ABSTRACT

CHOO, EUIJIN. Information Assurance in Electronic Commerce Market. (Under the direction of Ting Yu and Min Chi.)

Information assurance is the practice of ensuring the quality of information users get. Given a large volume of available information, the problem of assuring information quality draw attentions in many applications including sensor networks, web service applications, and cloud computing. Above all, information assurance in electronic commerce market is critical because it is directly linked to monetary problems.

In this dissertation, we address information assurance in three different directions. First of all, a system should perceive and reflect users’ true interest to make sure users get accurate information. To do so, our first work studies the problem of discovering implicit communities in one of electronic commerce platforms, a recommender system. In particular, we identify meaningful implicit communities based on users’ reply patterns in a recommender system. Then, we show that incorporating discovered communities into a recommender system significantly improve recommendation when compared with approaches that only consider user-item relationships.

Despite the efforts to provide a great quality of information, malicious parties often exploit the vulnerability of systems and manipulate to achieve their goals. Therefore, it is needed to build a trusted manager to make a system resilient to attacks. Reputation is one of primary mechanisms for trust management in electronic commerce market. To help each system administrator pick a right reputation system for their own purpose, it is necessary to measure true resilience of each system. To do so, in our second work, we propose a framework, called COMPARS. COMPARS models attackers’ realistic behavior and measures resilience of each system considering adaptive nature of attackers. As a case study, we analyze and compare three reputation systems in terms of resilience against manipulations.

Nowadays, many users in e-commerce market refer to public opinions (e.g., reviews in Ebay) to make their decision. Accordingly, although e-commerce industries try to manage risks by building a resilient trusted manager, attackers often try to mislead users with fraudulent opinions, namely opinion spams. Our third work aims to detect such opinion spammers by exploring community structure built through users’ replying activities coupled with analyzing sentiments of user relationships in one of e-commerce platforms, a review-system. Then, we show that our community-based approach achieves both accuracy and reliability. In other words, the proposed approach is reliable against content manipulation, while achieving a comparable level of accuracy to traditional content-based approaches.
Information Assurance in Electronic Commerce Market

by
Euijin Choo

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Computer Science

Raleigh, North Carolina
2015

APPROVED BY:

__________________________  __________________________
Munindar Singh              Rada Chirkova

__________________________  __________________________
Ting Yu                     Min Chi
Co-chair of Advisory Committee  Co-chair of Advisory Committee
DEDICATION

To my beloved parents - Bongjoon Choo and Kyesun Han - and family.
Euijin Choo was born in Seoul, Korea. She earned dual B.S. from Dept. of Computer Science & Engineering, and Dept. of Mathematics at Korea University, and later she earned M.S. from Dept. of Computer Science & Engineering at Korea University as well. She had been working as a research assistant and a research staff in Computer and Communication Security (CCS) Laboratory of Korea University under the supervision of Dr. Heejo Lee. Now, she is a member of the Privacy and Trust In Open Systems Lab and Cyber Defence Lab working as a research assistant under the supervision of Dr. Ting Yu and Dr. Min Chi. She had also served as a teaching assistant in mostly computer security classes, and she is a student member of IEEE.

During her doctoral studies, Euijin received the Best Paper Award at the 29th Annual IFIP Working Conference on Data and Applications Security and Privacy in 2015. During her undergraduate, master and doctoral studies, Euijin received a few scholarships including Samsung Networks Academy Scholarship (sponsored by Samsung Networks), Brain Korea 21 Scholarship (sponsored by Korea Research Foundation), and Provost Fellowship (Sponsored by North Carolina State University). She is also an awardee of a few travel awards sponsored by conferences and workshops including UIUC Summer Workshop in Science of Security, Trusted Infrastructure Workshop, Computer and Communication Security Conference, and IEEE Symposium on Security and Privacy.

Since 2006, Euijin has conducted several research in the field of computer and network security including encryption, key management, mitigation of DDoS attacks, generic unpacking of malwares, resilience evaluation of reputation systems, and spam detection in public forums. Her research has been funded by various government organizations including Korea Small and Medium Business Administration, National Security Research Institute in Korea, National Science Foundation in USA, and National Security Agency in USA.

