communities' sizes and users' trust behaviors etc., and (5) to express global reputations as linear or nonlinear combinations of the local reputations, including some hidden factors that affect the reliability of reputation evaluations such as local agents' observation sizes and feedback deceptions in local communities.

1.4 Dissertation Structure

This dissertation develops trust and reputation models in eCommerce and P2P applications. The dissertation consists of five parts. Chapter 1 is the introduction. Chapter 2 and Chapter 3 provide two different methods to obtain reliable recommendation trust respectively. Chapter 2 studies how to aggregate heterogeneous recommendations and builds an opinion filtered recommendation trust model in multiagent systems. Chapter 3 provides an alternative method to get trustworthy recommendations. Instead of integrating multiple recommendations, it develops a Hidden Markov Model based approach to evaluate agents' reputations as recommenders. Chapter 4 considers a special case in a distributed system where local agents are recommenders. Chapter 4 proposes distributed trust management when a trust system becomes very large and computation efficiency, scalability and availability become the issues. Chapter 4 builds a global reputation model for the users who participate in multiple local reputation systems. Chapter 5 concludes the research work.

In Chapter 2, an opinion filtered recommendation trust model is presented. This chapter studies the problem of heterogeneous and deceptive recommendations in trust management. It focuses on how to accurately and effectively derive trust value of an

unknown party from multiple recommendations. It designs an ordered process of depth-first search for recommenders. It also develops an algorithm to identify qualified recommenders and to aggregate their recommendations. The aggregation is done by back propagation neural techniques. Since the derived trust value is based on an agent's own trust standards, it makes trust decisions easier. The experimental results show that the neural network trust model converges at fast speed with high accuracy. More important, the model performs well under various accuracy requirements and is capable of aggregating multiple recommendations nonlinearly.

In Chapter 3, a Hidden Markov Model based approach is developed to measure an agent's reputation as a recommender. The approach models chained recommendation events as an HMM. Based on the assumption that agents learn to choose reliable recommenders, the transition probability matrix of the HMM actually measures the recommenders' reputations. The Baum Welch algorithm is modified so that it can accommodate cycles and non-reachable states in a recommendation network, and have the capability to model a global maximum of multiple recommendation chains.

In Chapter 4, a distributed trust management structure is proposed. This chapter designs a global reputation model. The model derives global reputations for users with multiple local reputations in a large and sparse distributed system. The distributed reputation model has the capability (1) to allow local or distributed communities using different trust models, (2) to allow overlapping observations of the same user by different local communities, (3) to adjust to changes of system restructuring, local communities' sizes, their trust models, and users' trust behaviors etc., (4) to distribute the load of computation and memory among global reputation management agent and local agents,

(5) to express global reputations in terms of nonlinear combinations of the local reputations as well as the various factors that affect the reliability of reputation evaluations such as observation sizes and rating deceptions. The experimental results showed that a three-layered neural network converges at 4^{th} iteration of the backpropagation algorithm and has accuracy of 94.4% and above when we compared the derived reputation from the distributed reputation model with a centralized reputation model. The rapid convergence speed and high accuracy meets the online reputation management requirements, that is, fairness, responsiveness, and reliability.

CHAPTER 2

OPINION FILTERED RECOMMENDATION TRUST MODEL

A multiagent distributed system consists of a network of heterogeneous peers of different trust evaluation standards. A major concern is how to form a requester's own trust opinion of an unknown party from multiple recommendations, and how to detect deceptions since recommenders may exaggerate their ratings. This chapter presents a novel application of neural networks in deriving personalized trust opinion from heterogeneous recommendations.

2.1 Background and Motivation

Trust evaluations in multiagent systems (Azzedin and Maheswaran 2002; Cornelli, Damiani, and Samarati 2002; Daniani, Vimercati, Paraboschi, Samarati, and Violante 2002; Gupta, Judge, and Ammar 2003) have been studied by many researchers. Examples of recommendation trust models are social network topological model (Pujol, Sang, and Halberstadt 2002), Bayesian rating model (Mui, Mohtashemi, and Halberstadt 2002), Bayesian Network model (Wang and Vassileva 2003), Dempster-Shafer belief model (Yu and Singh 2001; Yu and Singh 2003) and EigenTrust model (Kamvar, Schlosser, and Garcia-Molina 2003).

In a large distributed system of heterogeneous agents, there is no centralized trust storage or management facility. Agents rely on social mechanisms to form their trust opinions of other unknown parties (Montaner, L´opez, and Rosa 2002; Riggs and Wilensky 2001; Schafer, Konstan, and Riedl 1999; Schafer, Konstan, and J.Riedl 2002; Vassileva, Breban, and Horsch 2002; Zacharia and Mae 1999). However, agents of different estimation processes and trust evaluations may evaluate the same online service/product differently. Also, agents may provide deceptive recommendations.

Consider a simple scenario. A requester gets trust opinions from M agents (known as recommenders) about an unknown movie file provider. Assuming all the recommenders use weighted average methods to evaluate the provider's service, if there are m factors affecting the recommenders' trust evaluations, we have:

$$v_i = \sum_{j=1}^{m} w_j^i u_j^i, i = 1, 2, \cdots, M \qquad (2.1)$$

Where v_i stands for Recommender i's trust opinion, w_j^i stands for a weight assigned to trust factor j by Recommender i, and u_j^i stands for Recommender i's trust evaluation of Factor j. For example, u_j can be movie file download speed or file quality. w_j^i and u_j^i are unknown to the requester.

An opinion filtered problem is: given v_i based on Equation (2.1) and a requester's trust evaluation weights $w_j^0 (j = 1, 2, \cdots, m)$, how can someone obtain the requester's trust opinion of the party of interest v_0? We have M equations of $2Mm$ unknown variables.

There are other complexities involved in obtaining opinion filtered recommendation trust. Two of the notable situations are:

(1) Local trust models used by recommenders may vary. If recommenders use weighted average methods to evaluate a single file sharing application, an accumulated evaluation based on past behaviors becomes:

$$v_i = f_i(\sum_{j=1}^{m} w_j^i(t)u_j^i(t)), i = 1, 2, \cdots, M, t = 1, 2, \cdots, T_i \quad (2.2)$$

where f_i stands for Recommender i's trust model, t is the transaction, and T_i stands for total number of trust evaluations. f_i, w_j^i u_j^i and T_i are unknown to the requester.

(2) Recommenders may exaggerate their trust opinions. Equation (2.2) then becomes:

$$v_i = g_i(f_i(\sum_{j=1}^{m} w_j^i(t)u_j^i(t))), i = 1, 2, \cdots, M, t = 1, 2, \cdots, T_i \quad (2.3)$$

Where g_i stands for Recommender i's deception function. g_i is unknown to the requester.

The opinion filtered recommendation problem then becomes: given v_i based on Equation (2.3), a requester's trust evaluation weights $w_j^0(j = 1, 2, \cdots, m)$, and his trust model f_0, how should one obtain the requester's trust opinion of the party of interest v_0? There are $2Mm\sum_{i=1}^{M} T_i$ unknown variables and $2M$ unknown functions in M equations.

This chapter addresses these challenges and provides a solution to filter heterogeneous recommendations. It assumes that a requester's trust opinion is a function of the M recommenders' recommendations:

$$v_0 = F(v_1, v_2, \ldots, v_M). \quad (2.4)$$

What it does is to approximate function F. Suppose there are N movie file providers with whom the requester and the M recommenders all have direct experiences. Given the requester's and the recommenders' trust opinions of the N providers, v_i^j, where $i = 0, 1, \ldots, M$, $j = 1, 2, \ldots, N$, the original opinion filtered recommendation trust problem now changes into an optimal problem, i.e., to find a function F such that the summation of the squared estimation errors is minimized.

$$\min \sum_{j=1}^{N} (v_0^j - F(v_1^j, v_2^j, \ldots, v_M^j))^2 \tag{2.5}$$

Where v_0^j stands for the requester's trust opinion of movie file provider j, and v_i^j stands for recommender i's trust opinion of movie file provider j. Once F is found, we plug in the M recommenders' trust opinions of the party of interest and obtain the requester's trust estimation of the party of interest.

We use neural network techniques to solve optimal problem (2.5). Before solving the optimal problem, we need to find the M recommenders and N movie file providers. The M movie file providers are known as *qualified recommenders* (see Section 2.3). They have direct experiences with the N movie file providers as well as the party of interest. An ordered depth-first search algorithm and an algorithm to identify qualified recommenders are developed for this purpose.

2.2 Development of Depth-First Recommendation Network

An ordered depth-first search algorithm is developed. In order to facilitate an efficient search for recommenders, I rate recommenders and search in the order of their ranks.

2.2.1 Evolution of Ordered Recommender Set

A requester keeps a rated set of recommenders (context dependent). A recommender set R contains both *qualified recommenders* (see Section 2.3) QR and unqualified recommenders NQR. All qualified recommenders are ranked higher than unqualified recommenders. *Qualified recommenders* are ranked by the number of times they have been selected as qualified recommenders. Unqualified recommenders are ranked by the number of times they have selected as recommenders but are excluded from being the qualified recommenders. Initially, the *qualified recommender* set is empty and the unqualified recommender set consists of all the acquaintances of the requester. The acquaintances are ranked by their trust values. Recommender set R is updated and reordered after the requester sends a new query to every recommender $r \in R$ and obtains a current set of *qualified recommenders* QR_c and unqualified recommenders NQR_c. A requester's recommender set is updated as the following:

Algorithm 2.1. Update Ordered Recommender Set

1: for each $r \in QR_c$
2: if $r \in QR$
3: increase the rank of r by 1;
4: end if
5: else if $r \in NQR$
6: remove r from NQR,
7: append r to QR with $rank = 1$;
8: end else if
9: else
10: append r to QR with $rank = 1$;
11: end else
12: end for
13: for each $r \in NQR_c$
14: if $r \in NQR$
15: increase the rank of r by 1;
16: end if
17: else if $r \notin R$
18: append r to NQR with $rank = 1$;

19: end else if
20: end for

2.2.2 Ordered Depth-First Recommendation Network

A requester sends trust queries to the agents in his/her recommender set R. The agents send back their feedbacks. If an agent has direct trade experiences with the party of interest, his/her feedback is a recommendation, including his/her ID and the trust value of the target. Otherwise, the agent sends up to *Ref_Limit* referrals, where *Ref_Limit* is the branching factor of recommendation trust networks. A referral contains a referrer's ID and a referee's ID. The referees are the top ranked recommenders in the referrer's recommendation set R. The referrals are sent to the requester in the order of the referees' ranks. The referrals are also processed in the order of the referees' ranks. Once a referral is processed, it is processed sequentially till one of the following scenarios happens:

(1) A recommender is found;

(2) A referral reaches an agent that ends up nowhere;

(3) The referral chain reaches the chain length limit; or

(4) the referee has already been queried in a previously processed feedback.

The requester then starts processing the next highly ranked feedback till all the feedbacks are processed. That is how a trust recommendation network is built. By building the recommendation trust network in a depth-first style (Cormen, Leiserson, Rivest, and Stein 2001), every referee is queried exactly once. Figure 2.1 shows an example of an ordered depth-first recommendation trust network *trustNet*.

Feedback: Feedback is represented by a C++ struct containing a recommender ID, *fromAgnet_ id*, and a union *V al*. *V al* is either a referee ID, *toAngent_ id*, or the trust value of the party of interest, *trustVal*.

```
struct{
        String fromAgent_ id;
        union{
                String toAgent_ id;
                real trustVal;
        } Val;
}Feedback;
```

Recommendation: If *Feedback.Val* is a real type, the feedback is a recommendation. *Feedback.fromAgent_ id* is the recommender and *Feedback.Val* is the recommended trust value of the party of interest.

Referral: If *Feedback.Val* is a String type, the feedback is a referral. *Feedback.Val* is the referee ID.

Referral Chain: A referral chain is a sequence of referrals, where the previous referee is the next referrer in any two contiguous referrals.

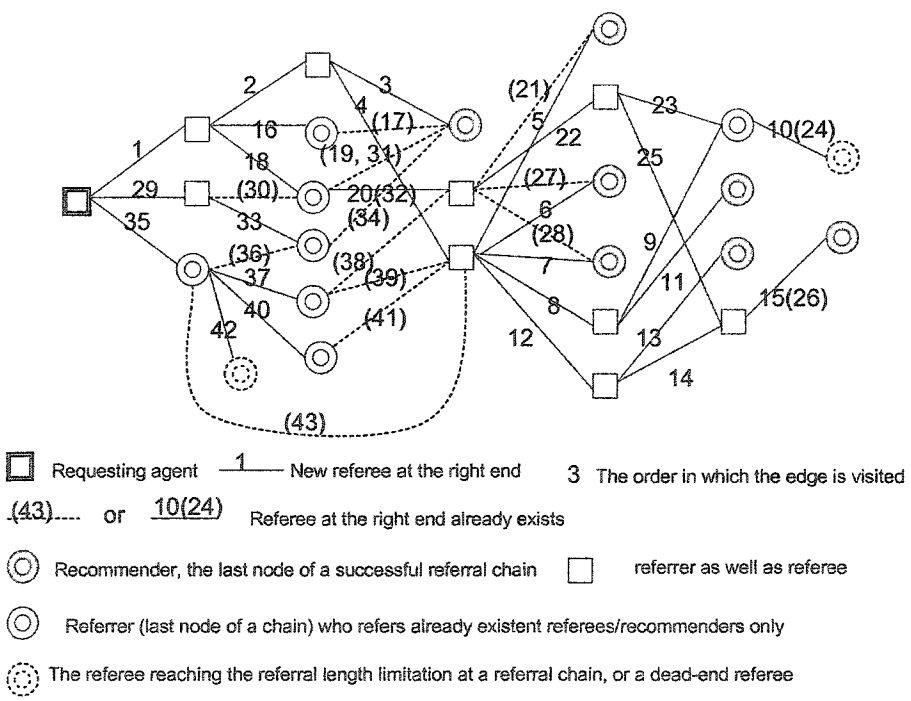


Figure 2.1: Ordered depth-first recommendation trust network. The edges are numbered in the order of the reference. A number inside a pair of parenthesis shows a repeated reference. A solid edge represents a first time reference. A dotted edge represents a reference already processed.

In Figure 2.1, users are identified by different shapes such as requester, recommender, and intermediate recommender (referrer as well as referee). The detailed algorithm (developed based on standard depth-first search algorithm by Cormen, Leiserson, Rivest, and Stein 2001) is shown in Algorithm 2.2 ODFS and Algorithm 2.3 ODFS-Visit, where nodes' colors are used to represent the search status. For example, a white node stands for the node whose reference is not yet processed; a gray node stands for the node whose reference and the chained sequences of the references have not yet fully searched; and a black node stands for the node whose references and the correspondent chained sequences of references have been fully searched.

Algorithm 2.2. Ordered Depth-First Search. ODFS (Requester)

```
1: length ← 0
2: for each r ∈ R of the Requester's recommender set (initialization)
3:       color[r] ← white (white: unprocessed)
4:       parent[r] ← nil
5: end for
6: time ← 0
7: for each r ∈ R (search)
8:       if color[r] = white
9:                length ← length + 1
10:               ODFS-Visit (r, Requester)
11:      end if
12: end for
```

Algorithm 2.3. Ordered Depth-First Visit. ODFS-Visit (r, Requester)

```
1: color[r] ← gray (gray: in process)
2: time ← time + 1
3: d[r] ← time (set discover time)
4: if r has direct experience or length > limit
(r is a recommender or the recommendation chain has reached length limit)
5:       color[r] ← black (black: finished)
6:       parent[r] ← Requester
7:       f[r] ← time ← time + 1 (finish time)
8:       length ← length-1 (backtrack one level)
9:       return
10: end if
```

```
11: for each recommender i in r's ordered recommender set
12:     if color[i] = white
13:             parent[i] ← r
14:             length ← length + 1
15:             ODFS-Visit(i, r)
16:     end if
17: end for
18: color[r] ← black
19: f[r] ← time ← time + 1 (finish time)
```

2.3 Identification of Qualified Recommenders

Qualified recommenders (*QRc*) is a subset of recommenders *Rc* obtained through building *trustNet*. Their trust opinions are used as training data to build a neural network (see details in Section 2.4). Algorithm 2.4 summarizes the process of selecting qualified recommenders. First, the requester selects top N active movie providers with whom he has direct experiences and exchanges his opinions with the recommenders. A two dimensional array *RP* is built where element *RP*[*i*][*j*] is either 1 or 0, representing that recommender $r_i \in R_c$ has or has not direct trust experiences with movie file provider $p_j \in P$. Recommenders that know less than $ceil_1$ movie file providers (in our case, 18) are excluded and movie providers that are known by less than $ceil_2$ recommenders (in our case, 4) are also excluded. A new *RP* array is built based on the selected recommenders and movie file providers. The top $ceil_3$ (in our case, 4) recommenders that know the majority of the movie file providers are selected as qualified recommenders. Those movie file providers that are known by all the $ceil_3$ qualified recommenders form new P.

Qualified Recommenders: recommenders who have direct experiences with a set of movie file providers that the requester also has direct experience with.

Algorithm 2.4. Identify Qualified Recommenders

1: build RP table;
2: set $T[i] = \sum_j RP[i][j]$;
3: set $S[j] = \sum_i RP[i][j]$;
4: set $QRc = \{r_i\}$, where $T[i] \geq ceil_1$;
5: set $P = \{p_j\}$, where $S[j] \geq ceil_2$;
6: rebuild RP table;
7: set $T[i] = \sum_j RP[i][j]$;
8: sort $T[i]$ and select the top $ceil_3$ recommenders as QRc;
9: set $P = \{p_j\}$, where $RP[i][j] = 1$ for each $r_i \in QR_c$;

2.4 Neural Network-Based Opinion Filtered Recommendation Trust Model

A recommendation trust neural network is trained by the qualified recommenders' and the requester's trust opinions of the movie file providers ($\forall p_j \in P$). The model is adaptable based on the requester's accuracy requirement and the dynamic nature of online trust (see Section 2.4.2).

2.4.1 Artificial Neural Network of Recommendation Trust

A recommendation trust neural network is composed of highly interconnected processing neurons. The neurons work together to estimate the requester's trust opinion of an unknown party from heterogeneous recommendations. Like people, a recommendation trust neural network learns by examples. A recommendation trust neural network has one input layer, one or more hidden layers, and one output layer. There are $|QRc|$ neurons in the input layer, which receives the trust opinions from the qualified recommenders QRc. The optimal number of hidden layers and the optimal number of neurons in those hidden layers are mainly determined by the nonlinear relationship between the heterogeneous recommendations and the desired convergence speed of the

model as well as the accuracy requirement. Since the desired output is an opinion filtered recommendation trust, we set only one neuron in the output layer.

The backpropagation algorithm (Mitchell 1997) is used to train the neural network. It adjusts the weights of neuron connections until an optimal estimation error is achieved. Iterations of three steps are involved:

(1) Qualified recommenders' trust opinions flow forward through the neural network. The output of Neuron i is:

$$o_i = \sigma(\sum_j v_j w_{ji}) \tag{2.6}$$

Where j stands for a neuron in the input layer if i is a neuron in the hidden layer, or, j is the neuron in the hidden layer if i is a neuron in the output layer. v_j stands for the input value of Neuron j. If j is a neuron in the input layer, v_j stands for Recommender j's trust opinion. Otherwise it stands for the input value from Neuron j in the hidden layer. w_{ij} represents the weight assigned to the connection between Neuron i and Neuron j. σ is the sigmoid function. It is also known as logistic function:

$$\sigma(y) = \frac{1}{1+e^{-y}} \tag{2.7}$$

(2) Approximation errors flow backward. The error term of the only one neuron at the output layer is:

$$\delta_k = o_k(1-o_k)(t-o_k) \tag{2.8}$$

Where Neuron k is the only neuron in the output layer. o_k stands for an output trust, specifically, an approximated opinion filtered recommendation trust. T stands for an actual trust opinion of the requester. $(t - o_k)$ stands for the difference

between an estimated recommendation trust and an actual trust opinion of the requester. $o_k(1-o_k)$ is the derivative of sigmoid function o_k. The error item of Neuron j in the hidden layer is:

$$\delta_j = o_j(1-o_j)w_{jk}\delta_k \tag{2.9}$$

Where o_j stands for the output of Neuron j in the hidden layer. w_{jk} stands for the weight of the connection between the hidden layer Neuron j and the only output layer Neuron k.

(3) Connection weights are adjusted based on the errors. The squared estimation errors are reduced for each data flow iteration. The purpose of backpropagating error items is to adjust the weights assigned to the neuron connections and decrease the errors. Weights are updated as:

$$w_{ij} = w_{ij} + \eta\delta_j v_i \tag{2.10}$$

Where i stands for an input layer or hidden layer neuron, j stands for a hidden layer or output layer neuron, v_i stands for the input value of Neuron i, and η stands for learning speed.

The three-step process continues until a stop condition satisfies. The stop condition can be an acceptable approximation error size, number of weight updates (known as iterations), or a certain accuracy level, or any combinations of the three. Algorithm 2.5 summarizes the training process of a neural network. Algorithm 2.6 develops an opinion filtered recommendation trust model.

Algorithm 2.5. Train Trust Network
1: initiate neural network *neuralNet*;
2: set up stop condition *stopCond*;
3: while (*stopCond* is not satisfied)
4: for each $p_i \in P$

```
5:              calculate the output of neurons;
6:              propagate errors backward;
7:              update connection weights ;
8:      end for
9: end while
```

Algorithm 2.6. Neural Network-Based Trust Model

1: ordered depth-first search for recommenders *Rc*;
2: identify *QRc* (See Algorithm 2.4);
3: update ordered recommender set *R*(see Algorithm 2.1);
4: train trust neural network (see Algorithm 2.5);
5: input recommendations from *QRc* into the nueral network;
6: output the opinion filtered recommendation trust

2.4.2 Adaptability and Optimization of the Model

One of the advantages of the neural network-based recommendation model is its adaptability and flexibility. The model is designed to be able to catch the dynamic nature of online trust, such as changes of trust behaviors, changes of trust models, and changes of expertise of agents. Figure 2.2 shows the architecture of the model. A requester keeps the last trained neural network in its memory. Once a new query of the same trust category comes up, the requester first communicates with the qualified recommenders whose opinions had been used to build the neural network (see Step 1 in Figure 2.2). If all of them know the new party of interest, the requester inputs their trust opinions through the neural network and immediately gets the requester's own trust opinion (see Step 6 following the thick solid arrows in Figure 2.2). However, if the neural network was built beyond a certain period of time, or some qualified recommenders have changed their trust models, or the requester changes his/her trust estimation accuracy requirement, the requester needs to collect up-to-date trust data and retrain the neural network (see Step 7 following the dotted arrows in Figure 2.2). If not all the neural network recommenders can provide current trust recommendations, the requester needs to build

recommendation trust network, identify qualified recommenders, update his/her rated recommendation set and build a new neural network (see Step 2-5 following the thin solid arrows in Figure 2.2).