Her current research interest sits at the intersection of security and data analytics. More specifically, it includes malware detection, spam detection in public forums, and security and information assurance in e-commerce market.
ACKNOWLEDGEMENTS

I would like to thank all those people who have helped and inspired me throughout my doctoral studies. First and foremost, I am deeply indebted to my co-advisors, Dr. Ting Yu and Dr. Min Chi, for encouraging and guiding me during my research. They have always been supportive and considerate. Since the beginning of my doctoral studies, Dr. Ting Yu constantly encouraged me to develop research skills as well as English skills, and lead me in the right direction. He taught me how to conduct thorough research, offered insightful suggestions, and patiently guided me through all these years. Since 2013, Dr. Min Chi gave inspiring feedback to take a look at research problems from a variety of perspectives, helped improve my writing, and handed out thoughtful advice not only on research but also on my career path. She has always had enormous faith and confidence in my abilities. It would not have been possible to finish my dissertation without the supervision and the constant help of both of my advisors.

I am very grateful for having an exceptional thesis committee, including Dr. Munindar Singh and Dr. Rada Chirkova, and a written qualifying exam committee including Dr. Khaled Harfoush and Dr. Edward Gehringer. They provided insightful and valuable comments on my research and presentations, which helped me to examine my research objectively.

It had been great working with Dr. Robert Rodman, Dr. Xuxian Jiang, and Dr. Khaled Harfoush as a teaching assistant. I am certain that the invaluable teaching experiences with them will be significant assets in my future career as well.

I wish also to record my appreciation to Dr. Douglas Reeves and staffs of computer science department for their prompt help on a lot of administrative work.

I would like to thank my former advisor, Dr. Heejo Lee, for introducing me to security research and encouraging me to pursue Ph.D. study abroad.

I also thank my fellow labmates in Cyber Defense Lab and classmates at North Carolina State University for helpful discussions on my research and presentations. I thank Zach Jorgensen and Collin Lynch for proof reading my research papers. I especially wish to express my gratitude to my previous labmate, and collaborator, Dr. Younghee Park for being such a great mentor to inspire and to help guide my career path, and for working with me in interesting research problems. My special thanks also goes to my lifetime childhood friends, Korea University alumni, my dance friends, and all of my dear friends from North Carolina State University, University of North Carolina at Chapel Hill, and Wake Technical Community College. You always stand by me in good or bad times. Thank all of you for always being there to support me, to help me get through challenging times, and to share memories with me.

I was fortunate to be part of the NC State Dance Program, which gave me the opportunity to develop my creativity and to widen my intellectual horizon. A dance studio at Carmichael
Gym has always been the place where I get to know amazing people to spend enjoyable time with, and where I can relieve my stress.

Last but not the least, I am eternally grateful to my family for their endless care, invaluable support, and love. Their continuous support and belief in me were the driving force that pushed me forward to achieve my goals throughout the journey.

I am greatly blessed to have all these people on my side. You are the reason that I can keep my happy smiles.
# TABLE OF CONTENTS

**LIST OF TABLES** ................................................................. viii

**LIST OF FIGURES** ............................................................. ix

**Chapter 1 Introduction** ...................................................... 1
  1.1 Information Assurance in Electronic Commerce Markets .............. 1
  1.2 Contribution ............................................................... 4
  1.3 Outline ........................................................................... 6

**Chapter 2 Electronic Commerce Market** .................................... 7
  2.1 Recommender System ..................................................... 7
  2.2 Reputation System ........................................................ 8

**Chapter 3 Dataset** ............................................................... 11

**Chapter 4 Revealing and Incorporating Implicit Communities to Improve Recommender Systems** .................................................. 13
  4.1 Motivation .................................................................... 13
  4.2 Related Work ............................................................... 15
  4.3 Revealing Implicit Communities in Recommender Systems ........... 16
    4.3.1 Modeling Interactions in Recommender Systems .................. 17
    4.3.2 Randomness of Interactions in Amazon ........................... 19
    4.3.3 User Relationship ................................................... 22
    4.3.4 Revealing Implicit Communities in Recommender Systems .... 23
  4.4 Recommendation ........................................................... 25
    4.4.1 Recommendation with Discovered Communities ................. 25
    4.4.2 Comparison with Collaborative Filtering Methods ................ 27
    4.4.3 Combining A Social Recommender System with Collaborative Filtering Methods .................................. 31
  4.5 Summary ........................................................................ 33

**Chapter 5 COMPARS: Toward An Empirical Approach for Comparing the Resilience of Reputation Systems** .................................... 35
  5.1 Motivation .................................................................... 35
  5.2 Related Work ............................................................... 37
  5.3 Optimal Attack Strategies ............................................... 39
    5.3.1 Games in Reputation Systems .................................... 39
    5.3.2 Attack Tree ............................................................ 40
    5.3.3 Attack Tree with Sample Sequences ............................... 42
  5.4 COMPARS: A Framework for Comparison of Reputation Systems .... 45
    5.4.1 Overview of COMPARS ............................................. 45
    5.4.2 The Initial State Generator ........................................ 46
    5.4.3 The Transaction Manager .......................................... 47
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.4.4 The Reasoning Manager</td>
<td>49</td>
</tr>
<tr>
<td>5.4.5 The Evaluator</td>
<td>49</td>
</tr>
<tr>
<td>5.5 Experimental Results and Analysis</td>
<td>50</td>
</tr>
<tr>
<td>5.5.1 Materialization</td>
<td>50</td>
</tr>
<tr>
<td>5.5.2 Analysis of Existing Reputation Systems with COMPARS</td>
<td>54</td>
</tr>
<tr>
<td>5.6 Summary</td>
<td>61</td>
</tr>
<tr>
<td>Chapter 6 Detecting Opinion Spammer Groups through Community Discovery and Sentiment Analysis</td>
<td>62</td>
</tr>
<tr>
<td>6.1 Motivation</td>
<td>62</td>
</tr>
<tr>
<td>6.2 Related Work</td>
<td>64</td>
</tr>
<tr>
<td>6.3 Discovering Opinion Spammer Groups and Competitors</td>
<td>67</td>
</tr>
<tr>
<td>6.3.1 Stage 1: Building a General User Relationship Graph</td>
<td>67</td>
</tr>
<tr>
<td>6.3.2 Stage 2: Sentiment Analysis on User Relationships</td>
<td>69</td>
</tr>
<tr>
<td>6.3.3 Stage 3: Positive Relationship Graphs</td>
<td>70</td>
</tr>
<tr>
<td>6.3.4 Stage 4: Identify Spammer Candidates by Decomposing Positive Relationship Graphs</td>
<td>70</td>
</tr>
<tr>
<td>6.3.5 Stage 5: Building a Negative Relationship Graph with Discovered Communities and Their Negative Neighborhood</td>
<td>72</td>
</tr>
<tr>
<td>6.4 Experimental Results and Analysis</td>
<td>73</td>
</tr>
<tr>
<td>6.4.1 User Statistics</td>
<td>73</td>
</tr>
<tr>
<td>6.4.2 Verified Purchase Analysis</td>
<td>77</td>
</tr>
<tr>
<td>6.4.3 Analysis on Spammer Classification with a Linguistic Model-based Classifier</td>
<td>78</td>
</tr>
<tr>
<td>6.4.4 Analysis on Spammer Classification with a Spam Indicators-based Approach</td>
<td>83</td>
</tr>
<tr>
<td>6.4.5 A Comparison of Linguistic Model-based and Spam Indicators-based Classifiers</td>
<td>106</td>
</tr>
<tr>
<td>6.4.6 Analysis on Negative Relationships of Discovered Reviewers</td>
<td>106</td>
</tr>
<tr>
<td>6.5 Summary</td>
<td>107</td>
</tr>
<tr>
<td>Chapter 7 Concluding Remarks</td>
<td>109</td>
</tr>
<tr>
<td>7.1 Summary</td>
<td>109</td>
</tr>
<tr>
<td>7.2 Future Work</td>
<td>110</td>
</tr>
<tr>
<td>7.2.1 Extension of Recommender System Research</td>
<td>110</td>
</tr>
<tr>
<td>7.2.2 Finding Optimal Attack Strategies in Complex Systems</td>
<td>111</td>
</tr>
<tr>
<td>7.2.3 Studying Other Types of Spamming Activities</td>
<td>111</td>
</tr>
<tr>
<td>7.2.4 Studying Dynamics of Implicit Communities</td>
<td>112</td>
</tr>
<tr>
<td>7.2.5 Analyzing Economic Effects of Opinion Spams</td>
<td>112</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>113</td>
</tr>
<tr>
<td>Table</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>Table 3.1</td>
<td>Dataset</td>
</tr>
<tr>
<td>Table 5.1</td>
<td>List of parameters for experiments</td>
</tr>
<tr>
<td>Table 5.2</td>
<td>Profit setting</td>
</tr>
<tr>
<td>Table 6.1</td>
<td>Spearsman’s rank correlation coefficient in each category</td>
</tr>
</tbody>
</table>
## LIST OF FIGURES