To save the time spending on the search of current recommenders, identification of current qualified recommenders and train of a new neural network, each agent may save several most recently used neural networks (under the same trust context). An agent searches for current recommenders only when all the recommendation neural networks are unable to provide recommendations for the current trust query. In this case we trade memory for speed.

2.5 Experimental Results on Simulation Data

My experiments are based on the simulation result of movie file sharing in a P2P network of 50 agents. Total 500 transactions were simulated. Movie file providers and downloaders were generated randomly from the 50 agents. Each movie file provider was randomly assigned an average file quality value and file download speed value. A movie file provider's trust behavior is evaluated by the weighted average of file quality and file download speed. The weights of those two factors are normalized and vary from one agent to another. The weights are unknown to other agents. I also simulated both deceptive and nondeceptive recommendations. I assume a recommendation trust follows:

$$v_{rec} = min(1, cv_{act}) \tag{2.11}$$

Where c is a factor larger than 0. v_{act} is the actual trust rating, and v_{rec} is the recommendation. If $c = 1$, the recommendation is honest. If $c < 1$, the recommendation is exaggeratively low, and if $c > 1$, it is exaggeratively high.

Three experiments were conducted based on the simulation data. Experiment 1 tests the model's convergence speed under various estimation accuracy requirements. Experiment 2 tests the robustness of the model with increased training data sets containing deceptive recommendations. Experiment 3 trains the trust model by the first half of the training data and uses the second half of the training data to test the model's reliability. I set 10 different estimation error sizes and ran the opinion filtered recommendation trust model (see Algorithm 2.5) twenty times for each error size. To train a trust neural network, I randomly set learning speed η in a range of [0.4, 0.6]. I randomly set the initial weights of the connections in a range of $[-0.05, 0.05]$. The stop condition of training the neural network was set as logic AND combination of: (1) there were at least 15 out of 16, or 17 out of 18, or 32 out of 36 correct estimations (see *correctness* definition in Equation (2.13)), given that there were 16, 18 and 36 sets of training data separately, (2) the summation of the squared estimation errors over the training data sets was less than 0.2, and (3) total iterations of training the neural network were less than 2,000,000.

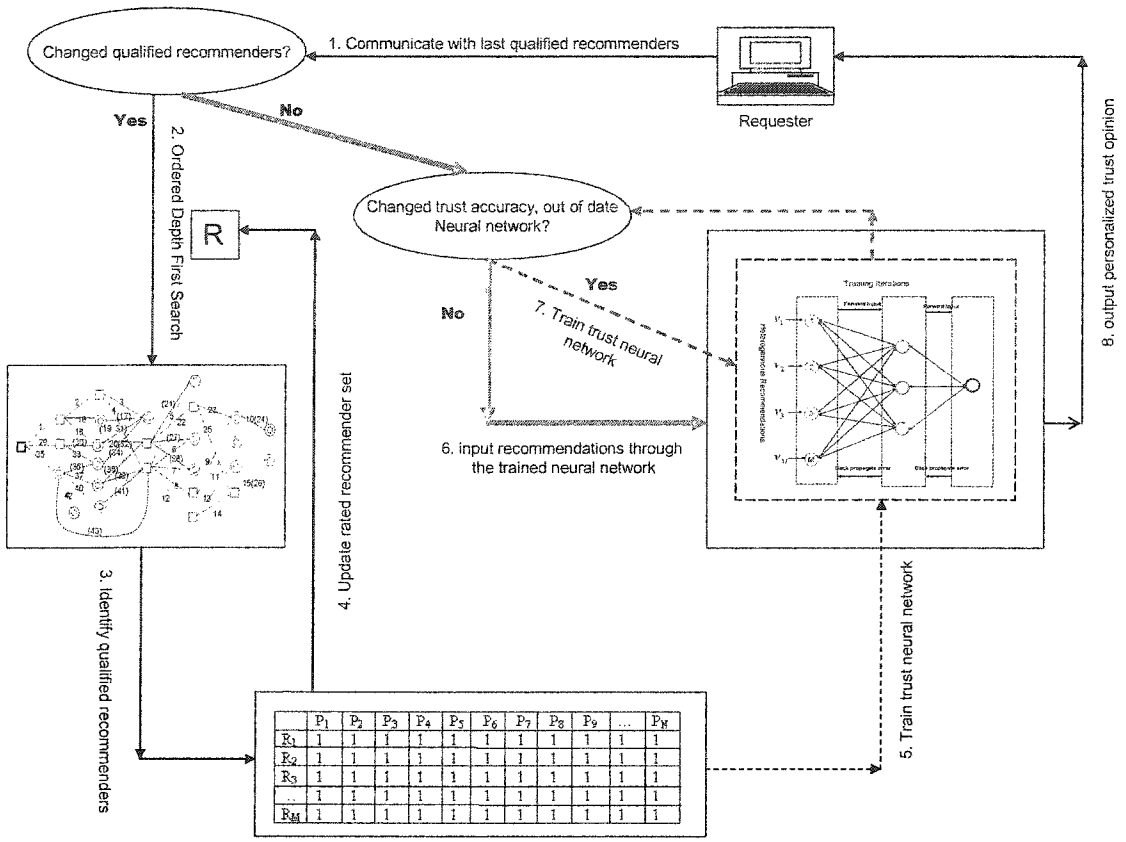


Figure 2.2: Adaptive recommendation trust neural network. The neural network is adaptive to various changes in trust behaviors, trust evaluations, recommenders' expertise and requester accuracy requirement .

2.5.1 Convergence Speed and Accuracy

Experiment 1 generated four qualified recommenders based on 200 transactions. There were 16 movie file providers with whom the requester and the recommenders all had direct experiences. Table 2.1 provides their trust opinions on those 16 movie file providers. Let o_i stand for the output trust value of movie file provider p_i from the neural network model, and t_i stand for the requester's actual trust opinion of p_i. *err* is defined as:

$$err = \frac{1}{2}\sum_i (o_i - t_i)^2 . \qquad (2.12)$$

A *correct* estimation is defined as the one satisfying $|o_i - t_i| \leq \theta$ (θ is a constant), i.e.,

$$T(o_i, t_i) = \begin{cases} 1, & for \mid o_i - t_i \mid \leq \theta \\ 0, & otherwise. \end{cases} \tag{2.13}$$

Table 2.1. Heterogeneous Trust Opinions

Movie File Provider	Qualified Recommender QR_c				Requester
	qr_1	qr_2	qr_3	qr_4	r_0
p_1	0.4	0.45	0.55	0.35	0.6
p_2	0.46	0.5	0.58	0.42	0.62
p_3	0.76	0.75	0.73	0.77	0.72
p_4	0.58	0.6	0.64	0.56	0.66
p_5	0.72	0.75	0.81	0.69	0.84
p_6	0.7	0.7	0.7	0.7	0.7
p_7	0.52	0.55	0.61	0.49	0.64
p_8	0.68	0.7	0.74	0.66	0.76
p_9	0.74	0.75	0.77	0.73	0.78
p_{10}	0.78	0.8	0.84	0.76	0.86
p_{11}	0.72	0.75	0.81	0.69	0.84
p_{12}	0.56	0.6	0.68	0.52	0.72
p_{13}	0.66	0.7	0.78	0.62	0.82
p_{14}	0.7	0.7	0.7	0.7	0.7
p_{15}	0.64	0.65	0.67	0.63	0.68
p_{16}	0.52	0.55	0.61	0.49	0.64

Where θ is an error threshold, o_i and t_i stands for the output trust and the requester's actual trust evaluation of movie file provider p_i. I ran the opinion filtered recommendation trust model twenty times on an initial neural network with a randomly assigned learning rate and neuron connections. The purpose is to examine the performance of the model under different neural network parameters (finding optimal parameters are out of our current discussion scope). Figure 2.3 demonstrates the convergence speed of the model when 93.75% estimations have error size less than 0.08. The average convergence speed was only 4545 iterations. The summation of estimation errors, i.e., *err*, varied from 0.019 to 0.020 for each of 20 runs.

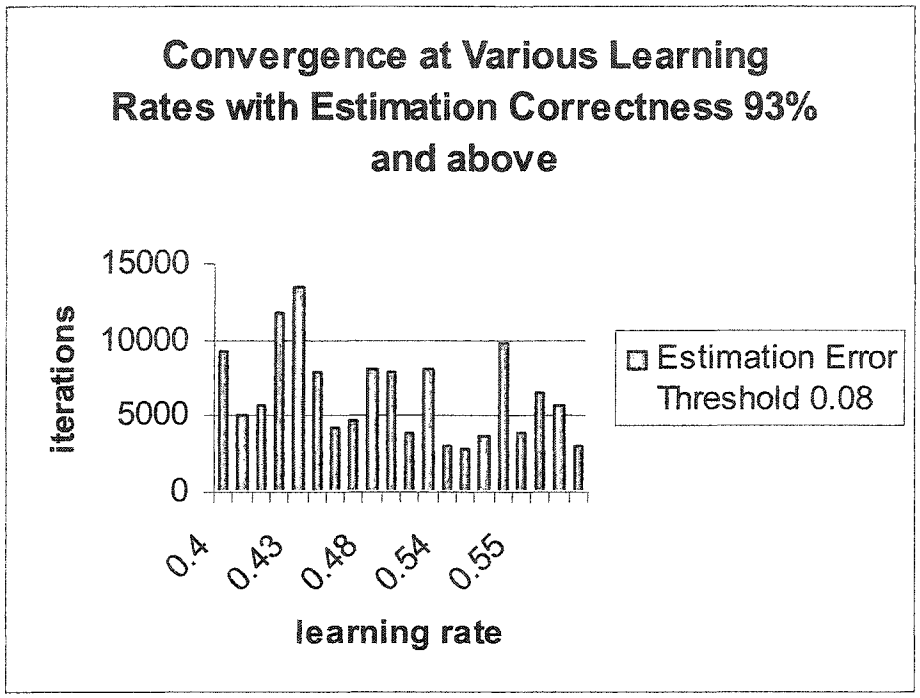


Figure 2.3: Convergence at various learning rates with estimation correctness 93% and above, where $\theta = 0.08$. All 20 runs had *err* in the range of [0.019, 0.020].

Table 2.2 shows the average convergence speed and estimation correctness under various estimation error thresholds. The estimations were 100% correct if the estimation threshold was allowed to be no more than 0.15. The model converged at 404,047 iterations. To increase the estimation accuracy such that the allowable error threshold is less than 0.05, I had 93.8% correct estimations. The average convergence speed was 1,435,770 iterations.

2.5.2 Reliability

I tested the reliability of the model in terms of the model's convergence speed and accuracy by:

(1) varying the estimation accuracy requirement, i.e., error threshold θ;

(2) introducing deceptions to the recommendations;

(3) varying training data sizes;

(4) varying learning rates;

(5) varying a mix of the above parameters;

(6) using testing data to test the accuracy of the trust model.

Figure 2.4 compares the convergence of the model at different estimation error thresholds. It shows that when θ was decreased from 0.08 to 0.05, in order to maintain the same high estimation correctness level, more iterations are required. Of the 20 runs, an average of 9275 iterations was taken when $\theta = 0.05$, which was 2834 more than the average iterations when $\theta = 0.08$.

Figure 2.5 compares the convergence of the model with and without deceptive recommendations under estimation error threshold $\theta = 0.05$. Deceptive recommendations follow Equation (2.11), where $c = 1, 1.3, 1.2$, and 0.9 for the four qualified recommenders individually. c is unknown to the requester. I ran the trust model 20 times with random learning rates and random initial connection weights. It took only 2023 more iterations in average to detect deceptive recommendations. All results, with or without deceptions, had 93% correct estimations. Deceptive recommendations do not have significant impact on the model's accuracy and convergence speed.

Table 2.2. Convergence and Correctness of Opinion Filtered Trust Model

Estimation Error θ	Learning Rate η	Convergence (number of iterations)	Correctness $\sum_i^{16} T(o_i, t_i)$
30%	0.485	54042	16
25%	0.48	165260	16
20%	0.49	273207	16
15%	0.515	404047	16
10%	0.555	551989	15
9%	0.475	684838	15
8%	0.5	810923	15
7%	0.59	1006285	15
6%	0.595	1221718	15
5%	0.595	1435770	15

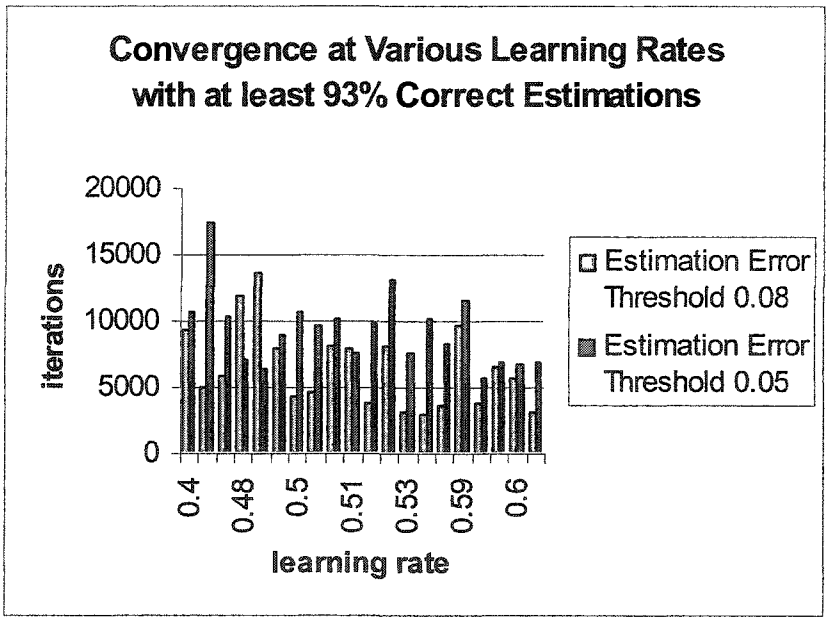


Figure 2.4: Convergence at various learning rates with at least 93% estimation correctness. Extra 2834 iterations were taken to detect deceptions and maintain the same level of estimation correctness.

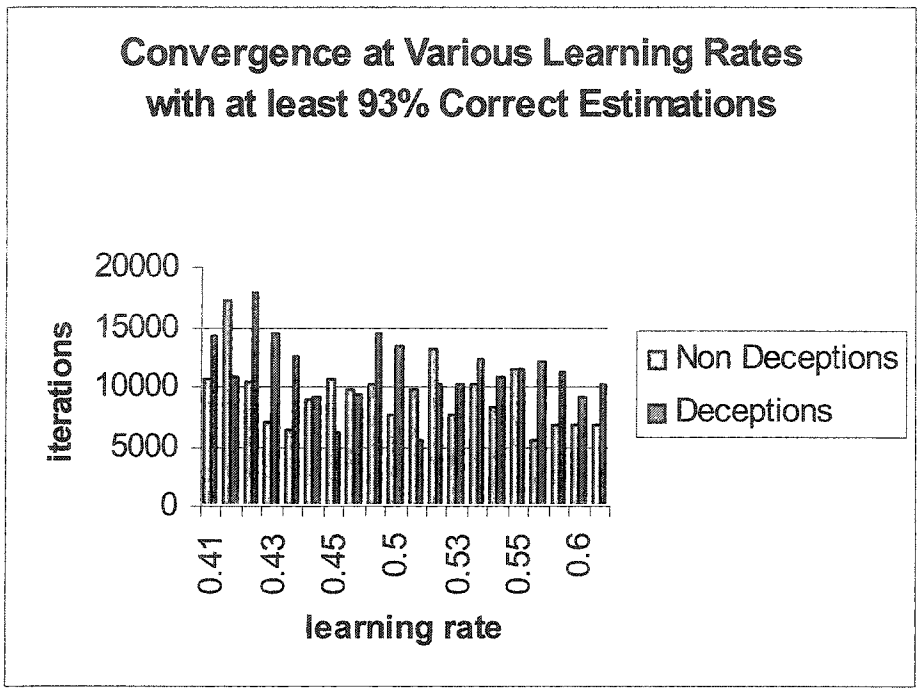


Figure 2.5: Convergence at various learning rates with at least 93% correct estimations. Estimation error threshold $\theta = 0.05$.

Figure 2.6 and Figure 2.7 show the performance of the opinion filtered trust model under different accuracy requirement with and without deceptive recommendations, where $c = 1$, 1.3, 1.2, and 0.9 (see Equation (2.11)) for the four qualified recommenders individually. c is unknown to the requester. The results show that the model converged at an average of 34.8% more iterations under deceptive recommendations than under nondeceptive recommendations when estimation error thresholds were set less than 0.15. When estimation error thresholds were in the range from 0.15 to 0.30, surprisingly, the model converged faster by 12.5% under deceptive recommendations than under nondeceptive recommendations. This might result from differences in randomly generated learning rates and initial neuron connections. Estimation error threshold $\theta = 0.10$ is a critical point. Once θ is less 0.10, the

convergence speed dramatically slows down. Figure 2.6 and 2.7 both demonstrate that deceptive recommendations do not significantly affect the reliability of the model.

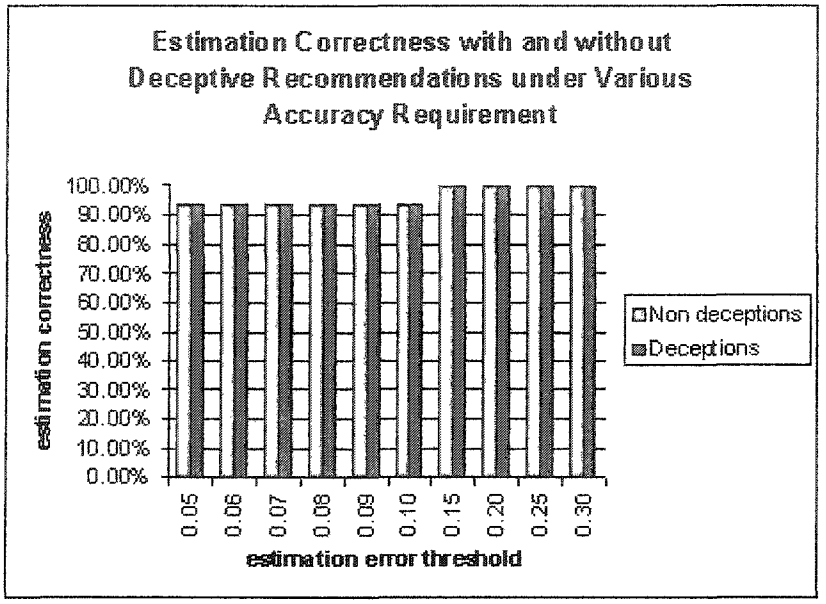


Figure 2.6: Estimation correctness of the trust model with and without deceptions under various estimation accuracy requirements.

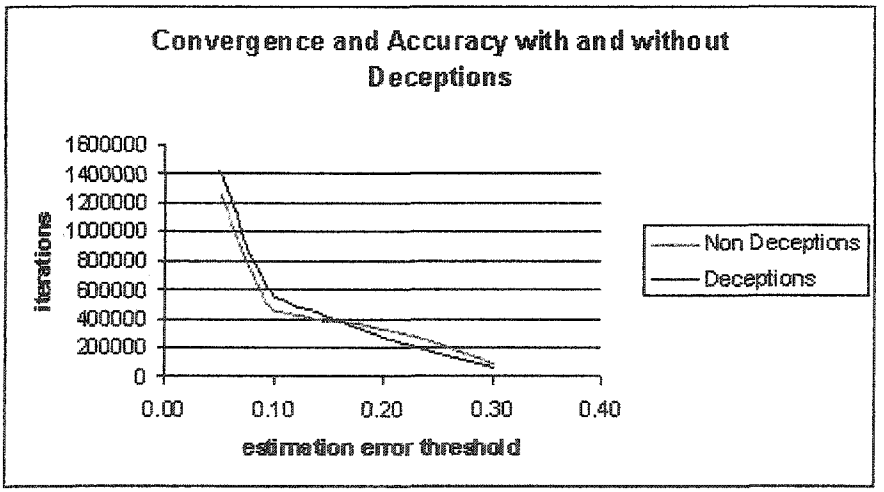


Figure 2.7: Convergence of the trust model with and without deceptions under various estimation accuracy requirements.

Experiment 2 generated four qualified recommenders based on all the 500 transactions. There were 36 movie file providers with whom the four qualified recommenders and the requester had direct experience. I compared the model's

performance trained by 36 sets of recommendations with the performance by 16 sets of recommendations. Both recommendation sets contained deceptions. Figure 2.8 shows that the convergence speed of the trust neural network trained by 16 sets of recommendations was slower by 6.4%. The reason is that since there was not enough training data to catch deceptions, it took more iteration to achieve the same level of estimation accuracy. Figure 2.9 compares correctness estimations under both cases. It demonstrates that the model provided comparable correctness estimations under 36 sets of recommendations with 6.4% less iterations. Thus, the model is robust and reliable.

In Experiment 3, four qualified recommenders gave trust recommendations on 36 movie file providers. I used the trust opinions of the first 18 movie file providers as the training data and the trust opinions of the second 18 file providers ($p_i^t, i = 1, 2, \ldots 18$) as the testing data to test the accuracy of the model. Table 2.3 shows the experimental results, where $|o_i - t_i|$ stands for the estimation error. The average estimation error is less than 0.012 for the training data set without deception recommendations and less than 0.015 for the training data set with deceptions. The estimations over the 18 testing data set were 100% for both nondeceptive data and deceptive data. Of all the 36 training sets, the estimation correctness was 97.2% with and without deceptive recommendations.

2.6 Experimental Results on Real Data

Due to the unavailability of eCommerce trust data, I design a survey as an alternative to test the proposed neural network-based recommendation trust model. Based on the real data collected from the survey obtained through ordered depth-first search, a neural network trust model is constructed. Neural network training, validation and testing techniques are applied to the constructed model.

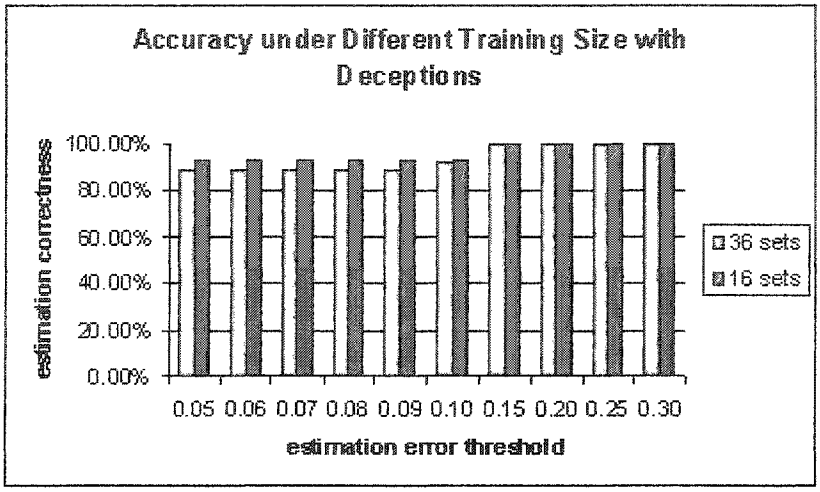


Figure 2.8: Estimation correctness of the trust model trained by 16 and 36 sets of recommendations under various estimation accuracy requirements. The training data contain deceptive recommendations.