| Figure 4.1 | In-degree and out-degree distributions of each user in Books and Electronics | 18 |
| Figure 4.2 | Incoming entropy and in-degree of each user in each category | 20 |
| Figure 4.3 | Outgoing entropy and out-degree of each user in each category | 21 |
| Figure 4.4 | Discovered communities by a proposed approach | 24 |
| Figure 4.5 | MAE in each category | 29 |
| Figure 4.6 | Coverage in each category | 31 |
| Figure 4.7 | MAE and Coverage of a hybrid approach in each category | 34 |
| Figure 5.1 | An attack tree | 41 |
| Figure 5.2 | The pseudocode for generation of an attack tree | 43 |
| Figure 5.3 | An attack tree with sample sequences | 44 |
| Figure 5.4 | The architecture of COMPARS | 45 |
| Figure 5.5 | A consumer’s behavior and the evolution of system states | 47 |
| Figure 5.6 | A provider’s behavior and the evolution of system states | 48 |
| Figure 5.7 | The required number of an attacker’s transactions with change in consumers’ feedback reliability rate | 55 |
| Figure 5.8 | The number of required transactions to reach a goal with two strategies | 56 |
| Figure 5.9 | The required number of an attacker’s transactions with different $\delta_T$ | 58 |
| Figure 5.10 | The required number of an attacker’s transactions with different $\delta_R$ | 59 |
| Figure 5.11 | The effect of the number of reasoning steps | 60 |
| Figure 6.1 | The general idea of the proposed approach | 67 |
| Figure 6.2 | Examples of user relationship graphs in Amazon | 69 |
| Figure 6.3 | Examples of positive interactions between two users who are defined to have a positive relationship by our approach in Amazon | 71 |
| Figure 6.4 | Examples of negative interactions between two users who are defined to have a negative relationship by our approach in Amazon | 72 |
| Figure 6.5 | The number of discovered reviewers in each category | 74 |
| Figure 6.6 | The average number of reviews (RNUM) in each category | 75 |
| Figure 6.7 | Positive vote ratio (PVR) in each category | 76 |
| Figure 6.8 | Verified purchase ratio (VPR) in each category | 77 |
| Figure 6.9 | Excerpt from review classified as 99.99999% spamminess by the linguistic model-based classifier | 79 |
| Figure 6.10 | Fake probability in each category | 80 |
| Figure 6.11 | ROC curve with different strengths as thresholds for classification | 81 |
| Figure 6.12 | Most significant features learned by Naive Bayes Classifier | 82 |
| Figure 6.13 | Contents similarity (CS) in each category | 84 |
| Figure 6.14 | Rating abused item ratio (RA) in each category | 85 |
| Figure 6.15 | Review duplicated items ratio (DUP) in each category | 86 |
| Figure 6.16 | Maximum one day review ratio (MOR) in each category | 87 |
| Figure 6.17 | First review ratio (FRR) in each category | 88 |
| Figure 6.18 | Early time frame ratio (ETF) in each category | 89 |
Figure 6.19  Deviated rating ratio (DEV) in each category ............... 90
Figure 6.20  Extreme rating ratio (EXT) in each category .................. 91
Figure 6.21  Review Burstiness (BST) in each category ................... 92
Figure 6.22  Interaction burstiness (IBST) in each category ............... 93
Figure 6.23  Contents similarity (CS) in each category .................... 94
Figure 6.24  Rating abused item ratio (RA) in each category ............... 95
Figure 6.25  Review duplicated items ratio (DUP) in each category .......... 96
Figure 6.26  Maximum one day review ratio (MOR) in each category ....... 97
Figure 6.27  First review ratio (FRR) in each category .................... 98
Figure 6.28  Early time frame ratio (ETF) in each category ............... 99
Figure 6.29  Deviated rating ratio (DEV) in each category ............... 100
Figure 6.30  Extreme rating ratio (EXT) in each category ............... 101
Figure 6.31  Review Burstiness (BST) in each category ................... 102
Figure 6.32  ROC curve with different strengths as thresholds for classification ...... 103
Figure 6.33  ROC curve without sentiment analysis ....................... 105
Figure 6.34  The number of negative relationships of spam reviewers in each category ... 107
Chapter 1