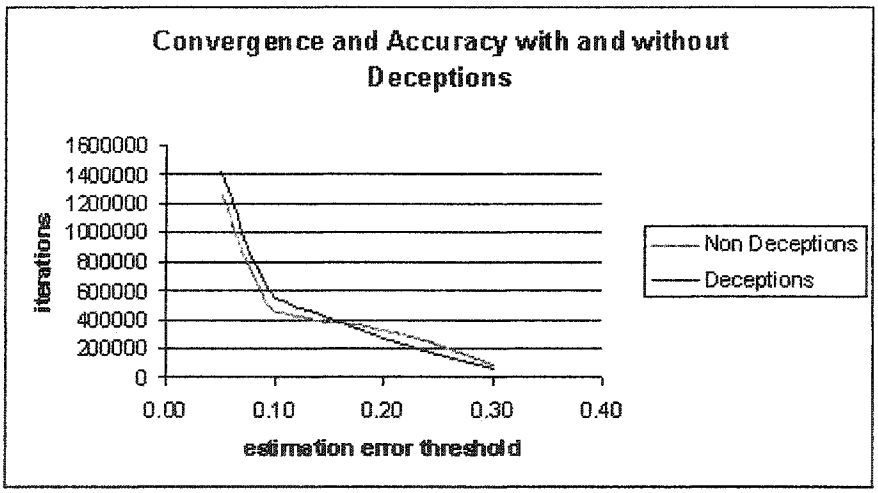


Figure 2.9: Convergence of the trust model trained by 16 and 36 sets of recommendations under various estimation accuracy requirements. The training data contain deceptive recommendations.

Table 2.3: Estimation error size of the opinion filtered recommendation trust model on the test data with and without deceptive recommendations. The neural network was trained at $\theta = 0.05$ with 94.4% correctness.

Movie File Provider	Non Deceptions $\lvert o_i - t_i \rvert$	Deceptions $\lvert o_i - t_i \rvert$
p_1^t	0.000635	0.004730
p_2^t	0.000761	0.006260
p_3^t	0.013394	0.010787
p_4^t	0.000045	0.004974
p_5^t	0.007191	0.005288
p_6^t	0.024573	0.033638
p_7^t	0.000761	0.006260
p_8^t	0.037626	0.042090
p_9^t	0.013394	0.010787
p_{10}^t	0.007191	0.005288
p_{11}^t	0.050580	0.056620
p_{12}^t	0.000761	0.006260
p_{13}^t	0.029345	0.045171
p_{14}^t	0.007191	0.005288
p_{15}^t	0.002656	0.001046
p_{16}^t	0.000635	0.004730
p_{17}^t	0.000761	0.006260
p_{18}^t	0.007191	0.005288
Average	0.011372	0.014537

2.6.1 Survey Design and Trust Model Construction

The purpose of the survey is to build a recommendation network and select qualified recommenders. The recommenders are faculty members in the computer science program, graduate students in the CAM and computer science programs and undergraduates in computer science at Louisiana Tech University. Recommendation chains are developed through ordered-depth first search, where faculty members have the highest rank, followed by senior graduate students, junior graduate students and undergraduate students. Of the recommendation network, four chains are fruitless and the last nodes of the chains do not provide any feedback. Figure 2.10 shows the recommendation network we developed in the survey. Different from the simulation where a node in the recommendation network is either a direct recommender or an

intermediate recommender, a node in the survey recommendation network can be both the direct recommender and the intermediate recommender. We chose some well known computer and mathematic books, widely used programming software, search engines, and eCommerce websites as the target objects for recommendations. Recommendation rates are real numbers in the range of 0 and 1, where 0 means least satisfaction and 1 means highest satisfaction. Out of the 16 recommenders, we selected the five recommenders who rated all the common 30 objects (see details in Appendix A). Of the five recommenders, we randomly selected four as the qualified recommenders and the other one as the requester. Thus the input of the recommendation neural network is the four qualified recommenders' rates. The requester's rates are used as the target values to adjust the weights of the neural network such that the output of the neural network trust model provides the least square estimation of the target values. Table 2.4 lists the recommendation rates.

2.6.2 Performance of the Neural Network Recommendation Model

Two sets of performance testing are conducted in terms of convergence speed and accuracy. Set one tests the performance of the trained recommendation neural network only. Set two validates and tests the performance of the trained recommendation neural network.

2.6.2.1 Testing the Model I use the first 15 sets of the recommendation data in Table 2.4 as the training data to train the neural network. I then use the second 15 sets of the data as the testing data to test the performance of the established neural network model. To train the neural network, I randomly set the learning rate and initial weights of the neural

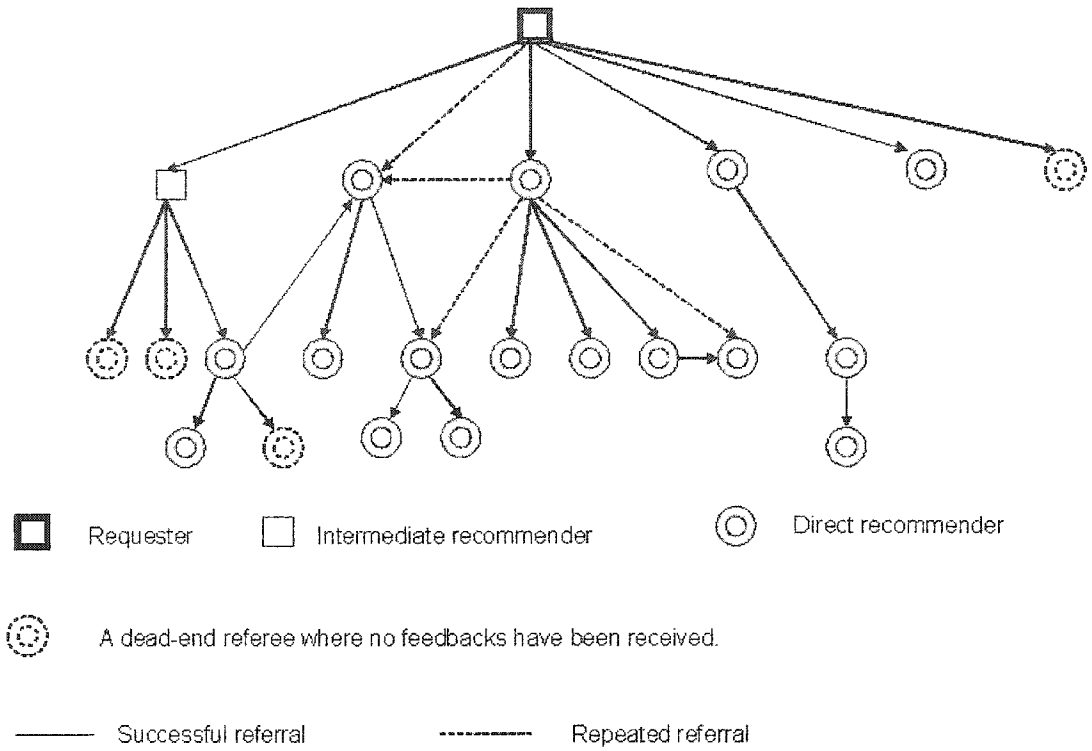


Figure 2.10: Ordered depth-first recommendation network in the survey. A solid edge represents a first time reference. A dotted edge represents a reference already processed.

network. Twenty experiments were carried out. The average estimation error size is 0.025, the average estimation correctness is 93.33%, and the average convergence speed is 140823. Using these twenty trained neural networks, I test how accuracy they are by the second 15 sets of the survey data. Table 2.5 shows that 6 out of the 20 trust models have estimation correctness higher than 73.3% and 12 out of the 20 trust models have estimation correctness of 60%, and the rest 2 trust models have estimation correctness of 53.3%. By average, the estimation correctness is 63.3%. The experiments indicate that the model may be overfitted to the training data sets since the estimation correctness of the test data is not very high. Also, the six trust models with the highest testing accuracy indicate that we may optimize the recommendation neural network model by adjusting

the initial weights and learning rate through validation. That is why the neural network is validated and tested in Section 2.6.2.2.

2.6.2.2 Validating and Testing the Model I use the first 9 sets of the recommendation data in Table 2.4 as the training data, the second 9 sets of the recommendation data as the validation data to modify the neural network to ensure higher accuracy, and the last 12 sets of the recommendation data as the testing data to test the accuracy of the model. I first tested the existence of neural network overfit. In the process of neural network training, I applied the neural network weights to both the testing data and the validation data sets and compute the estimation errors. Figure 2.11(a) shows that the estimation errors of the training data decrease monotonically with the iterations. Figure 2.11(b) shows that the estimation errors of validation data decrease first then increase dramatically. The dramatic increase of estimation errors in validation data is known as the problem of overfitting. This means the neural network fits the unique characteristics of the training data, instead of fitting the general properties.

To prevent overfitting, I applied validation techniques to the back propagation process. The back propagation process is terminated when the estimation errors of the validation data increases significantly. In other words, before the neural network fits the specific characteristics of the training data, the training process finishes and returns a neural network of higher accuracy on both the training data and validation data. Validation could also be done by adjusting the neural network parameters, for example,

Table 2.4. Training data of the recommendation neural network model

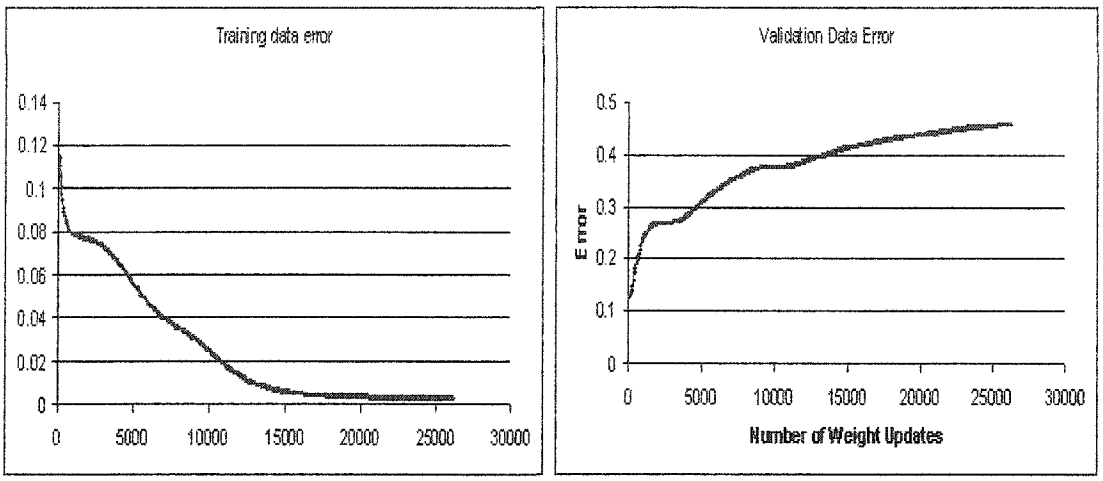
Object	Qualified Recommender QR_e				Requester
	qr_1	qr_2	qr_3	qr_4	r_0
p_1	0.8	0.8	0.4	1	0.9
p_2	0.6	0.9	0.7	1	0.9
p_3	0.85	0.9	0.3	1	0.95
p_4	0.8	0.8	1	0.9	0.8
p_5	0.7	0.85	0.4	0	0.7
p_6	0.5	0.9	0.9	0.2	0.65
p_7	0.6	0.95	1	0.2	0.85
p_8	0.9	0.9	1	0.8	0.85
p_9	0.85	0.98	0.2	0.8	0.6
p_{10}	0.8	0.9	0.2	0.1	0.6
p_{11}	0.75	0.95	0.5	0.5	0.8
p_{12}	0.85	0.88	0.3	0.3	0.85
p_{13}	0.9	0.9	0.3	0.3	0.65
p_{14}	0.7	0.85	1	0.8	0.85
p_{15}	0.7	0.9	1	0.8	0.7
p_{16}	0.7	0.8	1	0.3	0.7
p_{17}	0.8	0.8	1	0.5	0.7
p_{18}	0.8	0.8	1	0.1	0.75
p_{19}	0.85	0.85	1	0.5	0.7
p_{20}	0.75	0.85	0.8	1	0.65
p_{21}	0.8	0.8	0.5	1	0.6
p_{22}	0.8	0.8	0.4	0.9	0.6
p_{23}	0.7	0.9	0.4	1	0.65
p_{24}	0.75	0.85	0.6	0.9	0.7
p_{25}	0.8	0.9	0.6	0.9	0.7
p_{26}	0.8	0.8	0.6	1	0.75
p_{27}	0.75	0.85	0.4	1	0.8
p_{28}	0.85	0.85	0.8	0.9	0.7
p_{29}	0.8	0.8	0.4	1	0.7
p_{30}	0.8	0.8	0.5	0.8	0.7

Table 2.5: Testing of the recommendation neural network model.
Training data sets: 15; Testing data sets: 15.

Learning rate	Training			Testing	
	convergence	*err*	correctness	*err*	correctness
0.59	174155	0.0155	14	0.1964	11
0.46	214952	0.0154	14	0.1929	11
0.42	98945	0.0300	14	0.2352	9
0.51	90875	0.0299	14	0.2326	9
0.59	286127	0.0271	14	0.2265	8
0.55	94131	0.0295	14	0.2296	9
0.52	87948	0.0299	14	0.2315	9
0.59	173454	0.0154	14	0.1915	11
0.52	80557	0.0300	14	0.2337	9
0.53	90781	0.0299	14	0.2320	9
0.48	70000	0.0210	14	0.2403	8
0.45	100160	0.0299	14	0.2337	9
0.54	183942	0.0155	14	0.1938	11
0.4	249219	0.0155	14	0.1947	11
0.59	76359	0.0300	14	0.2330	9
0.42	200928	0.0289	14	0.2222	9
0.48	210783	0.0155	14	0.1957	11
0.4	127312	0.0296	14	0.2298	9
0.45	127739	0.0295	14	0.2300	9

the learning rate and the initial weights of the neural network. The reason is that the searching space may have multiple local minimums. Through parameter adjustment, we are most likely to have high accuracy and fast convergence speed, or be able to obtain the global minimum. In the experiments, I combine both parameter adjustment and overfitting prevention in the validation process. I then tested the accuracy of the validated neural network by the test data. Table 2.6 shows the training, validation and testing results when the error threshold is set to be 0.15. By average, the estimation correctness over the entire data sets is 74.7%. The optimal learning rate is 0.52, and the estimation correctness is 80%. Figure 2.12 shows the estimation correctness of the testing data after validation at various estimation error thresholds. When the estimation error threshold is

0.15, the estimation correctness of the testing data is 83.3%. When the estimation error threshold is above 0.15, the estimation correctness is 100%.



(a) Training Data Error (b)Validation Data Error

Figure 2.11. Evidence of overfitting in training recommendation neural network.

Table 2.6: Validation and testing of the recommendation neural network model. Training data: 9 sets; Validation data: 9 sets; Testing data: 12 sets. it: iteration; corr: correctness; corr pctg: correctness percentage

Learning rate	Training			Validation		Testing		Total		
	it.	*err*	corr	*err*	corr	*err*	corr	*err*	corr	corr pctg
0.46	21	0.118	7	0.124	6	0.094	9	0.336	22	73.3%
0.41	24	0.117	7	0.125	6	0.094	9	0.336	22	73.3%
0.41	*23*	*0.120*	*7*	*0.124*	*6*	*0.091*	*10*	*0.335*	*23*	*76.7%*
0.49	20	0.117	8	0.124	6	0.095	8	0.336	22	73.3%
0.4	25	0.118	7	0.125	6	0.093	9	0.336	22	73.3%
0.52	**17**	**0.121**	**7**	**0.122**	**7**	**0.092**	**10**	**0.335**	**24**	**80.0%**
0.59	16	0.119	8	0.123	6	0.094	8	0.337	22	73.3%
0.41	*24*	*0.118*	*8*	*0.122*	*7*	*0.094*	*8*	*0.334*	*23*	*76.7%*
0.41	24	0.117	7	0.126	6	0.094	9	0.336	22	73.3%
0.51	19	0.118	8	0.124	6	0.094	8	0.336	22	73.3%
0.48	19	0.119	7	0.125	6	0.093	9	0.337	22	73.3%

2.7 Related Work

Trust and reputation were first used in e-commerce systems (Schafer, Konstan, and Riedl 1999; Vassileva, Breban, and Horsch 2002) to encourage transactions between strangers. The use of trust and reputation has extended to areas of distributed computing (Azzedin and Maheswaran 2002), file sharing P2P system (Cornelli, Damiani, and Samarati 2002) and information filtering (Montaner, L´opez, and Rosa 2002).

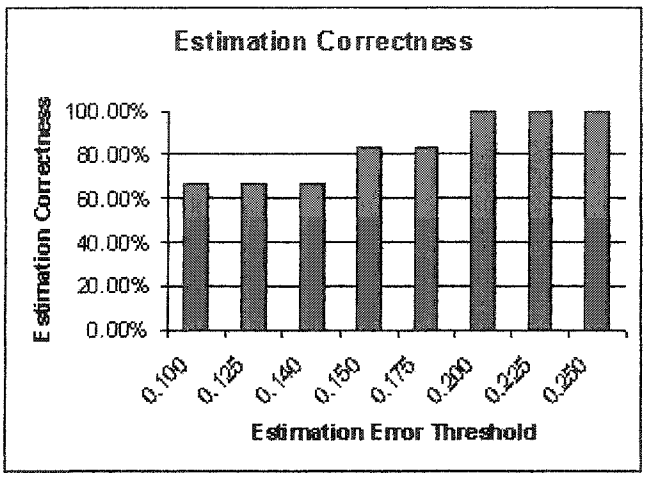


Figure 2.12. Estimation correctness of the testing data after validation.

In Amazon and eBay's trust systems, a user's reputation is a function of cumulative ratings by his trading partners. SPORAS system (Zacharia and Mae 1999) introduces raters' reputations, variation of the new ratings, and deviations of the new ratings from an expectation as additional factors to derive a user's trust. These models are based on a centralized rating system where users' reputations are globally accessible. Song and Phoha (2004a) has developed a global reputation model in a distributed trust management system, where users' multiple local reputations are aggregated.

However, in a system where no central or global reputation mechanism is available, an alternative is to aggregate recommendations. Current recommendation trust models provide various mechanisms to select recommenders, and to aggregate the selected recommenders' trust evaluations. HISTO model (Zacharia and Mae 1999) is based on the assumption that a requester trusts some agents more than others. It uses breadth first search algorithm to find all the referral chains within a certain length limit and branching size. The recommended trust is an aggregation of the selected recommenders' reputations, their recommendations and the deviations among the recommendations.

Similar to HISTO model, Riggs and Wilensky (2001) developed a quality filtering trust model. The model rates reviewers and applies the quality of the reviewers into merits of the reviewed papers. The idea is that a reviewer is reliable if he/she consistently ranks papers near their ultimate average. Thus, a reliable reviewer's rate is the actual rate of the paper. Mui, Mohtashemi, and Halberstadt (2002) proposed a recommendation trust model based on Bayesian probability theory. This model studies a parallel referral network between a requester and a party of interest. It assumes that trust is the probability that a user will be honest for the next online trade. The recommended trust of the party of interest is a weighted average of referral chains' trust values. A referral chain's trust is measured by successive applications of Bayesian probability of any two contiguous references along the chain. The weight of a referral chain is the productivity of each edge's weight. Chernoff Bound is used as a reliability measurement for trust information gathered along each chain.

Yu and Singh (2001) presented an evidence model to evaluate recommendation trust. It applies Dempster-Shafer theory to multiple witnesses. The witnesses are recursively selected from referrers' neighborhood. However, this model requires explicit ratings of each reference.

In order to avoid continuous and explicit ratings of references, Pujol, Sang, and Halberstadt (2002) proposed a recommendation trust model based on social network topology. Their model applies noderanking algorithm to infer a node's reputation. The noderanking algorithm is similar to the ranking algorithms for Web pages based on Web topology.

However, there are a few caveats to the approaches mentioned above. First, different users may arrive at significantly varying estimates of the performance of the same service provider. Second, different users may be able to observe different instances of the performance of a given service provider. Third, deceptive recommendations may exist. More sophisticated collaborative trust models are developed by Azzedin and Maheswaran 2002; Daniani, Vimercati, Paraboschi, Samarati, and Violante 2002; Gupta, Judge, and Ammar 2003; Kamvar, Schlosser, and Garcia-Molina 2003; Schafer, Konstan, and J.Riedl 2002; Wang and Vassileva 2003; and Yu and Singh 2003.

Wang and Vassileva (2003) applied Naive Bayesian network to recommendation trust since trust is multifaceted. Their trust model provides a differentiated trust in different aspects of peers' trust behaviors in file sharing peer-to-peer applications. It can be used to solve the first problem, i.e., different estimation process of the same online service.

Kamvar, Schlosser, and Garcia-Molina (2003) focus on detecting malicious file providers in peer-to-peer file sharing networks. Their model derives agents' global reputations from the distributed local trust values. The model is built on the notion of transitive trust. A peer trusts authentic file providers as well as their recommendations. A major issue of applying this model is to find pre-trusted peers that guarantee convergence of the algorithm and avoid malicious collectives; however, this has not been investigated.

Yu and Singh (2003) applied weighted majority technique to belief function and belief propagation. The model detects deceptions by decreasing the weights assigned to unsuccessful recommenders. Gradually, the weights assigned to successful recommenders are increased and the weights assigned to unsuccessful or deceptive recommenders are decreased. The conceived recommendation trust is a weighted average of all the recommendations. The model fails to discuss nonlinear aggregation of recommendations.

This work is designed to derive trust opinions from multiple heterogeneous recommendations of different estimation processes and different trust evaluation models with or without deceptions. It concentrates on the algorithm of selecting *qualified recommenders* and the algorithm of aggregating their recommendations both linearly and nonlinearly. The approach is a novel application of neural network techniques in recommendation trust management.

2.8 Conclusion and Future Work

A multiagent system consists of a network of heterogeneous peers whose trust evaluation standards may differ. Additionally, reliability of a recommendation lies in a recommender's expertise and the authenticity of the recommendation. This chapter

designs an adaptive recommendation trust model that filters different recommendation opinions and trust standards. The model derives a trust value based on an agent's own trust standards and thus makes trust decision easier. The model is designed to adapt to various changes, such as changes in trust behaviors, trust evaluations, agents' expertise and trust accuracy requirement. The neural-network based recommendation trust model has the following properties: (1) fast speed with high accuracy, (2) capability of non-linear aggregation of heterogeneous agent's recommendations, (3) capability of catching hidden variables in the hidden layers of the model, (4) robustness to noises in the training data, and (5) adaptivity.

The model is based on the assumption that agents are willing to exchange trust opinions. This is beneficial to both the requester and the recommender. However, the performance of the model will be affected if there are frequent emergence of new agents and exit of old agents. In the future, I plan to build an automatic mechanism to monitor the changes and adjust the model.

CHAPTER 3

HIDDEN MARKOV MODEL BASED REPUTATION MODEL

There is limited research on evaluating an agent's reputation as a trust recommender. A key challenge is that a recommender's reputation is affected by both the recommender's trustworthiness and the recommender's expertise, including the recommender's trust knowledge of others and the reliability of the recommender's trust evaluation models. In this chapter, I develop a Hidden Markov Model (HMM) based approach to measure an agent's reputation as a recommender.

3.1 Background and Motivation

Evaluation of recommenders' reputations is important in e-commerce and P2P networks (Resnick and Zeckhauser 2000; Sarwar, Karypis, Konstan, and Ridel 2000). Online users rely on recommendations to familiarize themselves with new services and products. Reliability of recommendations depends on the reputation of the recommenders, including both their trustworthiness and their expertise.

Many evaluation models have been developed to obtain the reputation of a party of interest from recommendations (Kamvar, Schlosser, and Garcia-Molina 2003; Pujol, Sang, and Halberstadt 2002; Riggs and Wilensky 2001; Song and Phoha 2004b; Song,

Phoha, and Xu 2004; Zacharia and Mae 1999). However, very limited research has explicitly studied the reputation of an agent as a trust recommender.

Yu and Singh (2001) apply Dempster-Shafer belief theory to aggregate recommendations on a referral chain. "*Their model requires explicit expressions of the recommenders' reputations on the chain. However, it does not address how to obtain the recommenders' reputations*" (Song, Phoha and Xu 2004c).

Mui, Mohtashemi, and Halberstadt (2002) use Bayesian theory to derive recommendation trust of a party of interest. In general, the model evaluates how agent A should trust agent C, given agent *A* trusts *B* to a certain degree, and agent *B* trusts *C* to another certain degree. Agent *B*'s reputation is interpreted as *A*'s trust opinion of *B* through their direct interactions rather than *B*'s reputation as a recommender. The model is limited to parallel referral networks only. Parallel referral networks are those with no shared nodes or paths.

Song and Phoha (2004a) use neural network techniques in evaluating the reputation of a party of interest with multiple local reputations in a distributed system. Their model hides the recommenders' (local agents') reputations in the neural network. However, the weights of the neural network connections fail to give explicit evaluations of the recommenders' reputations.

Pujol, Sang, and Halberstadt (2002) develop noderanking algorithm to infer a node within a social network. The noderanking algorithm is similar to the ranking algorithms for web pages based on web topology. Each node has an authority and a part of this authority is propagated to the out-nodes via out-edges. This model requires

measuring social relationships, such as email traffic, sharing of physical resources, and hierarchical structure among the nodes.

My approach explicitly evaluates a recommender's reputation in terms of the recommender's expertise and the trustworthiness of the recommendations. The idea is that all agents learn from experiences and always choose the most reliable recommenders at the current time. As time goes on, those frequently selected recommenders must be the trustworthy experts in the evaluations of other agents' trust (context dependent). The challenge then becomes: (1) how to find a most reliable recommender, and (2) how to model all the chained recommendation events (leading to the reliable recommender) and their transitions. Obviously, based on the observation of reliable recommendation events, the transition probability from Requester (or intermediate recommender) A to Recommender B measures B's reputation as a recommender in the eyes of A.

To address the challenges, I first develop an ordered depth first search algorithm with thresholds (ODFST) for the most reliable recommenders. I assume that a requester always searches for the most reliable recommender for the time being and get the recommended trust value of the party of interest. Second, based on the observations of repeated references of most reliable recommenders, I build a Hidden Markov Model (HMM) to model recommendation transitions. The transition probabilities are actually how reliable the recommenders are in the eye of the requesters (or intermediate recommenders). An attractive feature of the model is that it derives the learning speed of a requester as well as the recommenders' convergent reputations. The learning speed is measured by the minimum number of recommendation queries an agent has made. The learning speed can be used to measure the reliability of reputation evaluations when

sparse recommendation events are available. To our belief, the model can find other applications, such as identifying optimal recommendation paths and locating optimal file servers in P2P networks.

3.2 Development of Recommender's Reputation

Current recommendation trust models adopt different criteria of selecting recommenders, for example, depth first search (Song and Phoha 2004b), breadth first search (Zacharia and Mae 1999) and nearest neighbors (Montaner, L´opez, and Rosa 2002). I introduce an ordered depth first search with thresholds (ODFST). First, a requester always chooses the most reliable recommender for the time being. If the selected recommender can not provide recommendation for the current request, two scenarios may occur. One is that the recommender becomes a requester and forwards the trust request to the most reliable recommender he/she knows of. This may result in either a fruitless or a successful recommendation chain. If it comes up with a fruitless recommendation chain, the second scenario occurs, i.e., the requester selects the next most reliable recommender he knows of. The requester sets a maximum depth of recommendation chains as well as reputation thresholds for recommenders at each depth level. While searching for a most reliable recommender, if the recommendation depth is beyond the depth limit, the search retrieves back along the chain and looks for the next most reliable recommender. If the next most reliable recommender does not have a reputation higher than the preset threshold at that depth level, the search further retrieves back along the chain. The purpose of doing so is to guarantee the reliability of recommendation chains. The process continues till a recommender is found to provide

the trust opinion of the party of interest. Algorithm 3.1 and 3.2 show the details. In ODFST-visit (Algorithm 3.2), backtracks occur when (1) the depth is beyond the depth limit, (2) all recommenders' reputations are less than the threshold at that depth level, and (3) an intermediate recommender ends up with no successful recommendations.

Algorithm 3.1. ODFST (Requester)

```
1: length ← 0
2: for each r ∈ R of the Requester's recommender set (initialization)
3:       color[r] ← white (white: unprocessed)
4:       parent[r] ← nil
5: end for
6: time ← 0
7: for each r ∈ R (search)
8:       if color[r] = white
9:             length ← length + 1
10:            ODFST-Visit (r, Requester)
11:      end if
12: end for
```

Algorithm 3.2. ODFST-Visit (r, Requester)

```
1: color[r] ← gray (gray: in process)
2: time ← time + 1
3: d[r] ← time (set discover time)
4: if r has direct experience or length > limit or R_r < Threshold[length]
(r is a recommender or the recommendation chain has reached length limit or r's
trust value is less than the threshold at the depth level)
5:       color[r] ← black (black: finished)
6:       parent[r] ← Requester
7:       f[r] ← time ← time + 1 (finish time)
8:       length ← length-1 (backtrack one level)
9:       return
10: end if
11: for each recommender i in r's ordered recommender set
12:      if color[i] = white
13:            parent[i] ← r
14:            length ← length + 1
15:            ODFST-Visit(i, r)
16:      end if
17: end for
18: color[r] ← black
19: f[r] ← time ← time + 1 (finish time)
```

Initially, when recommenders' reputations are not available, a requester sends his trust request of a party of interest to all recommenders. Based on the recommended trust value of the party of interest and the actual transaction experiences, the requester updates the recommenders' reputations. A recommender's reputation can be updated by a weighted average method, such as:

$$V_R^i = w_0^i V_R^i + (1 - w_0^i)(1 - |v_r - v|) \tag{3.1}$$

Where V_R^i stands for recommender R's reputation evaluated by the direct requester i. Initially, $V_R^i = 1$. v_r and v stand for the recommended trust value by R and the trust rating by requester i based on the actual transaction respectively. w_0^i is the weight, reflecting reputation change rate per update. w_0^i varies with requesters. In general, a requester updates recommender R's reputations as:

$$V_R^i = f_i(V_R^i, v_r, v) \tag{3.2}$$

Where f_i is the reputation update function and does not necessarily to be a weighted average model. If there are several recommenders of the highest reputation, the requester sends trust queries to one of them and updates the recommender's reputation based on Equation 3.2. Also rating v is forwarded along the recommendation chain, such that every intermediate recommender (a requester as well) can update the final recommender's reputation. Thus, through repeated references, requesters get to know their recommenders and the recommenders build up their reputations. Figure 3.1 shows an example of how ODFST works. An arrow line represents a recommendation event. The number on the arrow line is the event's sequential order on the recommendation chain. The edge value is the recommender's reputation evaluated by the requester (or intermediate recommender).

3.3 HMM Based Recommender's Reputation Model

Intermediate recommenders in a recommendation chain may not be obligated to provide explicit reputation evaluations of the next recommenders. What we can observe are a series of recommendation events. Based on the observations and the assumptions that requesters always choose the most reliable recommenders at the current time, I develop an HMM-based reputation model to evaluate recommenders. The general idea is, the more often a recommender is referenced, the higher his/her reputation is. Since if a recommender gives false or inaccurate trust information, his/her reputation value would decrease. The recommender may not be referenced next time if he is not the most reliable recommender among other competing recommenders.

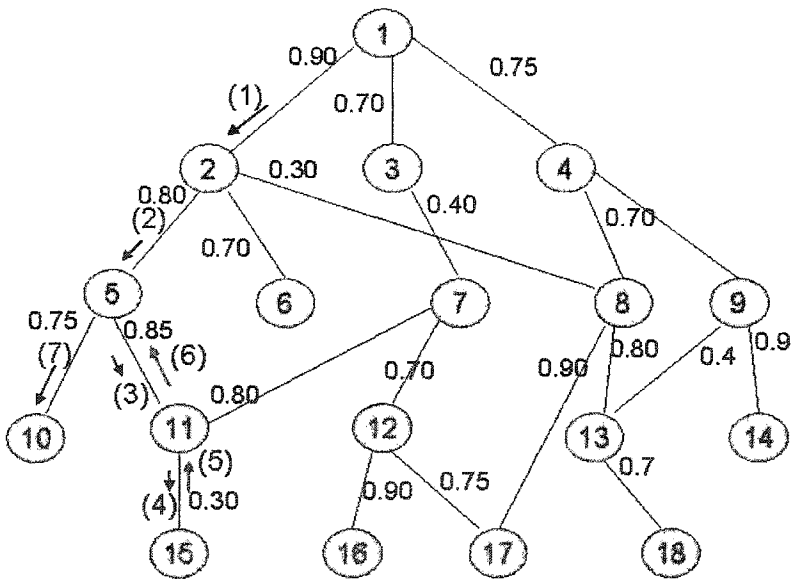


Figure 3.1: Ordered depth first search with thresholds. The nodes with lower values are requester or intermediate requesters. The leaf nodes are all possible final recommenders. The numbered arrow lines show the search process. The edge value is the recommender's reputation evaluated by the requester (or intermediate requester). The most reliable recommendation chain is: 1 → 2 → 5 → 10.

3.3.1 HMM Model of Recommendations

A recommendation HMM is a state diagram. The starting state represents a requester. All the other states are possible recommenders. When a trust query is forwarded from one recommender to another, a recommendation event occurs. To build an HMM (Rabiner 1989) is to find the three parameters A, B, and Φ. $A = \{a_{ij}\}$ is the state transition probability matrix. Each element $a_{ij} = p(q_j|q_i)$ is the probability that agent i forwards the trust query to recommender j, given that agent i is selected as an intermediate recommender (a direct requester for the current recommendation event). $B = \{b_{ij}\}$ is the recommendation event distribution matrix. Each element $b_{ij} = p(\sigma_{ij}|q_i)$ is the probability that recommendation event σ_{ij} occurs given that agent i is the requester. $\Phi = \{\varphi_i\}$ is the initial trust query distribution. A recommendation HMM can be uniquely identified as $\lambda = (A, B, \Phi)$. In recommendation HMM, matrix A and B's relationship is shown in Equation 3.3. Since we study trust queries from Agent 1 (the requester) only, $\varphi_i = 1$ if $i = 1$, otherwise it is zero.

3.3.2 Extended Baum-Welch Algorithm (EBW)

Baum-Welch algorithm (BW) (Baum, Petrie, Soules, and Weiss 1970; Rabiner 1989) is used to model an HMM λ The basic idea of BW algorithm is to repeatedly produce a better HMM model λ_{t+1} such that $P(O|\lambda_{t+1}) \geq P(O|\lambda_t)$ until λ_t converges, where $O = \{o_1, o_2, \ldots, o_T\}$ is a recommendation chain of length T and λ_t is the HMM model obtained at the t^{th} iteration of BW algorithm. At each iteration, matrix B is improved, and so is matrix A. The relationship between A and B is:

$$a_{ij} = b_i(\sigma_{ij}) \tag{3.3}$$

Where σ_{ij} stands for the observed event that requester (or intermediate recommender) i sends the trust query to recommender j.

Let $\alpha_t(i)$ be the probability of the partial observation sequence, $o_1, o_2, \ldots, o_t$, and at time t the recommender is i, given a model λ $\alpha_t(i)$ can be computed iteratively as:

$$\begin{aligned} \alpha_1(i) &= \phi_i b_i(o_1) \\ \alpha_{t+1}(j) &= [\sum_{i=1}^{n} \alpha_t(i) a_{ij}] b_j(o_{t+1}) \end{aligned} \tag{3.4}$$

where $1 \leq j \leq n$ and $1 \leq t \leq T-1$. Let $\beta_t(i)$ be the probability of the partial observation sequence from $t+1$ to the end, given model λ and at time t the recommender is i. $\beta_t(i)$ is solved inductively as:

$$\begin{aligned} \beta_T(i) &= 1 \\ \beta_t(i) &= \sum_{j=1}^{n} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j) \end{aligned} \tag{3.5}$$

Where $t = T-1, T-2, \ldots, 1$ and $1 \leq i \leq n$. T is the recommendation chain length. n is the total number of agents in the recommendation network. Let $\gamma_t(j)$ be the forward backward probability, representing the probability that the trust query is forwarded to recommender j at time t given the observed recommendation chain and the model λ $\gamma_t(j)$ is computed as:

$$\gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{\sum_{i=1}^{n} \alpha_t(i)\beta_t(i)} \tag{3.6}$$

Matrix B can be calculated as:

$$b_j(\sigma_{jk}) = \frac{\sum_{t=1 \text{ and } o_t=\sigma_{jk}}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \tag{3.7}$$

Where o_t stands for the observed recommendation event at time t, and σ_{jk} stands for the reference event from recommender j to recommender k. The initial HMM λ_0 is set as:

$$\phi_i = \begin{cases} 1, i = 1 \\ 0, i \neq 1 \end{cases}$$
$$b_{ij} = \begin{cases} \frac{1}{V_i}, 1 < j < V_i \\ 0, \text{otherwise} \end{cases} \quad (3.8)$$
$$a_{ij} = b_i(\sigma_{ij}) = b_{ij}$$

Where V_i is the number of all possible recommenders of requester i (or intermediate recommender).

However, there are two limitations of the BW algorithm. One is that BW algorithm only takes one observation sequence. Given multiple observation sequences, i.e., recommendation chains, we are unable to achieve a global maximum. Additionally, BW algorithm does not allow $\sum_j a_{ij} = 0$ for any i. This is unreasonable. Due to the thresholds imposed on the ordered depth first search for the most reliable recommenders, some recommenders may never be referenced.

Extended Baum-Welch algorithm is developed to derive a recommendation HMM. To allow for unreachable states in a HMM, instead of requiring $\sum_j b_{ij} = 1$, we modify the constraint as:

$$\sum_j b_{ij} < 1 - \theta \quad (3.9)$$

where $0 < \theta << 1$ is a constant, standing for the probability of any event that is not modeled in the HMM.

To avoid local maximum, we derive HMM λ_i, $i = 1, 2, \cdots, N$, for each of the N most reliable recommendation chains. $\overline{\lambda_1}$ is the mean value of the N HMMs, i.e., $\overline{\lambda_1} = mean(\lambda_1, \lambda_2, \ldots, \lambda_N)$. Then we increase the observations by another N recommendation chains. Similarly, we obtain $\overline{\lambda_2}$ based on the total $2N$ recommendation

chains. We keep increasing the observation size of the recommendation chains till $\overline{\lambda}_i = mean(\lambda_1, \lambda_2, \ldots, \lambda_{iN})$ converges. The convergence condition is that the Euclidean distance between $\overline{B}_i$ and $\overline{B}_{i-1}$ is less than a predefined threshold ε, i.e.,

$$d(\overline{\lambda}_i, \overline{\lambda}_{i-1}) = \sqrt{\frac{1}{n}\sum_{j=1}^{n}\sum_{k=1}^{n} \| \overline{b}_{jk}^{\,i} - \overline{b}_{jk}^{\,i-1} \|^2} \tag{3.10}$$

Where n stands for the total number of recommenders. The approximated global maximum Matrix B is:

$$B_{global} = \overline{B}_i = \frac{\sum_{k=1}^{iN} \overline{B}_k}{iN} \tag{3.11}$$

Global maximum approximation of Matrix A can be obtained by Equation 3.3. a_{ij} is the transition probability that requester i forwards a trust request to recommender j. a_{ij}, in effect, measures recommender j's reputation in the eye of agent i. The relative reputation evaluation includes both recommender j's expertise and trustworthiness.

Figure 3.2 shows how EBW works. Algorithm 3.3 describes a step-by-step HMM-based approach using the EBW algorithm to derive agents' reputations as recommenders.

Algorithm 3.3. HMM-Based Recommender Reputation Model

initialize HMM λ_0 (Equation 3.8);
set $\overline{\lambda}_0 = \lambda_0$;
i = 0;
Do
    increase i by 1;
    increase recommendation chains by N;
    for each of iN recommendation chains
        generate a HMM λ_i by running the
        BW algorithm (Equation 3.3 and 3.7);
    Calculate $\overline{\lambda}_i$;
    calculate $d(\overline{\lambda}_i, \overline{\lambda}_{i-1})$ (Equation 3.10);
until $d(\overline{\lambda}_i, \overline{\lambda}_{i-1}) < \varepsilon$

calculate B by Equation 3.11;
set A by Equation 3.3;
return A as the recommender reputation matrix;

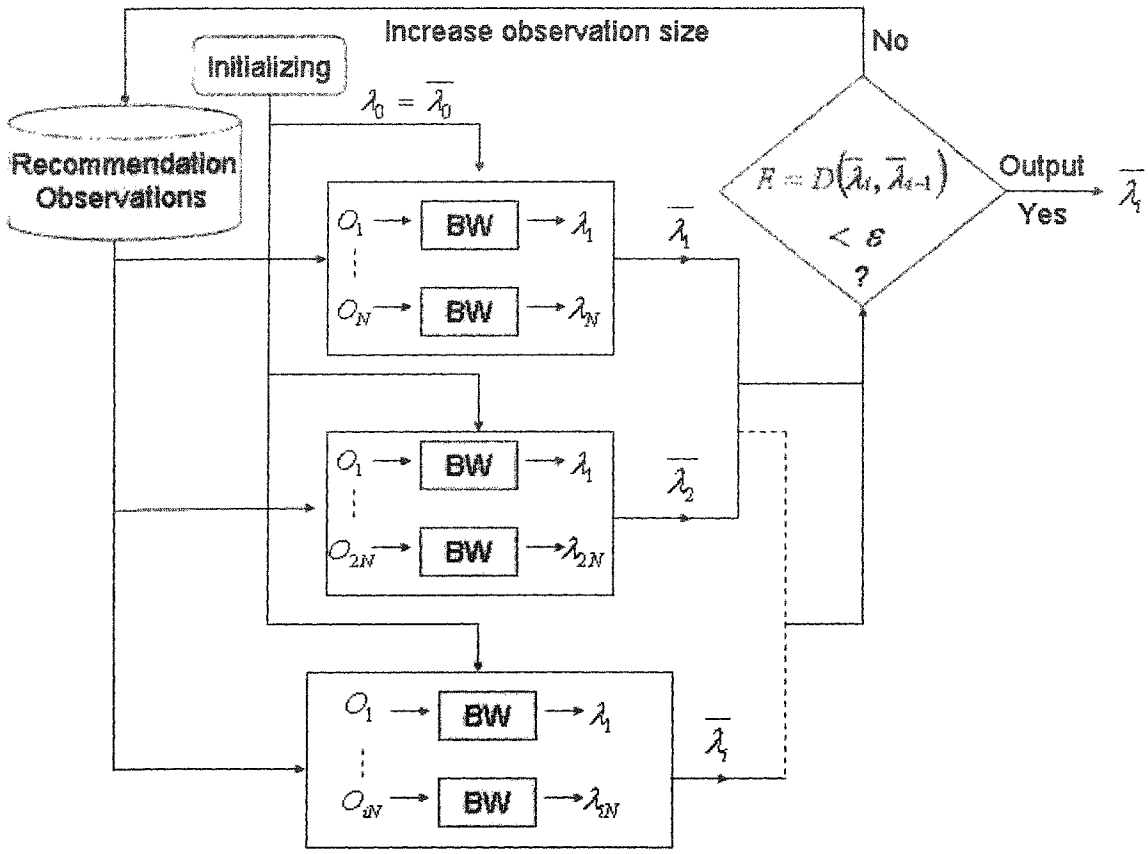


Figure 3.2: Extended Baum-Welch algorithm. The algorithm derives a mean HMM based on multiple recommendation chains. The number of recommendation chains keeps increasing till the mean HMM converges.

3.4 Experiments

Based on the simulation results of the most reliable recommendation chains, I ran the EBW algorithm to obtain the HMM λ, where Matrix A represents recommenders' reputations.

3.4.1 Simulation of the Most Reliable Recommendation Chains

I simulated recommendation events (context dependent) based on a recommendation network of Figure 3.3. ODFST is used to derive the most reliable

recommendation chains. Recommendation chain length is limited to 6. Reputation thresholds at different depth levels are all set to be 0.5. Recommenders' reputations were updated by the weighted average model as in Equation 3.1, where $w_0 = \frac{1}{2}$ and the initial recommenders' reputations are all set to be 1. Notice that the updated reputations are only known to the requester himself.