Introduction

1.1 Information Assurance in Electronic Commerce Markets

Our first research studies the problems of providing accurate information in recommendation context. Users in e-commerce markets (e.g., Amazon, Ebay) are often overwhelmed by a deluge of available information and alternative choices. Recommender systems are designed to enable users to make an efficient decision. These systems are required to identify the most relevant items to individuals to assure users get appropriate information. A number of recommender systems have been proposed in the literature [30, 66, 92, 96, 103, 104]. These systems operate by gathering individual users’ preferences and recommend items to them.

In recent years, researchers have begun to pay attention to the role of social connections in making recommendations [5, 11, 23, 38, 48, 71]. Such systems, so called social recommender systems, take advantage of users’ social relationship information to improve recommendation accuracy. Intuitively, when people have difficulty making a decision, they often ask their friends for advice, which, in turn, plays an important role in their final decision. Social recommender systems, while promising, are often hindered in practice. Existing social network systems such as Facebook are not designed for recommendations and thus contain many irrelevant relationships. Existing recommendation platforms such as Amazon do not permit users to establish explicit social relationships. In our first work we address these issues by focusing on the extraction of implicit and relevant social relationships among users based upon their review-reply patterns.

In this work we investigate several fundamental questions in the context of item recommendations on the Amazon platform: (1) Do meaningful implicit relationships exist in practice, and can we distinguish them from the meaningless relationships? (2) Given a pair of users, how can we quantify the strength of their relationship if any and identify any meaningful relationships? (3) Once meaningful implicit communities are discovered, will they have any relevance to recommender systems or, indeed, any relationship to the individual users preferences?
We first develop a random graph model that reflects users’ habitual review/reply behavior in a system. We then investigate on whether there exist any meaningful review/reply patterns among users by comparing their real review/reply behavior to the random graph model. Then we propose a probabilistic mechanism to quantify the strength of relationships and distinguish meaningful relationships from meaningless relationships. We finally show discovered relationships are indeed relevant to their item preference, and we can improve recommender systems by incorporating discovered communities into hybrid recommender systems.

Although e-commerce industries try to provide accurate information with recommender systems, malicious parties, so called opinion spammers, often attempt to manipulate marketplace with fraudulent opinions [21, 41]. One way to ensure users get correct information is to build a trusted manager and promote trustworthy users in decentralized e-commerce systems. Another is to demote opinion spammers by detecting and preventing them.

In decentralized e-commerce systems, users often have transactions with strangers without pre-existing knowledge of each other. Reputation is a primary mechanism to help users decide who to trust. Many reputation-based trust functions have been proposed in the literature [6, 16, 52, 54, 126, 128, 134]. However, picking the right trust function for a given decentralized system is a non-trivial task. One has to consider and balance a variety of factors, including computation and communication costs, scalability and resilience to manipulations by attackers. Although the former two are relatively easy to evaluate, the evaluation of resilience of trust functions is challenging. Most existing work bases evaluation on static attack models, which is unrealistic as it fails to reflect the adaptive nature of adversaries (who are often real human users rather than simple computing agents).

Our second research aims to study the resilience of a system. In this work we highlight the importance of the modelling of adaptive attackers when evaluating reputation-based trust functions, and propose an adaptive framework—called COMPARS—for the evaluation of resilience of reputation systems.

A particular trust function may be resilient to one type of attack yet vulnerable to another. For a fair evaluation of resilience, one thus has to compare the worst case of each trust function (i.e., the most effective attack of attackers to exploit the vulnerability of a given trust function). Theoretically, attackers would have optimal strategies to achieve the most effective attacks, which are likely to be different for different trust functions. Given the complexity of reputation systems, however, it is often difficult, if not impossible, to exactly derive the optimal strategy of an attacker. Therefore, COMPARS takes a practical approach that attempts to capture the reasoning process of an attacker as it decides its next action in a reputation system. Specifically, given a trust function and an attack goal, COMPARS generates an attack tree to estimate the possible outcomes of an attackers action sequences up to certain points in the future. Through
attack trees, COMPARS simulates the optimal attack strategy for a specific reputation function \( f \), which will be used to evaluate the resilience of \( f \). By doing so, COMPARS allows one to conduct a fair and consistent comparison of different trust functions.

As mentioned above, demoting opinion spammers by detecting them is another way to assure accurate information in e-commerce systems. To do so, in our third work, we investigate on detection of opinion spammer groups in one of e-commerce platforms, a review system. Most existing approaches typically build pure content-based classifiers, using various features extracted from review contents [25, 27, 68, 83, 90, 91, 120, 129]. One major limitation of such approaches is that spammers can superficially alter their review contents to avoid detections without affecting the effectiveness of opinion spams. To tackle such a problem, we take a novel approach that identifies opinion spammers by focusing on user relationships, instead of review contents, built through interactions.

We first introduce a new angle of spamming behavior (i.e., boosting rankings to spread spams by supporting each other) and propose a method to find such spammer groups. In our first work we reveal the existence of implicit communities among users based upon the patterns of their interactions. In this work we further explore the community structures to distinguish spam communities from non-spam communities with sentiment analysis on user interactions. Concretely, we propose to find communities built through abnormally non-random positive interactions based on the intuition that spammers need to form artificial communities to make their opinions influential.

In this work we investigate four key research problems and our approach can generally be divided into four steps to address those four problems: (1) What is a random/non-random relationship? (2) What is an abnormally non-random relationship, and how can we quantify the level of abnormalities of relationships? (3) What is a positive relationship? (4) How to detect spam communities?

We first build general user relationship graphs representing how users interact with one another. In particular, we employ the definitions of relationships and their strengths proposed in the first work to define random/non-random relationships. And we use the definition of the strength of relationships to represent the level of abnormalities of relationships. We thereby obtain a series of user relationship graphs with different strengths. Then, we derive the sentiment of each relationship by aggregating sentiments of all responses between any two users. We then extract positive relationship graphs from the general user relationship graphs to capture artificial boosting behavior. More specifically, motivated by link-based web spam detection, we focus on strongly connected communities in positive relationship graphs. Finally, we analyze extracted strongly positively connected communities and their negative connections to find opinion spammer groups and their competitors.
Through extensive experiments over a dataset collected from Amazon, we found that the discovered strong positive communities are more likely to be opinion spammer groups and their reviewing behavior deviates greatly from that of non-spammers. Specifically, their review contents present strong spam indicators, according to existing content-based classifiers. Our results also show that our approach is comparable to the existing state-of-art content-based classifier, meaning that our approach can identify spammer groups reliably even if spammers successfully alter their contents.

1.2 Contribution

In this dissertation we contribute to the research of ensuring information quality in electronic commerce markets by addressing three issues. (1) In the first work we ensure users get correct information by focusing on providing better recommendation through discovering and incorporating implicit communities in one of e-commerce platforms, a recommender system; (2) In the second work we study the resilience of a system to make sure a trusted manager work correctly given attackers’ manipulation. To do so, we propose an adaptive framework to model attackers’ behavior and to evaluate true resilience of different systems; (3) In the third work we propose a community-based approach to detect attackers in one of e-commerce platforms, a review system.

Specifically, our main contributions of the first work are summarized as follows.

• We develop a random graph model that represents users as vertices and reviewer-commenter interactions as edges. The graph allows us to distinguish between random and non-random interactions and to study the pattern of real user interactions. If all user interactions occur by chance then real user data will closely match the random graph. If, however the user data deviates substantially from this model then meaningful social relationships are likely to be found.

• We develop a quick global indicator that can be used to highlight when, and to what extent, interactions among users deviate from the random model. We observe that interactions in subjective item categories (e.g. book and movie reviews) tend to deviate substantially from the random model while more objective categories (e.g. electronics and tools) tend to have more random interactions.

• We propose a probabilistic mechanism to quantify the strength of social connections between a pair of users based upon the random model which we use to construct a series of social graphs.
• We apply existing social recommender systems using the extracted implicit social graphs and show that the resulting recommendations are more accurate than those produced by existing collaborative-filtering methods.

Our main contributions of the second work are summarized as follows.

• We propose a general methodology to unbiasedly and practically evaluate the resilience of trust functions by considering the adaptive nature of realistic attackers. The essential principle is to compare different trust functions based on their worst cases, i.e., to examine their vulnerability to manipulation when attackers adopt the optimal strategy specific for each trust function.