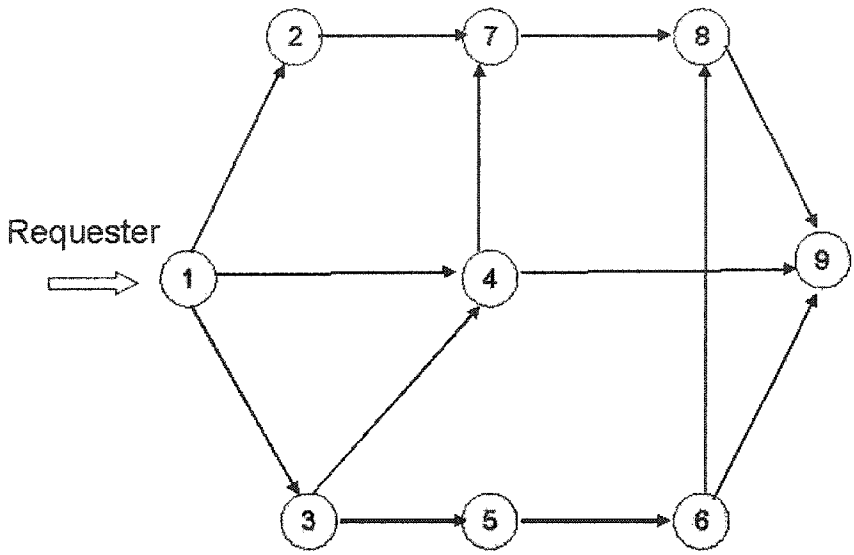


Figure 3.3: Recommendation network. The starting state is the requester. All other states are possible recommenders. One state transits to another when a recommendation event occurs.

Two simulation sets of recommendation events were generated. Simulation 1 is based on the recommendation network given in Figure 3.4(a). In this simulation, Agent 1 randomly (a uniform distribution) selects an agent and sends the trust query of the selected agent to his/her recommenders. We assume all recommenders are trustful and their trust evaluations of the queried agents are exactly how the requester (the querying agent) perceives those agents, i.e., $v_r = v$ (see Section 3.2); however, the recommenders may have different knowledge area. For example, recommender i evaluates 60% of the agents (context dependent) while recommender j evaluates the other 40% of the agents.

In Simulation 2, I assume some recommenders are more knowledgeable of agents' reputations than others as shown in Figure 3.4(b). For simplicity, I assume that all recommendations provided by the recommenders satisfy $v_r = v$. In theory, false recommendations are also applicable.

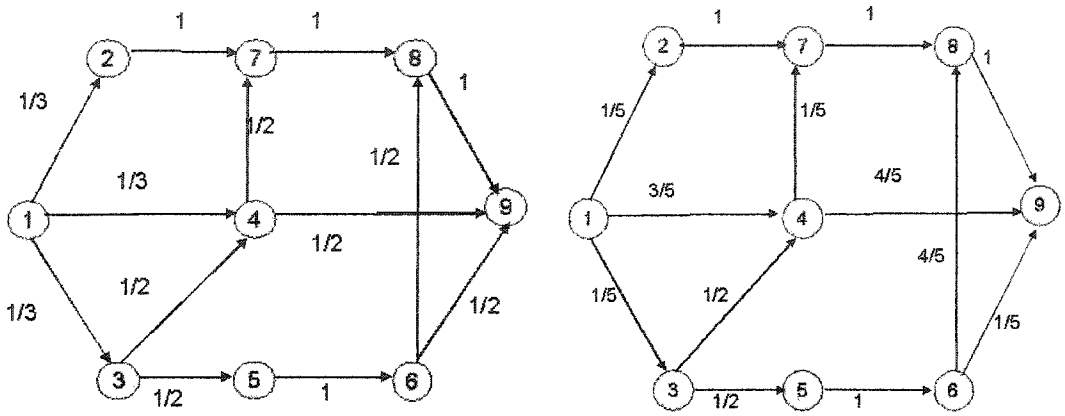


Figure 3.4. Recommendation event distributions of the two simulations.

A total of 9500 recommendation chains were generated for each simulation. Of both simulations, all recommenders' reputations are evaluated by the weighted average reputation model (see Equation 3.1). Obviously, the weighted average model does not catch all the factors affecting the recommenders' reputations, for example, their knowledge size of other agents' trust. This is another reason that I introduce the HMM-based reputation model.

3.4.2 Reliability of HMM-Based Reputation Model

Based on the simulated 9500 recommendation chains in Simulation 1, recommender reputation matrix A_1 obtained via EBW is as follows:

$$A_1 = \begin{pmatrix} 0 & .325 & .329 & .326 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & .98 & 0 & 0 \\ 0 & 0 & 0 & .493 & .487 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & .499 & 0 & .481 \\ 0 & 0 & 0 & 0 & 0 & .98 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & .499 & .481 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & .98 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & .98 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Recommender reputation matrix A_2 over another 9500 simulated recommendation chains in Simulation 2 is obtained as:

$$A_2 = \begin{pmatrix} 0 & .195 & .585 & .2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & .98 & 0 & 0 & 0 & .98 & 0 & 0 \\ 0 & 0 & 0 & .481 & .499 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & .198 & 0 & .782 \\ 0 & 0 & 0 & 0 & 0 & .98 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & .764 & .216 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & .98 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & .98 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Matrix A_1 tells us that Agent 1 has three recommenders, Agent 2, Agent 3 and Agent 4. The recommendation transition probabilities to those three agents are 0.325, 0.329 and 0.326 respectively. This means that Agent 2, 3 and 4 are equally reliable and knowledgeable of agents' reputations (context dependent). Agent 2 has one recommender only, Agent 7. As long as Agent 7's reputation is higher than the predefined threshold, Agent 2 will always forward the trust query to Agent 7. In our simulation, Agent 7's reputation is 1 (updated by Equation 3.1). Therefore, a_{27} = 0.98 fits our simulation data (ideally, a_{27} should be 1). In matrix A_2, Agent 6 ranks Agent 8 and Agent 9's reputation as 0.764 and 0.216 respectively, which indicates that Agent 8 is more trustful or more knowledgeable as a recommender.

Figure 3.5 and 3.6 show the convergence process of obtaining the HMMs of Simulation 1 and Simulation 2 respectively. The convergence speed accelerated dramatically for the first 1500 recommendation chains in both cases. This indicates that an HMM based on less than 1500 recommendation chains is not stable. Correspondingly, the derived recommenders' reputations may not be reliable. When the observation size of recommendation chains increased to 3500, the HMMs started to converge. In fact, recommendation trust matrix A_2 obtained from the 9500 recommendation chains is insignificantly different from the recommendation trust matrix obtained from the 3500 recommendation chains.

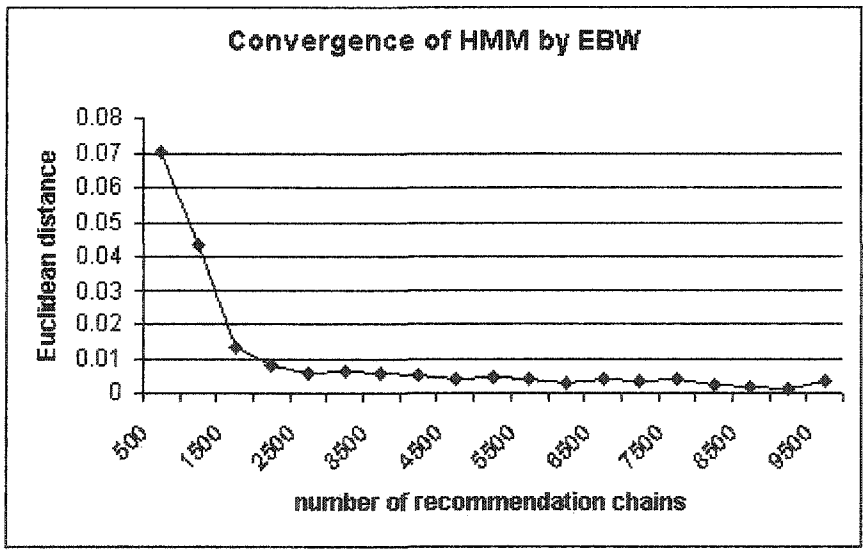


Figure 3.5: Convergence of HMM obtained from EBW based on recommendation chains generated in Simulation 1.

I also analyze the accuracy of the HMM-based reputation model by comparing the derived reputation matrix A via the EBW algorithm with the simulation data. Figure 3.7 shows the estimation error by comparing A_1 with the Simulation 1 data set in Figure 3.4(a). Figure 3.8 displays the estimation error by comparing A_2 with the generated Simulation 2 data set in Figure 3.4(b). The error of the worst case is less than 10%. The

result shows that the reputation value obtained from the HMM model is close to the true value. So the HMM based approach via the EBW algorithm is a reliable method to generate the reputation of recommenders for a given recommendation network.

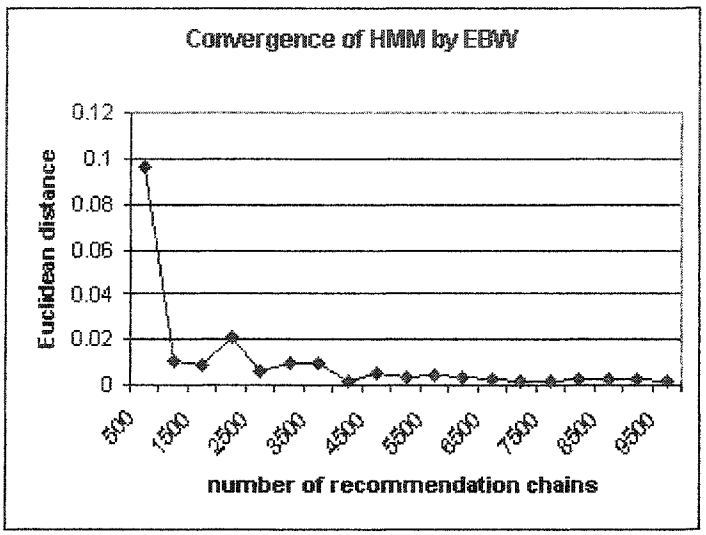


Figure 3.6: Convergence of HMM obtained from EBW based on recommendation chains generated in Simulation 2.

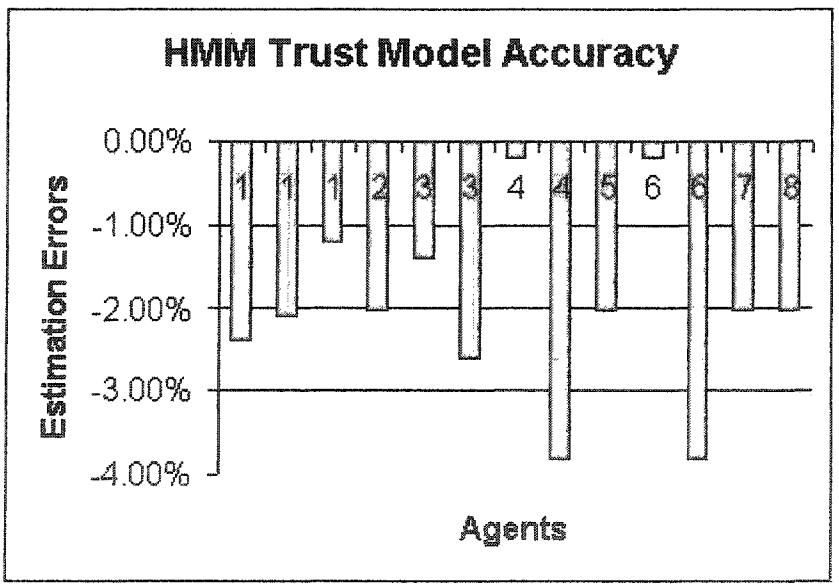


Figure 3.7: Error analysis of recommender's reputation obtained through HMM using Simulation 1 data set.

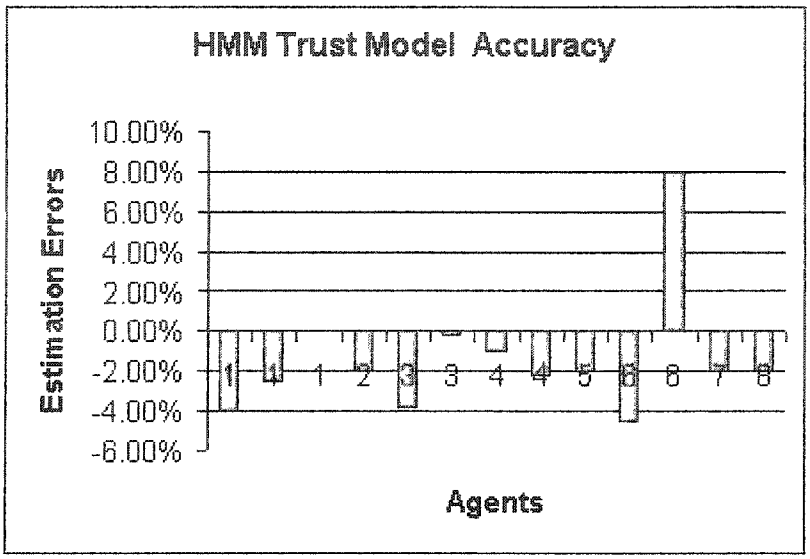


Figure 3.8: Error analysis of recommender's reputation obtained through HMM using Simulation 2 data set.

3.5 Conclusions

This chapter presents an HMM-based technique to model recommendation networks in distributed trust systems. The EBW algorithm is developed to derive recommenders' reputations. We uniquely identify a recommendation network by its initial states, state transition probability matrix, and event probability matrix, i.e., an HMM. Based on the assumption that agents are rational and they forward trust requests only to trustworthy recommenders, the state transition matrix, in effect, represents recommenders' reputations (an integrated measurement of both their trustworthiness and their trust knowledge of other agents). The model does not only estimate the reputation of every recommender in the recommendation network, it also provides reputation reliability measurement based on the convergence of the derived HMM via the EBW algorithm.

The model can be applied to evaluate agents' expertise under a certain settings. It can also be used to derive optimal recommendation paths across different trust management systems. In P2P applications, it evaluates qualifications and trustworthiness of online processors, and contributes to routing an online application to a correct processor.

CHAPTER 4

GLOBAL REPUTATION IN A DISTRIBUTED TRUST SYSTEM

Current trust models do not provide a mechanism in managing users' global reputations in a distributed system. This chapter proposes a distributed master-slave reputation management structure. The distributed structure has advantages over a large and sparse central system in optimal local reputation management and in load balance of computation time and memory storage. The chapter also develops a global reputation model to derive global reputations from distributed local reputations. The model is adaptive (1) to heterogeneous local reputation models, and (2) to changes of a distributed system structure, users' trust behaviors, and local reputation models.

4.1 Introduction

Current trust or reputation models can be classified into two categories, central and personalized. Central reputation models evaluate online users' reputations based on the entire observations of their online behaviors. Examples are average models, such as e-Bay and Amazon.com, and pairwise trust models, such as SPORAS (Zacharia and Mae 1999). Personalized trust models are also known as recommended trust models. They are based on selected observations. There are four representative personalized trust models. They are the social network topological model developed by Pujol, Sang, and Halberstadt

(2002), Bayesian rating model of Mui, Mohtashemi, and Halberstadt (2002) and Wang and Vassileva (2003), Dempster-Shafer belief model by Yu and Singh (2001), and EigenTrust model by Kamvar, Schlosser, and Garcia-Molina (2003). All these personalized trust models are pairwise. These models use different trust propagation and aggregation algorithms. For example, the social network referral trust model applies the noderanking algorithm (Pujol, Sang, and Halberstadt 2002) to infer reputation of a node within a social network. The noderanking algorithm is similar to the ranking algorithms for Web pages based on Web topology. Bayesian rating model (Mui, Mohtashemi, and Halberstadt 2002) iteratively applied conditional probability along each referral chain and aggregated the derived trust from multiple referral chains by a weighted average method. Wang and Vassileva (2003) introduced naive Bayesian network in managing trust in P2P networks. In the Dempster-Shafer belief model, Yu and Singh (2001) used a belief function. The model was later improved by introducing deception detection mechanisms. Riggs and Wilensky (2001) proposed a distributed and secure method to compute global trust values based on Power iterations. The approach is based on the notion of transitive trust, i.e., trade partners of high reputation are also trustful in making recommendations.

Now, current trust models do not provide a mechanism in managing global reputation of a distributed or decentralized system. This chapter proposes a reputation management structure and designs a reputation model for a very large and sparse distributed system. A distributed reputation system (see Figure 4.1) has a global reputation management agent. It authorizes local agents to manage each of its sub-trust-systems. The sub-trust-system may be grouped by certain criteria; for example, similarity of traders, or traders' connections, in particular, highly connected traders are grouped into

one sub system. A sub trust system is defined as a local community in our research. The authorized local agents apply centralized or distributed trust management to their communities. The global agent collects trust values from its most direct local agents and computes the global reputations of users. The organization of the distributed reputation system is context dependent, i.e., the communities and their members may differ under different contexts.

This chapter also proposes a global reputation model based on the theory of neural networks (Mitchell 1997; Ham and Kostanic 2001). The global agent applies back propagation algorithm (Rumelhart, Hinton, and McClelland 1986) to trust evaluations from its local agents and builds multilayered neural networks. The neural network takes the local trust evaluations as the input and generates approximated global reputations as the output as if the system is centrally managed. Our simulation results show that the neural network converges at a speed of four computation iterations with the approximation accuracy of 94.4%. The neural-network based reputation model has the following properties: (1) The model converges very quickly with high accuracy, and therefore meets the responsible and reliable requirements for online reputation management. (2) The model has the capability to approximate global reputation by nonlinear aggregation of local trust evaluations. (3) Variables that affect global reputation reliability, such as deceptive ratings and local agents' expertise (as a result of their observation sizes, and reliability of their trust models) can be captured by the multiple layers of the neural network. (4) The model is adaptive to changes, including changes in local communities' sizes, their trust models, users' trust behaviors, and restruction of the

distributed system. (5) The model has better performance in terms of computation time and memory than a large and sparse centralized pairwise trust system.

4.2 Distributed Reputation Systems

Reputation is the rate of online users' past behaviors. It is computed based on observations or direct experiences within a specific context at a given time. In a distributed reputation system, local trust management agents may rate the same user differently even under the same context. The top level global reputation management agent is responsible for deriving a user's system-wide reputation from multiple local trust evaluations, no matter whether those ratings are based on completely disjoint or partially overlapped observations.

In a distributed reputation management structure, the global agent aggregates a user's local trust evaluations and derives his global reputations. Figure 4.1 shows the distributed reputation management structure in a large sparse system with multiple local agents. There are one global master agent and many distributed local slave agents. The master agent groups users (context dependent) to distributed local agents. If under the same trust context, a user is assigned to different local agents, the global master agent aggregates the user's multiple local reputations into a global reputation. Grouping criteria can be users' similarities and transaction natures. Local slave agents observe and evaluate trust behaviors of the assigned users only. Local agents determine optimal reputation evaluation models (context dependent). There are total three processes running at the master agent side. One is the global reputation model. The global reputation model is invoked only when the user being evaluated has multiple local reputations. Another is the central reputation model. The central reputation model collects a user's multiple local

reputations and runs a centralized evaluation algorithm. The derived central reputations are used to train the user's global reputation neural network (see details in Section 3). The third process is a monitor process. The monitor process monitors performance of groupings and performance of global reputation neural networks. Once there are changes affecting the performance of a global reputation neural network, the monitor process activates the central reputation model, which in turn generates training data and runs the global model to retrain the neural network. As shown in Figure 4.2, the entire system is made up of four highly connected local systems, A, B, C and D. There is a local agent for each local trust system. Local agents observe their own system members' trust behaviors only and may use different trust models to evaluate their members' local trust. Local agents may also adopt distributed global trust management. The dotted rectangle areas at the bottom layer of Figure 4.2 stand for the local trust system A and B respectively. The small cycles inside a local system represent the local system users/members. The edge between two users stands for a pair of transaction ratings of one toward the other. Local agents are represented by the small rectangles labeled with A, B, C and D at the middle level. The heavy directed edges from a local agent towards its users stand for the users' local trust. The big oval area at the top stands for the global trust model, i.e., a neural network that generates a user's global trust by inputting its multiple local trust evaluations. Derivation of the neural network is displayed by Algorithm 4.1.

The advantages of introducing distributed reputation management in a very large and sparse e-commerce system or P2P networks are:

(1) Local reputation evaluations are more reliable than a large sparse centralized evaluation since the local reputations are evaluated based on the ratings from users of similar trust standards and opinions;

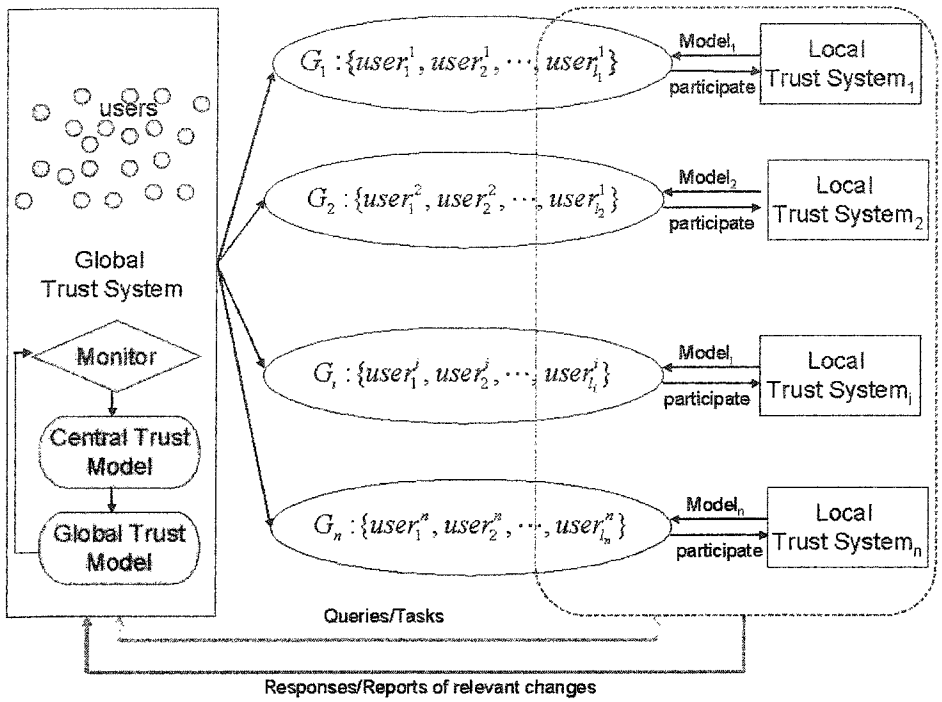


Figure 4.1: Distributed reputation system, where G_i stands for the i^{th} user group evaluated by a local agent. Local slave agents evaluate their users' local reputations. A global master agent evaluates users' global reputations from their multiple local reputations (context dependent).

(2) Local communities have the flexibility to decide which trust model best suits its community members' interests in a certain context and therefore have high impact on users' online transaction decisions;

(3) Distributed and decentralized pairwise trust management is more efficient than centralized pairwise management in terms of memory and data retrieving time. A large centralized sparse system uses either more memory or more tracking time to retrieve pairwise ratings depending on the data structures that the system uses. For example, a central system of N users takes $O(N^2)$ memory to keep the rating

records under a single context if arrays are used. However, online traders are usually very sparsely connected. Some users have strong brand or quality preferences; they may never trade with some other users. If the same system can be divided into n (in a very large sparse system, n can be very large) highly connected sub systems (assumed of equal size), it takes only $O(\frac{N^2}{n})$ memory. Similarly, if the central system uses ordered linked lists to store the pairwise ratings, it takes on average N^2 retrieving time while the retrieving time is only $\frac{N}{2n}$ in the distributed system.

(4) A distributed reputation system distributes loads among its global reputation management agent and local trust management agents. The local agents decide the optimal trust models (context dependent) and evaluate their members' local reputations. The global agent organizes and structures efficient distributed systems, including the grouping of the local communities under different contexts. The global agent also provides global reputation evaluations for users with multiple local reputations.

4.3 Global Reputation Model in Distributed Systems

This section develops a distributed reputation model based on feed-forward neural networks. I apply backpropagation algorithm to build and retrain the networks at a satisfying accuracy level. The basic idea is to aggregate a user's multiple local reputations through a neural network to approximate his/her global reputations. I use SPORAS trust model to evaluate local reputations. I also use this model to derive central reputations.

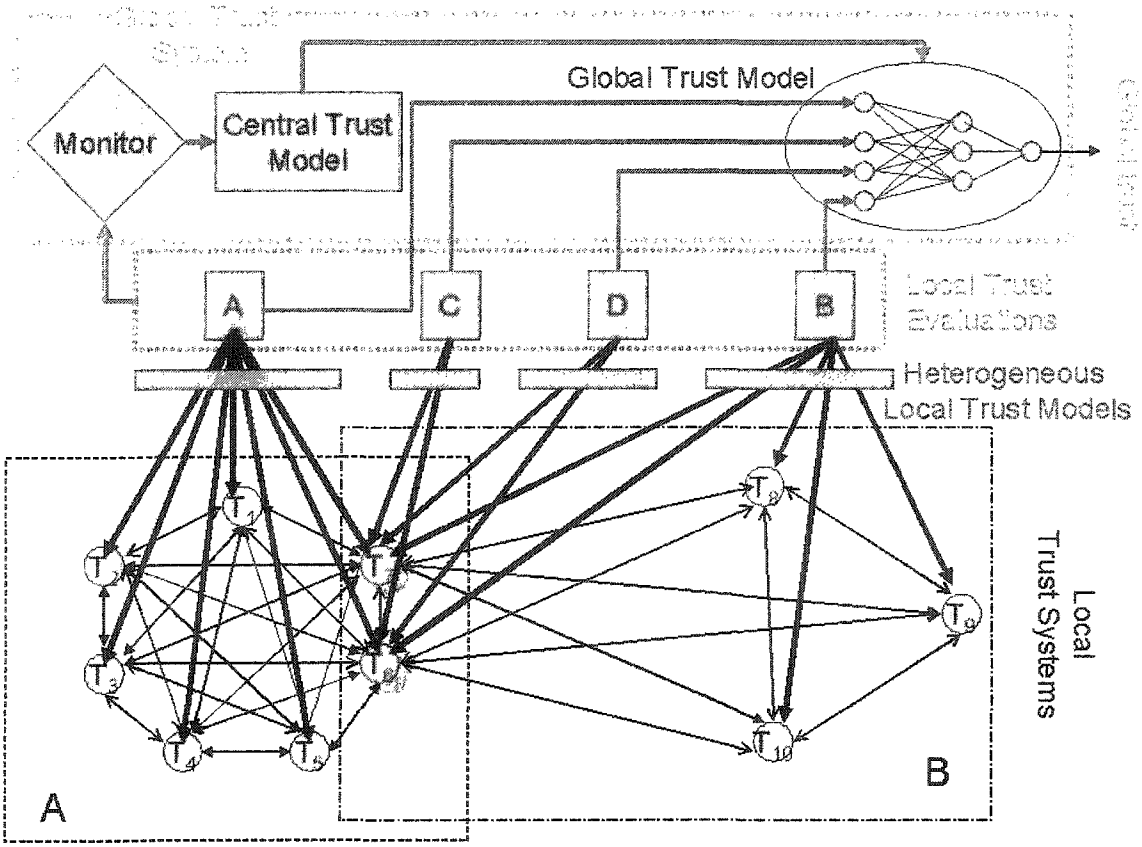


Figure 4.2: Distributed reputation management structure: The entire system is divided into several highly connected local communities managed by local agents. Local agents evaluate users' local reputations using local trust models (see an example in Figure 4.3 and Figure 4.4). The global agent uses a global reputation model to derive a user's global reputation from his multiple local reputations (see Figure 4.6).

4.3.1 Derivation of Local Reputations: SPORAS

In SPORAS, new traders are assigned a minimum positive trust value. Trading partners evaluate each other for each transaction. The local agent aggregates those ratings by assigning weights based on the raters' reputations and the deviations of their ratings. The local reputations are computed as a summation of all the ratings or feedbacks occurred so far, i.e.,

$$R_{t+1} = \frac{1}{\theta}\sum_{i=1}^{t}\Phi(R_i)R_i^{other}(W_i - E(R_i))$$

$$\Phi(R_i) = 1 - \frac{1}{1+e^{\frac{-(R_i - D)}{\sigma}}} \tag{4.1}$$

$$E(R_i) = \frac{R_i}{D}$$

Where, R_{t+1} is the trust value of the ratee at timestep $t + 1$; $E(R_i)$ is the expected trust value of the ratee at timestep $i + 1$, $i = 1, 2, \ldots, t$; θ is a constant integer greater than 1, it is a context dependent parameter; W_i represents the rating given by the rater at timestep i, $i = 1, 2, \ldots, t$, W_i is the feedback value; R_i^{other} is the rater's reputation at timestep i ($i = 1, 2, \ldots, t$) when he/she evaluated the ratee; D is a constant. It is the system-defined upper bound of the reputation value; Φ is the dumping function. It determines the system's sensitivity to changes and it is context specific; σ is a constant. It is used as an acceleration factor to reflect the correspondent trust sensitivities of different e-commerce markets. For example, expensive laptop online market is more trust sensitive than low priced postcards.

One obvious advantage of SPORAS trust model is that it introduces weights on raters' feedbacks. The weights are the combination of raters' reputations and their ratings' diversity. The model has the capability to detect deceptive ratings in some sense. Figure 4.3 shows pairwise transaction ratings among users in a single local community.

Local agents can update a user's local reputation by Equation 4.2:

$$R_{t+1} = R_t + \frac{1}{\theta}R_t^{other}(W_t - E(R_t)). \tag{4.2}$$

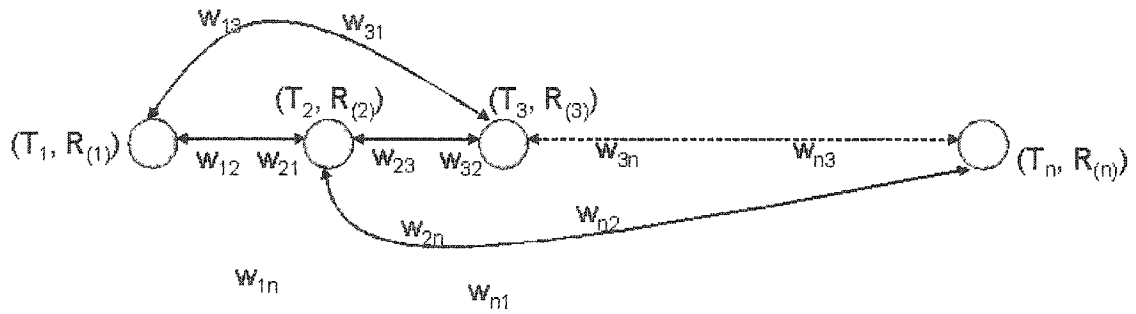


Figure 4.3: Pairwise trust evaluations in a local community. T_i stands for Trader i, $R_{(i)}$ is the local reputation of Trader i and w_{ij} is the rate given by Trader i (rater) to Trader j (ratee).

Figure 4.4 shows how Trader T_1's reputations are updated, where the combination of the new rating, the rater's reputation and the diversity of the ratings take effect.

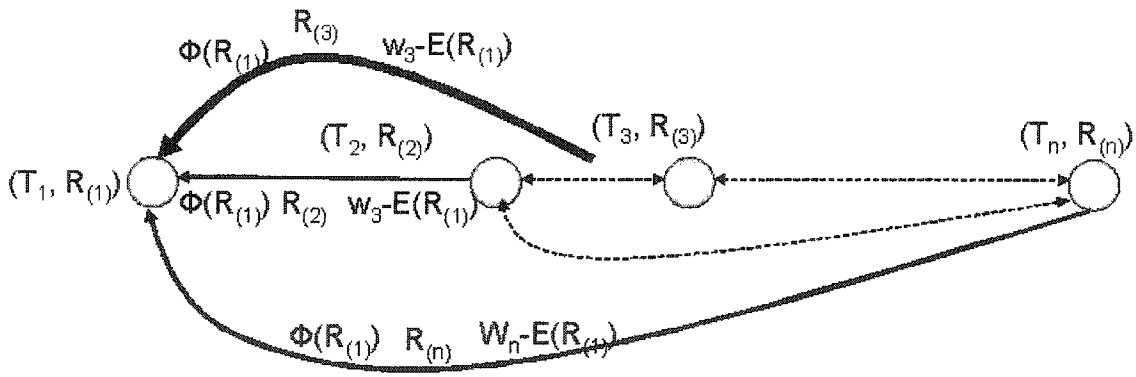


Figure 4.4: Updating Trader T_1's Reputation. The new reputation of T_1 is determined by the three single directed weighted arrows. The two heavy arrows stand for high weights assigned to the new ratings.

I choose to use SPORAS trust model for the following reasons:

(1) SPORAS (Zacharia and Mae 1999) is a reputation system designed for highly connected communities made up of users of similar trust opinions. Therefore, SPORAS provides good trust management for local communities since the reputation evaluation can be widely agreeable by those of similar opinions.

(2) SPORAS trust model is fair, reliable and responsive in certain senses according to a case study that was conducted which is provided in Appendix B. In fact, as the

information in Appendix B shows, no matter what the local trust models are, those models only affect the training data value and have no effect on the global reputation model itself.

(3) Distributed trust management is more efficient than centralized pairwise trust management for a large sparse e-commerce community or P2P network in term of memory and computational time as analyzed in Section 2.

4.3.2 Nueral Network Training: Backpropagation Algorithm

Backpropagation is one of the most common, practical and successful artificial neural network (ANN) approaches. Neural network-based reputation model suits the needs in the distributed reputation management system because:

(1) Multiple local reputations of a user can be easily represented by the input units in the neural network.

(2) Global reputations may be discrete-valued, real-valued, or a vector of real or discrete values. For example, a music file provider's reputation may be represented by the music quality and the file download speed. Global reputations, as the output units of the neural network(s), can be represented by any combination of the input local reputations linearly or nonlinearly. If a vector reputation is used, we can have one neural network for each of the vector element.

(3) Local reputations may contain errors due to users' fraud ratings. Backpropagation has proved to have very good performance to noises in training data (Mitchell 1997). This makes our neural network based global reputation model more robust to erroneous local reputation evaluations.

(4) Local reputations may be based on completely disjoined or partially overlapped observations. Neural networks updated by backpropagation rule are capable of expressing a rich variety of nonlinear decision surface. In fact, a network of three layers of units (see Figure 4.6) is able to approximate any function to arbitrary accuracy given a sufficient number of units in each layer. Our model is set to be three-layered to ensure estimation accuracy and convergence speed.

(5) Neural network-based global reputation model trained by backpropagation can meet different accuracy requirement as long as the network contains sufficient inner layer units (Mitchell 1997). This empowers reputation evaluations under different contexts since some online transactions may have rigid requirement on the reliabilities of reputation evaluations.

Backpropagation algorithm uses gradient descent to minimize the squared estimation errors of users' global reputations. The hypothesis space considered by the algorithm is the space of all functions that can be represented by assigning weights to the given network. Therefore, the reputation model has the capability to catch other hidden factors other than the multiple local reputations through multi-layered network structures.

Two steps are involved in building a neural network. The first step is to initialize layers, units in those layers and weights of unit connections. The number of input units is equal to the number of a user's local reputations. There is only one unit in the output layer representing the user's global reputation. Hidden layers and hidden units can be set as many as necessary such that they can catch nonlinear relationship between local reputations and the desired global reputations, and can meet accuracy and convergence speed requirements. There is a neural network per user context. The second step is to

train the neural network so that the summation of squared approximation errors is minimized. The backpropagation algorithm is used for this purpose. The user's local reputations at different time periods are forwarded through the neural network. Approximation errors are the differences between the generated global reputation values by the output layer unit of the global reputation model and the central reputation values obtained through a central reputation model. These errors are propagated backward through the neural network to adjust the connection weights. Every time the weights are updated, the approximation errors are decreased. The weight update process (Mitchell 1997) continues till the neural network provides satisfying global reputation estimations. The output at unit u is:

$$o_u = \frac{1}{1+e^{-\vec{W}_u \vec{R}_u}} \tag{4.3}$$

Error terms at the output unit and the hidden units are δ and δ_h individually:

$$\delta = o(1-o)(t-o) \tag{4.4}$$

$$\delta_h = o_h(1-o_h)w_h\delta \tag{4.5}$$

Weights are updated as:

$$w_{ij} = w_{ij} + \eta\delta_j o_i \tag{4.6}$$

Where $\vec{W}_u$ and $\vec{R}_u$ represent a weight vector and a reputation vector forwarded to unit u respectively. o is the estimated global reputation generated by the output unit, and o_i is the output value of unit i. t is the central reputation obtained from a central reputation model. w_h is the weight of a connection between hidden unit h and the output unit. w_{ij} is the weight from unit i to unit j.

Once a user's global reputation neural network is built, if there are no changes in system structures, local agents' reputation models, or the user's trust behaviors, we can simply feed the user's multiple local reputations through the neural network and immediately get the user's global reputation. However, when any of the changes occur, the monitor process will invoke the central reputation model, which in turn retrains the global reputation neural network. Algorithm 4.1 shows the details.

Algorithm 4.1. Global Reputation Model of Distributed Trust Management

```
while (the master agent is waiting for reputation queries)
        run monitor process;
        if (there are changes)
                for (each related user of multiple local reputations)
                        run central reputation model;
                        train global reputation neural network;
                end for
        end if
end while
if (the user is evaluated by one local agent)
        forward the reputation query to that local agent;
else
        collect the user's multiple local reputations;
        input the local reputations through the user's global
        reputation neural network;
        output the approximated global reputation;
end if
```

4.4 Experiments

The experiments consider a distributed network of four local trust communities A, B, C and D. Within each community, the members are highly connected and have similar trust opinions. I simulated a distributed reputation network of 10 users and four local reputation communities. Total 2000 transactions were generated randomly among the users. Table 4.1 shows the local communities' sizes and members. User 1-10's trust behaviors are listed in Table 4.2 regardless of their communities. User 6 and User 7 are evaluated by all the four trust communities.

The experiments evaluated User 6 and User 7's local and global reputations, and tested the performance of the neural network-based global reputation model. For simplicity, all the local slave agents were assumed using SPORAS pairwise reputation model. The master agent also used SPORAS as his central reputation model. In theory, local reputation models and the central reputation model can be different. The parameters of SPORAS model were set as: $\theta = 5$, $\sigma = 10$, and $D = 3000$.

Table 4.1. Simulated distribute system

Local Community	Community Member	Community Size
A	User1-7	7
B	User6-10	5
C	User5-7	3
D	User4-8	5

Table 4.2. User's trust behavior

User 1	User 2	User 3	User 4	User 5	User 6	User 7	User 8	User 9	User 10
[.6, 1.0]	[.7, .9]	[.8, .8]	[.8, 1.0]	[.9, 1.0]	[1.0, 1.0]	[.4, 1.0]	[.7, 1.0]	[.6, 0.9]	[.8, .9]

4.4.1 Local Reputations

Figure 4.5 compares User 6 and User 7's reputation development in the four local communities. User 6 and User 7's central reputations are also plotted. User 6's local and central reputations converged at a stable value, 3000, which is the highest possible trust level of the SPORAS system. This agrees with User 6's trust behaviors specified in Table 4.2. User 7's local and central reputations converged at a vibration of a central value 2100, agreeing with the product of his average trust behavior value and the system highest reputation value. An interesting phenomenon is that User 6 and User 7 were still in the phases of building up their familiarity and reputation in Community C while their reputations were convergent in the other three communities A, B and D. The reason is

that User 6 and User 7 traded least frequently in Community C. As shown in Table 4.3, User 6's transactions at Community A, B, C, D were 202, 127, 98, and 48 respectively. User 7's transactions at those communities were 190, 27, 86 and 46 respectively. Transaction volumes lead to the differences in local reputation evaluations and in reputation convergence.

Table 4.3: Transaction volumes of User 6 and User 7 in different local communities

Transaction Volumes	Community A	Community B	Community C	Community D
User 6	202	127	48	98
User 7	190	127	46	86

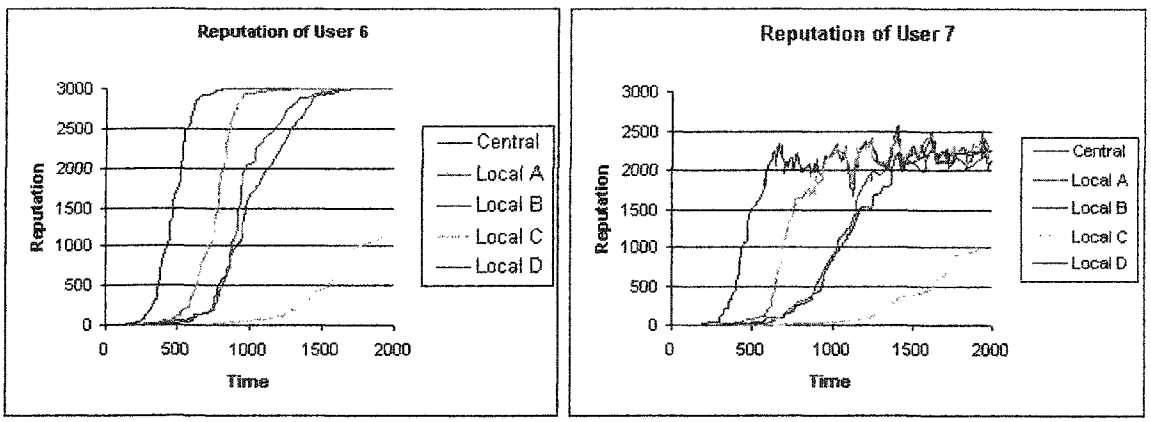


Figure 4.5: Reputation evolution of User 6 and User 7. The speed of convergence monotonically increases with trade volumes.

4.4.2 Robustness of the Global Reputation Model

I tested the reliability of the global reputation model in terms of the model's correctness and convergence speed by varying:

(1) learning speed,

(2) training data size, and

(3) estimation accuracy.

I set a three-layered neural network where there are three units in the hidden layer (See Figure 4.6). The initial neural network connection weights were set randomly in the range of [-0.05, 0.05]. Learning rate η was set randomly in the range of [0.4, 0.6]. Estimation error threshold d was set in the range from 0.05 to 0.30. An estimation is correct if it satisfies: $|o - t| < d$, where o is the output of the neural network and t is the desired reputation (obtained through the central reputation model). I stopped training the neural networks when at least 92% estimations were correct at a predefined error threshold.

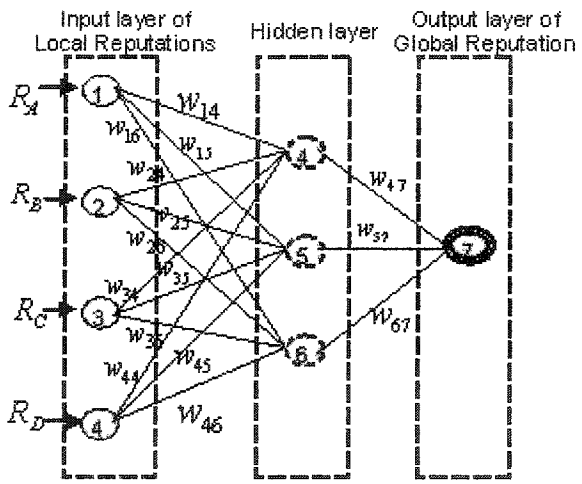


Figure 4.6: Neural Network: a three-layered neural network with 3 units in the hidden layer. R_i, $i = A, B, C, D$, stands for the local reputation and w_{ij} stands for the connection weight from unit i to unit j.

Given a training set of 16 local reputations, learning rate $\eta = 0.59$, and the initial connection weights as:

$$W_{in}^0 = \begin{pmatrix} -0.05 & -0.05 & 0.05 \\ 0.05 & 0.05 & -0.05 \\ 0.05 & 0.05 & 0.05 \\ -0.05 & 0.05 & -0.05 \end{pmatrix}, \quad W_h^0 = \begin{pmatrix} -0.05 \\ 0.05 \\ -0.05 \end{pmatrix}.$$

Where W_{in}^{0} is the connection weights between the input layer and the hidden layer, and W_{h}^{0} is the connection weights between the hidden layer and the output layer. The neural network converged at the 4^{th} iteration (of algorithm 4.1) with 100% successful estimations. The summation of the squared estimation errors is only 0.1403. The connection weights of the converged neural network are:

$$W_{in} = \begin{pmatrix} -0.06754 & -0.0809 & 0.1682 \\ 0.1448 & 0.1550 & -0.0454 \\ 0.6981 & -0.0717 & 0.0700 \\ -0.0458 & 0.1559 & -0.0464 \end{pmatrix}, \quad W_{h}^{0} = \begin{pmatrix} 0.8896 \\ 1.1106 \\ 0.8961 \end{pmatrix}.$$

Table 4.4 and Table 4.5 show the experimental results with various initial neural network connection weights and learning rates. There were 16 sets of training data. I repeated the experiments over twenty times. All the results showed that the neural network converged at no more than 5 iterations with at least 93.75% successful estimations. The convergence speed and the high accuracy satisfy the requirement of online trust evaluations, i.e., responsiveness and reliability.

Table 4.4: Convergence and correctness of User 6's global reputation under various neural network learning rates and initial connection weights

Learning Rate	Convergence Speed	Estimation Correctness	Approximation Error $\sum_{i=1}^{16}(o-t)^2$
0.59	4	100%	0.1403
0.57	4	100%	0.150
0.47	4	100%	0.188
0.43	4	100%	0.234
0.45	5	100%	0.156

Table 4.6 shows the experimental results after training 18 sets of User 6's local reputations. The distributed reputation model converges at 3rd or 4th iteration with 94.44% successful estimations (17 out of 18).

Figure 4.7 and Figure 4.8 show the correctness and convergence of the global trust model under various accuracy requirements. I set 10 different estimation thresholds. At each threshold, I ran the global reputation model 20 times. Each time I randomly selected a learning rate and initial neural network connection weights. The mean values of correctness and convergence speed are compared. Figure 4.7 shows that all the estimations of the global reputation model had at most an estimation error size of 0.10. When estimation error threshold was 0.05, there were at least 93.75% correct estimations. Figure 4.8 shows that in the worst case, the model converged at an average of 71,696 iterations such that 93.75% estimations were at the accuracy level of the estimation error less than 0.05. The convergence speed was reduced from 267 iterations to 5714 iterations when User 7's global reputation estimation error threshold was reduced from 0.30 to 0.05. I noticed a dramatic slowdown of User 6's global reputation convergence speed when the estimation error threshold was reduced to 0.05. Nonlinearity of local reputations and global reputations, or learning rate size may contribute to the slowdown.

Table 4.5: Convergence and correctness of User 7's global reputation under various neural network learning rates and initial connection weights

Learning Rate	Convergence Speed	Estimation Correctness	Approximation Error $\sum_{i=1}^{16}(o-t)^2$
0.43	4	100%	0.2096
0.50	4	100%	0.1672
0.43	4	93.75%	0.2156
0.57	4	100%	0.1533
0.51	4	100%	0.1703

4.5 Conclusions and Future Work

Centralized trust models are not appropriate to apply in a large and sparse distributed trust system. First, online users are usually sparsely connected under one or more reputation contexts. Second, local reputations may be evaluated by different trust models based on partially overlapped observations. To improve reputation evaluation reliability, this paper proposes a distributed trust management structure, and designs a neural network based reputation model. The model derives users' global reputations from their multiple local reputations. The distributed reputation model is a novel application of neural networks. A user's global trust neural network takes the user's multiple local reputations as input and output the user's global reputation as if the system were centrally managed. The model has several important properties for online reputation evaluations:

(1) It can be used to derive global vector reputations.

(2) It is applicable to any distributed trust systems based on different criteria, such as similarity of users, geographic closeness, and connectivity of users.

Table 4.6: Convergence and correctness of User 6's global reputation given 18 sets of training data

Learning Rate	Convergence Speed	Estimation Correctness	Approximation Error $\sum_{i=1}^{18}(o-t)^2$
0.52	3	94.4%	1.1564
0.47	3	94.4%	1.1672
0.57	3	94.4%	1.2087
0.58	4	94.4%	1.6042
0.54	3	94.4%	1.2595

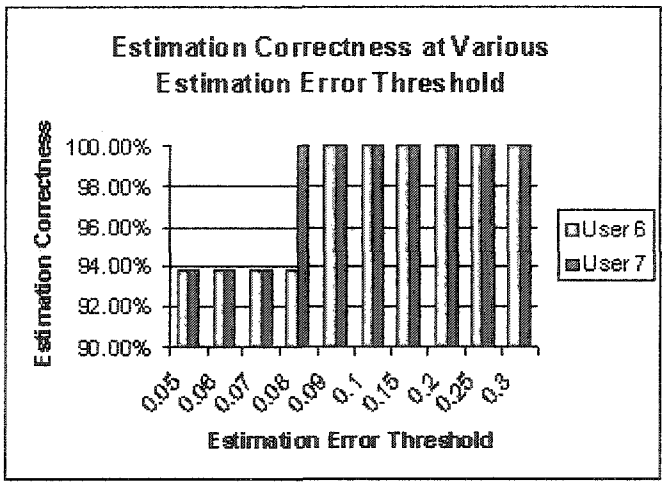


Figure 4.7: Estimation correctness of User 6 and User 7's global reputations at various estimation error thresholds.

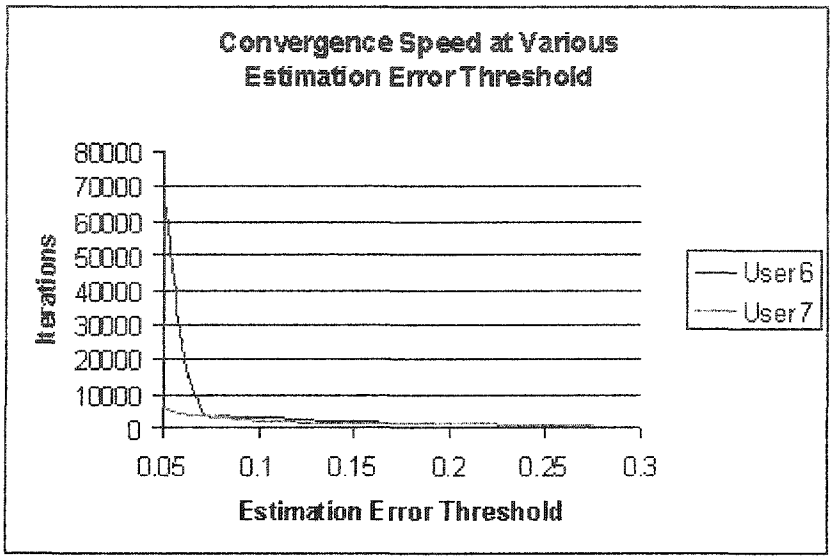


Figure 4.8: Convergence of User 6 and User 7's global reputations at various estimation accuracy levels.

(3) It allows different local reputation models.

(4) It is adaptable to any changes in local trust systems or distributed trust system restructuring.

(5) It has robust performance of quick convergence speed and high estimation accuracy.

The model has several important applications. It can be applied in a decentralized trust system. It can also be used to aggregate recommendation trust from multiple resources. If used in a P2P network or in pairwise trust management, the model distributes the load of memory space and computation time between global and local reputation management agents. In addition, the model can be used to derive both single valued and vector valued global reputations. For example, a global reputation vector may include attributes of product quality, in-time-delivery, and after-sales-services.

The performance of the global reputation model depends largely on the monitor process, which detects any related changes that may require neural networks retraining. Future research includes: (1) developing change detection algorithms used in the monitor process; (2) testing the global trust model's performance in a dynamic environment, where agents change their trust models, or users change their trust behaviors; and (3) applying real data to test the reliability of the global trust model, if real data are available.

CHAPTER 5

CONCLUSIONS AND SUGGESTIONS FOR FURTHER WORK

5.1 Summary

Trust plays an important role in e-commerce, distributed computing and peer-to-peer networks. To solve the recommendation trust problem, this dissertation begins with two alternative methods to obtain recommendation trust. One way is to filter and integrate heterogeneous trust opinions (Chapter 2). The other way is to evaluate the trustworthiness of the recommenders (Chapter 3). Then I discuss the scalability and efficiency of a trust system by introducing distributed trust management, where local trust agents can be taken as recommenders. To solve this special trust recommendation problem in a distributed system, I develop a global trust model (Chapter 4).

Current recommendation trust models fall into one category, i.e., seeking recommendations from agents that have similar trust standards and evaluation models. However, in multiagent systems, we may be unable to find such recommendations. It is with this background that we develop a method to derive an agent's trust opinion from heterogeneous recommendations. First, I develop an ordered depth-first search algorithm to find reliable recommenders (Section 2.2.2). Then I develop an algorithm to identify qualified recommenders from those recommenders (Algorithm 2.4). A neural network is

built and trained by the trust opinions of the qualified recommenders. The neural network detects the patterns between the requester's trust opinion and the recommenders' trust opinions. By feeding the neural network with the heterogeneous recommendations, we get the requester's trust opinion. This work is a new application of neural networks in recommendation trust. It filters opinions of different trust standards, evaluation models and trust observations. The experimental results showed that a three-layered neural network converges at an average of 12528 iterations and 93.75% of the estimations have an error size less than 0.05. More important, the model is adaptive to trust behavior changes and has robust performance when there is high estimation accuracy requirement or when there are deceptive recommendations.

Alternatively, instead of finding the patterns between the requester's trust opinions and the recommenders' trust opinions, we evaluate recommenders, since we can get reliable recommendations from trustworthy recommenders. A Hidden Markov Model (HMM) is used to model recommendation events with the assumption that agents learn to choose the most reliable and knowledgeable recommenders. There are four attractive features of the approach. First, it does not require explicit reputation evaluations of chained recommendations. Second, it integrates a recommender's expertise as well as his trustworthiness into his reputation evaluations. Third, it is applicable to any possible recommendation networks including those with unreachable nodes and loops. Fourth, the approach quantifies the learning speed of an agent's converged reputation as a recommender. The learning speed can be used as a reliability measurement when recommendation events are sparse. The model can be applied to identify optimal recommendation paths and to locate reliable servers in P2P networks.

With the growth of trust systems, scalability and load balance need to be taken care of. As a result, distributed trust systems are in demand. Due to the fact that in a very large distributed trust system, local level reputation management may use different trust evaluation models, and users' reputations may be reported by multiple agents based on different (may be partially overlapped) observations, I propose a distributed trust system including its structure, architecture, and trust computations. Correspondingly, a global trust model is developed to aggregate a user's multiple local reputations. We use the back propagation algorithm to train the global neural network model. The experimental results show that the global reputation model had estimation error size less than 0.10. When estimation error threshold was 0.05, there were at least 93.75% correct estimations. In the worst case where strong nonlinearity exists, the model converged at an average of 71,696 iterations such that 93.75% estimations had estimation error less than 0.05.

In summary, this work is made up of three parts. The first two parts solve the recommendation trust problem in two different ways. Part one (Chapter 2) filters and integrates heterogeneous trust recommendations by building an opinion filtered recommendation trust model in multiagent systems. Part two (Chapter 3) evaluates recommenders' reputations to get reliable trust recommendation by constructing HMM model. Part three (Chapter 4) considers a special case where local agents are the recommenders in a distributed trust system. A global reputation model is developed.

5.2 Further Work

This section consists of two parts. First, we introduce the further work to improve the proposed three trust models. Then we discuss the need to develop metrics to evaluate various trust models.

5.2.1 Improvement of the Three Proposed Trust Models

First, a monitor process needs to run on top of the opinion filtered recommendation trust model. The opinion filtered recommendation trust model is not appropriate to apply in a newly developed P2P network where agents have not developed enough direct experiences with each other. Given insufficient recommendations from the qualified recommenders, the trained neural network may not provide a desirable accuracy level. This can be avoided by integrating an opinion filtered recommendation trust model with another recommendation model, for example, Bayesian model by Mui, Mohtashemi, and Halberstadt (2002). When the system is first established, the other trust model is put into use. While training data grows, the opinion filtered recommendation trust model takes the place of the other trust model. However, when qualified recommenders change their trust evaluation models, the neural network model needs to be retrained to maintain high accuracy. That is why a monitor process needed to ensure the reliability and accuracy of the opinion filtered recommendation trust model. In the future, a monitor algorithm needs to be developed such that the monitor process can automatically make the decision to retrain the neural network to improve its accuracy.

Second, there is a need to design a recommender's reputation model in a dynamic environment. Since the hidden Markov reputation model requires fixed state transition networks and independent recommendation observations (a series of recommendation

events under the same trust context), the hidden Markov reputation model is not applicable to a dynamic system. For instance, nodes in a P2P network may be removed or replaced by new nodes. This leads to frequent changes of the recommendation trust network that results in the failure of building a HMM-based recommender's reputation. Therefore, new methods need to be developed to accommodate the dynamic property of a changing trust network.

In addition, there is a need to develop an algorithm to detect changes in users' trust behaviors. The performance of the global reputation model in a distributed system is subject to changes in users' local trust behaviors. If such changes are detected in time, a user's global reputation can be reevaluated by retraining the underlying neural networks and thus improve the model's reliability and accuracy. Some initial results have shown that quality control technique is a good method in detecting changes at the early stage (Xu, Phoha, and Song 2005a; Xu, Phoha, and Song 2005b). In the future, I plan to apply other statistical related methods and anomaly detection algorithms to detect trust behavior changes and false trust reports.

5.2.2 Development of Trust Model Evaluation Metrics

Although much research has been conducted in building trust models, very little research is available in evaluating these trust models, and more importantly, in developing formal methods or metrics to measure the robustness of the various trust models. Existence of contradicting research findings also demonstrates the need for a well established discipline to evaluate trust systems. For example, Dellarocas (Dellarocas 2003) argued for the efficiency and robustness of eBay-like online feedback mechanisms, but Mui et al. (Mui, Mohtashemi, and Halberstadt 2002) claimed "a few recent high

profile fraud cases in eBay by individuals with high eBay ratings suggest that the company should seriously consider enhancing their simple rating system". Rensnick et al. (Resnick and Zechhauser) found Pollyanna effect (disproportionately positive feedbacks from users and rare negative feedbacks) in eBay reputation reporting system and interpreted this as an evidence of the poor functioning of eBay's trust mechanism. However, Dellarocas (Dellarocas 2003) explained this as a supporting evidence of the efficiency of eBay trust mechanism.

In the future, I aim to conduct research in formalizing evaluations of various trust models. The research will focus on a characterization and analysis of a *fair*, *reliable* and *responsive* trust system, and on the development of metrics to evaluate trust systems in terms of fairness, reliability and responsiveness. The research will be carried out specifically in four aspects: (1) formally defining fairness, reliability and responsiveness; (2) developing metrics to evaluate trust models in terms of fairness, reliability and responsiveness; (3) studying the effect of trust on users' profitability by building optimization models under various product life cycle scenarios. The optimization models are used to test whether a trust system punishes users of oscillating transaction qualities and makes their overall profit less optimal; and (4) conducting case studies to evaluate and compare different trust models, for example SPORAS, and some average trust models.

APPENDIX A

LIST OF THE RECOMMENDATION OBJECTS

(1) JBuilder

(2) JCreator

(3) Visual C++

(4) SAS

(5) Blackboard

(6) University Email

(7) Yahoo Email

(8) Google search

(9) Yahoo search

(10) MSN search

(11) java.sun.com

(12) Amazon.com

(13) eBay.com

(14) Numerical Analysis (6^{th} edition) by Richard L. Burden, J. Douglas Faires, Brooks/Cole Publishing Company, 1997, ISBN: 0-534-95532-0.

(15) Partial Differential Equations by Lawrence C. Evans.

(16) Introduction to the Finite Element Method by Erick G. Thompson, John Wiley & Sons, 2005

(17) A first course in Probability (6^{th} edition) by Sheldon Ross, 2001.

(18) Time Series Analysis by William W. S. Wei, ISBN: 0-201-15911-2.

(19) Design and Analysis of Experiments by Angela Dean and Daniel Voss, Springer, 1999, ISBN: 0-387-98561-1.

(20) Fundamentals of Sequential and Parallel Algorithms (6^{th} edition) by Kenneth A. Berman and Jerome L. Paul, PWS Publishing Company, 1996, ISBN: 0-534-94674-7.

(21) Introduction to Algorithms by Thomas H. Cormen.

(22) Concepts of Programming Languages (6^{th} edition) by Robert W. Sebesta.

(23) Software Engineering by Ian Somerville.

(24) Operating System Concepts by Abraham Silberschatz et. al.

(25) Computer Organization and Architecture (5^{th} edition) by Williams Stallings, Prentice-Hall International, Inc., ISBN: 0-13-085263-5.

(26) Modern Database Management (7^{th} edition) by Jeffrey A. Hoffer et al., 2004.

(27) Fundamentals of Database Systems (4^{th} edition) by Ramez Elmasri and Shamkant B. Navathe.

(28) The C Programming Language by Brain W. Kernighan, Dennis Ritchie et al.

(29) Advanced Computer Architecture: A design space approach by Dezso Sima, Terence Fountain and Peter Kacsuk, 1997.

(30) High performance cluster computing: Architectures and Systems (Vol. 1) by Rajkumar Buyya (editor), ISBN: 0-13-013784-7.

APPENDIX B

CASE STUDY: SPORAS SYSTEM

This case study measures update scales, effectiveness and significance of transaction feedbacks, and response speed, to evaluate SPORAS (Zacharia and Mae 1999), a pairwise central trust model.

The case study is based on simulated data. Data simulations were made up of two parts. In the first part I simulated users' online transactions. 500 transactions were simulated among 5 users. Trading partners and the 1000 transaction feedbacks were randomly generated. The transaction feedbacks are in a range from 0 to 1. In the second part we applied SPORAS trust model and simulated the evolution of the users' trust values. The parameters in SPORAS trust model were set as: $D = 3000$, $\theta = 5$, and $\sigma = 10$. The trust value is updated whenever there is a transaction feedback. I assume each transaction has the same transaction size.

I conducted the study of SPORAS in the following three aspects:

(1) Fair and Appropriate Trust Update: This part tests the fairness and reliability of SPORAS. We measured the trust update scales by computing the five users' trust value at different timesteps. Figure B.1 shows the five users' trust values at different timesteps. The five users had identical transaction feedbacks randomly distributed in the range from 0 to 1. In the beginning, users' trust values increase. The increase rate at the 158^{th}

timestep is the largest (the slope is the sharpest at timestep158). After the first 500 updates, trust values vibrate around a centered value, 1500. This central value is actually the convergence trust value (Zacharia and Mae 1999) $D\overline{w}$, where $\overline{w}$ is the mean value of the transaction feedbacks. Figure B.2 shows the users' trust values when the users do not have identical transaction feedbacks. Table B.1 shows the five users' transaction feedbacks in detail. Table B.2 shows the convergence trust values and the transaction sizes of the five users. Both Figure B.1 and Figure B.2 indicate that SPORAS is fair in terms of distinguishing users of different trust levels.

Table B.1. Online transaction feedbacks of User 1-5

User 1	User 2	User 3	User 4	User 5
[0.3, 0.5]	[0.5, 0.7]	[0.6, 0.9]	[0.8, 0.9]	[0.9, 1.0]

Table B.2. Online transaction feedbacks of User 1-5

Users	User 1	User 2	User 3	User 4	User 5
Total Transaction Size	117	135	134	123	123
Transactions before Convergence	72	64	78	98	86
Mean Trust Value	1197.14	1825.68	2232.43	2547.25	2843.45

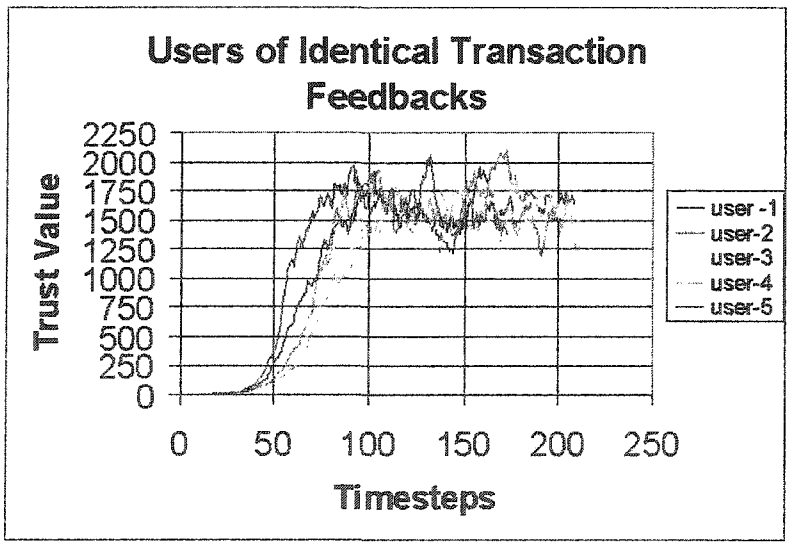


Figure B.1: Trust value evolutions of five identical users. There are two evolutionary stages of SPORAS system: trust building stage before timestep 158 and trust convergence stage after timestep 158.

Figure B.3 demonstrates the trust update scales at the trust building stage and the trust convergence stage. The update scales in the building stage are 5 times as large as the update scales in the convergence stage. This reflects the relationship between the transaction size and the trust value update size, i.e., a single transaction has different effect on users of different transaction histories.

To further measure the effectiveness and significance of a single transaction feedback on a user's trust value, I interpolate User 1's trust value and his transaction size. Since the transactions are uniformly and randomly generated, and all the transactions are of equal size, trust value update timesteps can be used to represent User 1's transaction size. Therefore, an interpolation of User 1's trust values and the update timesteps would demonstrate the effect and significance of transaction feedbacks. Among least squares, minimax and absolute deviation methods, I chose to use least squares method because of the following theoretical considerations:

(a) The minimax approach generally assigns too much weight to a bit of data that is badly in error.

(b) The method using absolute deviation simply averages the error at the various points and does not give sufficient weight to a point that is considerably out of line with the approximation.

(c) The least square approach puts substantially more weight on a point that is out of line with the rest of data but will not allow that point to completely dominate the approximation.

(d) Least square approach has better performance in terms of the statistical distribution of error (pp463-481, Larson 1984).

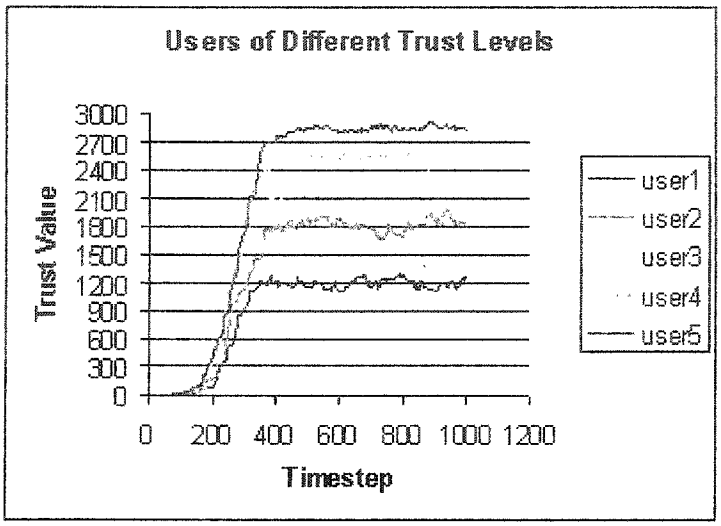
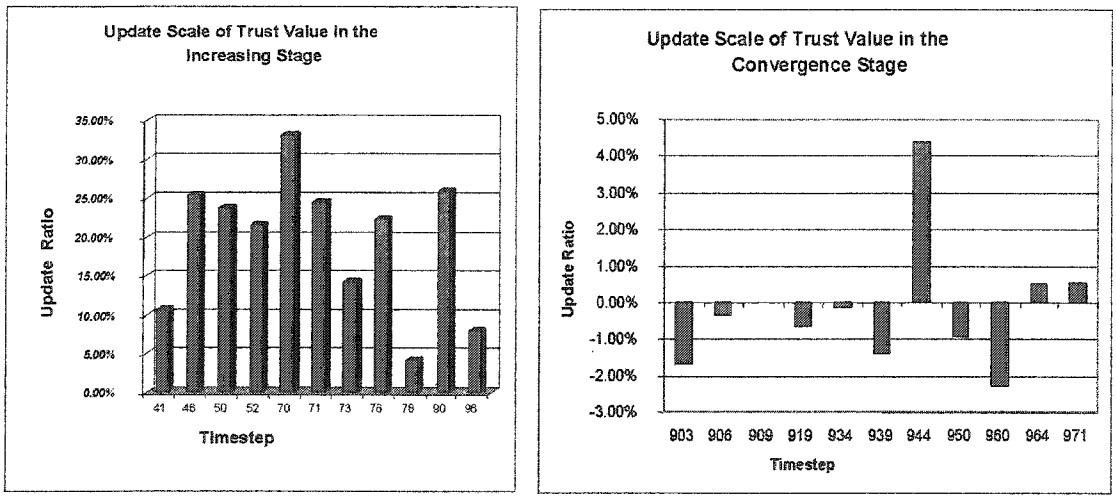


Figure B.2: Discrimination of the users of different trust feedbacks (see Table B.1 for details).



(a) Update Scale in Increasing Stage (b) Update Scale in Convergence Stage

Figure B.3: Comparison of trust update size of User 3 at increasing and convergence stages.

I use piece-wise approximations since the simulation results shown in Figure B.1 and B.2 demonstrate two distinct stages in trust value evolutions. I use quadratic method to approximate User 1's trust values in the trust building or increasing stage. Let $P_2(t)$

stand for the approximated trust value at timestep t, we have: $P_2(t) = a_0 + a_1 t + a_2 t^2$. Our aim is to minimize the squared estimation errors, i.e.,

$$\min E = \min \sum_{t \in \text{increasing stage}} (R_t - P_2(t))^2$$

The squared error summation E is minimized when $\partial E/\partial a_j = 0$ for each $j = 0, 1, 2$. This gives 3 normal equations with 3 unknown a_j 's, i.e.,

$$\sum_{k=0}^{2} a_k \sum_{t=1}^{72} t_i^{j+k} = \sum_{i=1}^{72} R_i t_i^j \ , \ j = 0, 1, 2.$$

Given that t_i's are distinct, the three normal equations have a unique solution. Since in the convergence stage, the trust values converge to $D\overline{w}$ (Zacharia and Mae 1999), where $D = 3000$ is the system maximum trust value, and $\overline{w}$ is the mean feedback value. User 1's trust function is interpolated as:

$$R(t) = \begin{cases} 0.02t^2 - 3.5526t + 102.2598, 143 \le t < 344 \\ 1200, t \ge 344 \end{cases} \quad (1)$$

Function (1) shows that User 1's trust value update scales are positively related with his transaction size (represented by update timesteps due to uniformly randomized transaction generation and equal transaction size). This is because in the increasing stage, we have $\frac{dR}{dt} = 0.0410t - 3.5526 > 0$. Figure B.4 demonstrates the approximation accuracy of Function (1). The average error of 161 sets of data is only 3.64%, and the standard deviation of the errors is 0.189. The error is defined as $\frac{|R_{app} - R|}{R}$, where R_{app} and R represent the estimated trust value by Function (1) and the derived trust value by SPORAS model respectively.

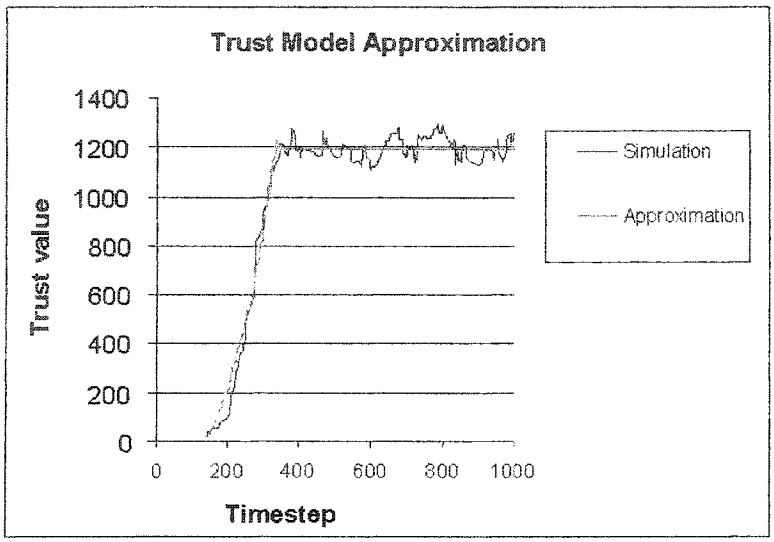


Figure B.4: Approximation of User 1's trust value. The estimation function demonstrates the effectiveness and significance of transaction feedbacks, i.e., each single transaction matters and transaction history matters.

(2) Reliable Response to changes: *Response time* is used to test the short term responsiveness of SPORAS. *Response time* is the number of transactions between the user's two convergence trust values before and after the trust behavior changes. We conducted two tests to study the responsiveness of SPORAS.

In Test 1, among the five users only User 1's trust behavior changed. User 1's trust behavior changed from a uniform distribution of [0.3, 0.5] in the first 500 transactions to [0.9, 1.0] in the second 500 transactions. The results (see in Figure B.5) show that before the new convergence, User 1's trust value increased at an average rate of 104.03% each consecutive timestep. It took only three updates (out of total 20 system-wide updates) before User 1's trust value converged to a new equilibrium value. The test also shows that the model provided a fair evaluation mechanism, since the changes in User 1's trust behavior did not have significant effects on the other users' trust values although the system applies higher weight to User 1's feedbacks because of his improved trust values. This property is very important since the system is capable of

deciding to which extend the rater's reputation takes effect on other users' (ratees') trust values. According to SPORAS model, a ratee's trust value is determined by the product of the changes in the ratee's trust behavior ($W - E(R)$) (see Section 4.3.1) and the rater's reputation. However, when factor $W - E(R)$ is zero or small enough, the rater's reputation has no or neglectable effect on the ratee's trust value. This explains why the trust values of the other four users were almost unaffected.

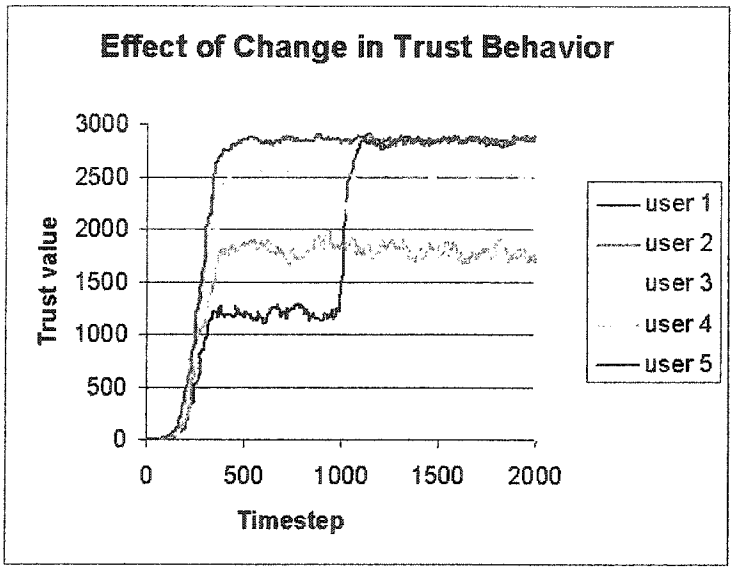


Figure B.5. Responsiveness of SPORAS: User 1's trust behavior changes.

Test 2 simulated the scenario that three out of the five users changed their trust behaviors. Table B.3 shows the details of the changes. Figure B.6 shows how the system responses to the changes. User 1 and User 3's trust values increased by 12.645% and 2.886% respectively before their trust values converged. User 5's trust value decreased by 9.977%. The changes in those three users' trust behaviors did not have significant effect on other users' trust values. Test 2 confirmed the analysis that a rater's reputation takes effective only when the changes in the ratee's trust behavior ($W - E(R)$) (see Section 4.3.1) is significant.

Table B.3. Trust behaviors of User 1-5 in the 1000 transactions

User	1^{st} 500 Transactions	2^{nd} 500 Transactions
User 1	[0.3, 0.5]	[0.9, 1.0]
User 2	[0.5, 0.7]	[0.5, 0.7]
User 3	[0.6, 0.9]	[0.85, 0.95]
User 4	[0.8, 0.9]	[0.8, 0.9]
User 5	[0.9, 1.0]	[0.3, 0.5]

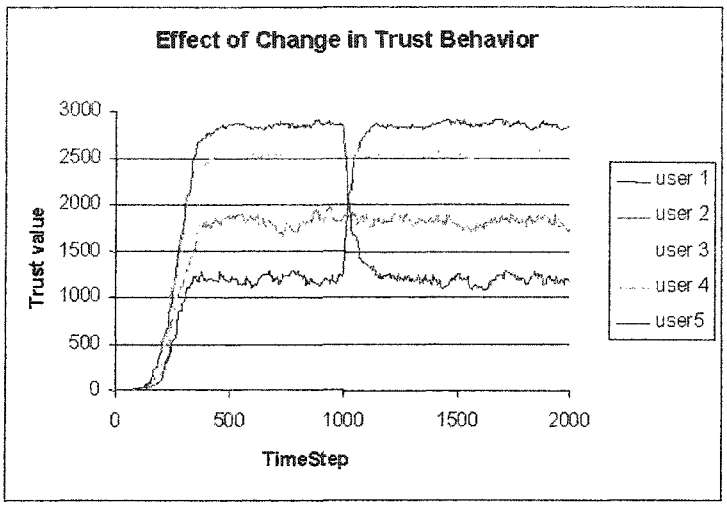


Figure B.6: Responsiveness of SPORAS: User 1, User 3 and User 5's trust behavior changes.

(3) Effect of deceptive transaction feedbacks: The effect and significance of transaction feedbacks is also applied to test how SPORAS trust system performs when there are deceptive feedbacks. SPORAS assumes that feedbacks are more reliable from the raters of high reputations than from the raters of low reputations. Given that the assumption holds, to study the impact of deceptive transaction feedbacks on a user's trust is actually to study the effect and significance of the raters' reputations. Two tests were conducted. The first test is to study the effect of a rater's reputation on a ratee's

trust value. The second test is to study the effect of the deviations of the raters' reputations on a ratee's trust value.

I simulated two ratees with same transaction size and transaction feedbacks. The only difference is that one ratee traded with a partner (a rater as well) of a constant transaction quality, 0.8. The other ratee traded with a partner of a constant transaction quality 0.5. Figure B.7 shows that the rater's reputation has a positive effect on the ratee's trust value in the trust building stage. The rater's high reputation also speeds up the ratee's trust value convergence. However, the positive effect disappears after the convergence, given that the ratee's trust behaviors keep unchanged.

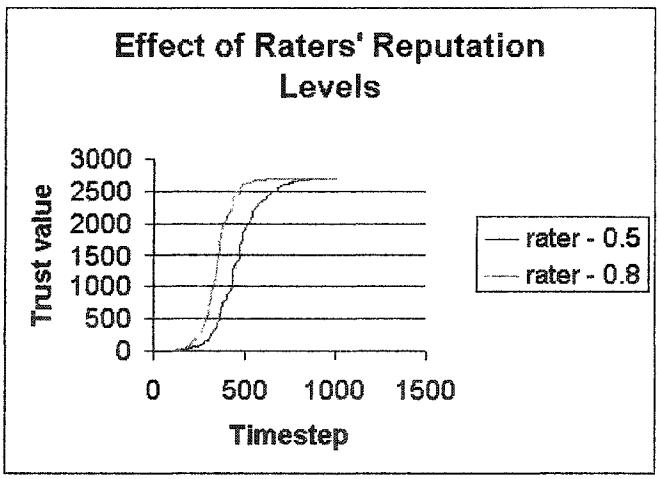
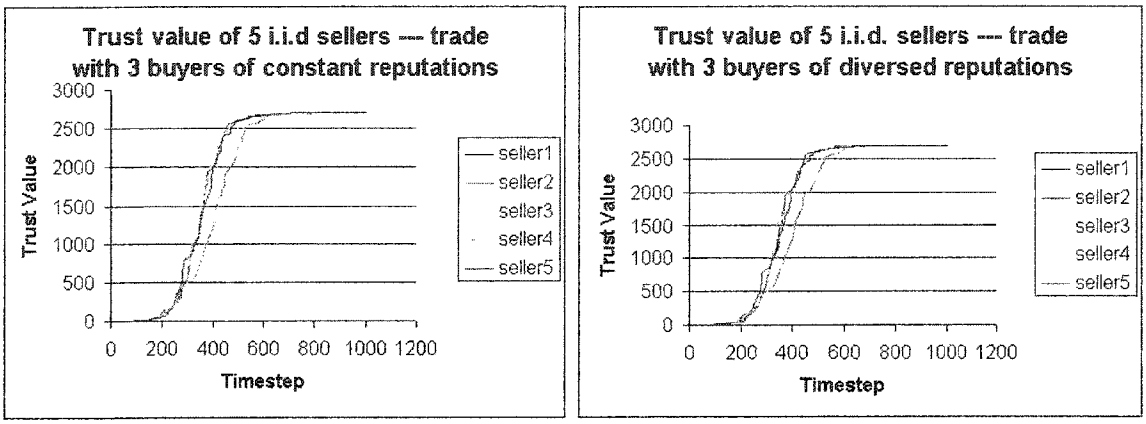


Figure B.7. Effect of raters' reputation levels on a ratee's trust value.

I also studied how the variation of raters' reputations would affect a ratee's reputation given that all the other factors (transaction size, and trust update intervals etc.) are the same. I simulated a SPORAS trust system of five sellers (ratees) and three buyers (raters). All the five sellers have constant online transaction feedbacks of 0.9. In the first scenario, the three buyers have constant transaction feedbacks of 0.8. In the second scenario, the three buyers have the same mean value of transaction feedbacks of 0.8, but

of different variations. The three buyers' transaction feedbacks are uniformly distributed in the range of [0.7, 0.9], [0.6, 1.0], and [0.8, 0.8] separately. Figure B.8 shows the effects of raters' reputations in both scenarios. The results show that the variances in the three raters' trust values do not have significant effect on the five ratees' trust values in the trust building stage and have no effect on their trust convergence speed.



(a) Raters of constant reputations (b) Raters of disperse reputations

Figure B.8. Effect of raters' reputations.

In summary, the findings show that SPORAS is fair, reliable and responsive in terms of (1) how it discriminates traders of different transaction qualities, (2) how large the correlations of traders' reputations are, (3) when raters' reputations take effect, and (4) how the system responds to changes.

REFERENCES

Axelrod, R. and W. D. Hamilton (1981). The evolution of cooperation. *science 211*, 1390.

Azzedin, F. and M. Maheswaran (2002). Evolving and managing trust in grid computing systems. In *Proceedings of IEEE Canadian Conference on Electrical & Computer Engineering (CCECE02)*.

Baum, L. E., T. Petrie, G. Soules, and N. Weiss (1970). A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *Ann. Math. Stat. 41(1)*, 164–171.

Bhattacharya, R., T. Devieney, and M. Pillutla (1998). A formal model of trust based on outcomes. *Academy of Management Review 23* (3), 459–472.

Cole, S. J. (1998). Testimony before the subcommittee on telecommunications, trade and consumer protection committee on commerce, u.s.house of representatives, washington, d.c. In *Better Business Bureau, www.bbb.org/alerts/cole.asp*.

Cornelli, F., E. Damiani, and S. Samarati (2002). Implementing a reputationaware gnutella servent. In *Proceedings of the International Workshop on Peer-to-Peer Computing*.

Daniani, E., S. Vimercati, S. Paraboschi, P. Samarati, and F. Violante (2002). A reputation-based approach for choosing reliable resources in peer-to-peer networks. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS02)*.

Dellarocas, C. (2003). Efficiency and robutstness of ebay-like online feedback mechanisms in environments with moral hazard. In *Working paper 170 of Massachesetts Institute of Technology*.

Gefen, D. (2000). E-commerce: The role of familiarity and trust. *Omega: The international Journal of Management Science 28* (6), 725–737.

Gupta, M., P. Judge, and M. Ammar (2003). A reputation system for peer-to-peer networks. In *Proceedings of the 13th ACM International workshop on Network and Operating Systems Support for Design Audio and Vedio (NOSSDAV03)*.

Ham, F. and I. Kostanic (2001). *Principles of Neurocomputing for Science and Engineering*. McGraw-Hill.

Jarvenpaa, S. L. and N. Tractinsky (1999). Consumer trust in an internet store: A cross-cultural validation. *Journal of Computer Mediated Communication 5* (2), 1–35.

Kamvar, S., M. Schlosser, and H. Garcia-Molina (2003). The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the twelfth ACM International World Wide Web Conference (WWW03)*.

Kreps, D. (1990). *A Course in Microeconomics*. Princeton University Press, Priceton, NJ.

Larson, H. J. (1984). *Introduction to probability theory and statistical inference*. PWS-Kent Publishing, Boston, MA. Montaner, M., B. L´opez, and J. Rosa (2002). Opinionbased filtering through trust. In *Sixth International workshop on Cooperative Information Agents (CIA02)*.

Maynard, J. S. and G. R. Prince (1973). The logic of animal conflict. *Nature 246*, 15–18.

Montaner, M., B. L´opez, and J. Rosa (2002). Opinionbased filtering through trust. In *Sixth International workshop on Cooperative Information Agents (CIA02)*.

Mui, L., A. Halberstadt, and M. Mohtashemi (2002). Notions of reputation in multi-agents systems: A review. In *Proc. AAMAS02*.

Mui, L., M. Mohtashemi, and A. Halberstadt (2002). A computational model for trust and reputation. In *Proc. 35th Hawaii International Conference on System Sciences*.

Mui, L., M. Mojdeh, A. Cheewee, and P. Szolovits (2001). Ratings in distributed systems: A bayesian approach. In *Workshop on Information Technologies and Systems (WITS01)*.

Pujol, J., R. Sang, and A. Halberstadt (2002). Extracting reputation in multi agent systems by means of social network topology. In *Proceedings of the first International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS02)*.

Rabiner, L. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of IEEE 77(2)*, 257–286.

Ratnasingham, P. and K. Kumar (2000). Trading partner trust in electronic commerce participation. In *Proceedings of the Twenty First International Conference on Information Systems*.

Resnick, P. and R. Zechhauser. Trust among strangers in internet transactions: Empirical analysis of ebay's reputation system. *working paper for the NBER workshop on Empirical Studies of Electronic Commerce.*

Resnick, P. and R. Zeckhauser (2000). Trust among strangers in internet transactions: empirical analysis of ebay's reputation system. In *NBER Workshop on Empirical Studies of Electronic Commerce Paper.*

Riggs, T. and R. Wilensky (2001). An algorithm for automated rating of reviewers. In *Proceedings of first ACM and IEEE Joint Conference on Digital Libraries (JCDL01).*

Rotter, J. (1967). A new scale for the measurement of interpersonal trust. *Journal of Personality 35.*

Rumelhart, D., G. Hinton, and J.L.McClelland (1986). A general framework for parallel distributed processing. *Parallel distributed processing: Explorations in the microstructure of cognition 1.*

Sabater, J. and C. Sierra (2001). Regret: A reputation model for gregarious societies. In *Proc. 4th Workshop on Deception, Fraud and Trust in Agent Societies.*

Sabater, J. and C. Sierra (2002). Reputation and social network analysis in multiagent system. In *AAMAS02.*

Sarwar, B., D. Karypis, J. Konstan, and J. Ridel (2000, October). Analysis of recommendation algorithms for e-commerce. *EC'00* .

Schafer, B. K., A. J. Konstan, and J. Riedl (1999). Recommender systems in ecommerce. In *Proceedings of ACM Conference on Electronic Commerce(EC99).*

Shafer Glenn. *A Mathematical Theory of Evidence.* Princeton University Press, Princeton, NJ, 1976.

Schafer, J. B., J. Konstan, and J.Riedl (2002). Meta-recommendation systems: User-controlled integration of diverse recommendations. In *Proceedings of ACM Conference on Information and Knowledge Management (CIKM02).*

Sen, S. and N. Sajja (2002). Robustness of reputation-based trust: Boolean case. In *AAMAS'02.*

Sen, S. and N. Sajja (2003). Selecting service providers from noisy reputations. In *AAMAS'03.*

Song, W. and V. V. Phoha (2004a). Neural network-based reputation model in a distributed system. In *IEEE Conference on E-commerce Technology (CEC2004).*

Song, W. and V. V. Phoha (2004b). Opinion filtered recommendation trust model in peer to peer network. In *Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS04)*.

Song, W. and V. V. Phoha (2004c). The HMM-Based Model for Evaluating Recommender's Reputation. In *IEEE Conference on Electronic Commerce East (CEC-East04)*, 2004.

Song, W., V. V. Phoha, and X. Xu (2004). An adaptive recommendation trust model in multiagent system. In *2004 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT04)*.

Cormen, T. H., C. E. Leiserson, R. Rivest, and C.Stein (2001). *Introduction to Algorithms*. the MIT Press and McGraw-Hill.

Mitchell, T. (1997). *Machine Learning*. McGraw-Hill.

Truett, L. J. and D. B. Truett (1987). *Economics*. Times Mirror/Mosby College Publishing.

Vassileva, J., S. Breban, and M. Horsch (2002). Agent reasoning mechanism for long-term coalitions based on decision making and trust. *Computational Intelligence 18* (4), 453–502.

Wang, Y. and J. Vassileva (2003). Beyesian network-based trust model. In *Proc. of IEEE/WIC International Conference on Web Intelligence (WI2003)*.

Williamson, O. C. (1993). Trust and economic organization. *Journal of Law and Economics 34*.

Xu, X., V. V. Phoha, and W. Song (2005a, March). Early detection and propagation mitigation of malicious mobile code. *43rd annual ACM Southeast Conference, Atlanta, GA*.

Xu, X., V. V. Phoha, and W. Song (2005b, July). Stochastic propagation modeling of self-replicating programs. *The 3rd International Conference on Computing, Communications and Control Technologies (CCCT '05), Austin, TX*.

Yu, B. and M. P. Singh (2001). Towards a probabilistic model of distributed reputation management. In *Proc. 4th Workshop on Deception, Fraud and Trust in Agent Societies*.

Yu, B. and M. P. Singh (2002). Distributed reputation management for electronic commerce. *Computational Intelligence 18* (4), 535–549.

Yu, B. and M. P. Singh (2003). Detecting deception in reputation management. In *AAMAS'03*.

Zacharia, G. and P. Mae (1999). Collaborative reputation mechanisms in electronic marketplaces. In *Proc. 32nd Hawaii International Conf on System Sciences*.

Zucker, L. (1986). *Production of Trust: Institutional Sources of Economic Structure*. JAI Press.