• We present the design of a highly configurable platform for trust function resilience evaluation. Through a set of well-defined interfaces, the platform allows us to plug in key modules of a reputation system, including honest user behavior models, initial system environment parameters, attack objectives and trust functions, so that we can study the resilience of a reputation system under various configurations. Once these basic modules are provided, the platform will automatically approximate the optimal strategies for an attacker to reflect the true resilience of the system (i.e., the worst case scenario).

• As a case study, we use COMPARS to compare several influential trust functions, including EigenTrust [54], PeerTrust [126] and TNA-SL [52], and observe their differences in terms of resilience to malicious manipulations.

Our main contributions of the third work are summarized as follows.

• We propose a general unsupervised hybrid approach that is based on user interactions coupled with sentiment analysis. The intuition behind our idea is that users who have strong relationships with another can favor or disfavor each other’s opinions. To the best of our knowledge, this is the first attempt to identify opinion spammer groups through analyzing users’ interactions rather than their review content. A key advantage of our approach is that it can detect opinion spammers even when spammers manipulate review contents and traditional review content-based approaches fail.

• We introduce a new angle of collusive spamming behavior that opinion spammers deliberately build strongly positively connected communities to make their own opinions influential. We thus propose to explore community structures and a strength of relationships (i.e., how much the relationships are likely to be built intentionally) as spam indicators.
• We run extensive experiments over a dataset collected from Amazon to evaluate the effectiveness of the proposed approach. Our experiments show that even though our community-based approach differs markedly from pure content-based approaches, it reaches the same level of accuracy as the state-of-art content-based approach targeting at Amazon spams.

1.3 Outline

The remainder of the thesis is organized as follows. Chapter 2 presents abstract models of two types of e-commerce platforms (i.e., recommender systems and reputation systems) and introduce some basic concepts and notations used throughout this thesis. In Chapter 3, we describe how we collected data for our experiments used in our first and third work, and present a few characteristics of the dataset. In following chapters 4, 5, 6, we present three proposed work, respectively. Specifically, Chapter 4 introduces the first work to reveal and incorporate implicit social communities to improve recommender systems. Chapter 5 offers the second work to evaluate the resilience of reputation systems. Chapter 6 provides the third work to detect opinion spammer groups through community discovery and sentiment analysis in review systems. Finally, we conclude this thesis and lay out future work in Chapter 7.
Chapter 2

Electronic Commerce Market

2.1 Recommender System

A recommender system analyzes large datasets to help users make personal decisions. The systems apply users’ implicit or explicit preferences to filter the available items, which may include digital contents (e.g., webpages or video clips on Youtube), products (e.g., kitchenware), or people (e.g., potential friends in social networks) [99].

In general, there are two components in a recommender system: users and items. Users consume and evaluate items and the system recommends items to users. Both users and items have a set of system-specific attributes that are relevant to recommendations. Hulu and Last.fm, for example, classify items by artist and genre while other systems track demographic information and purchasing history.

In our research we focus on two different user actions: reviewing items, and commenting on reviews. Reviewing occurs when a user posts her evaluation/opinion about an item to the system. Commenting occurs when a user replies to an prior comment or review. Both reviews and replies may take a variety of forms including assigning star ratings and writing text. A user may thus take one of two roles in any relationship: reviewer and commenter.

Every time that a commenter replies to an existing review we consider that to be an interaction with the reviewer. We define implicit social connections based upon these reviewer-commenter interactions in Section 4.3. Note that in a recommender system, user $u_1$ can leave a comment, say $c_1$, on an existing comment, say $c_2$ by another user $u_2$, which generates threads of comments. In such a case, the interaction would be between $u_1$ and $u_2$. A user can also post multiple comments on the same review/comment in a system. Such actions often occur when a user wants to have a more detailed discussion on the item, or seek additional information.

Let us assume that two users like one item in common by chance so that they had a lot of discussions on the item (i.e., generating threads of comments, or post multiple comments to one
Although they had a lot of interactions with each other, it does not mean they have a social relationship (e.g., socially connected or share the same item preference in general) with each other. Instead, those users are only interested in the specific item. In the following, we thus only consider the interaction between a reviewer and a commenter, not between a commenter and another commenter; and we count multiple comments on the identical review/comment as a single interaction.

### 2.2 Reputation System

Reputation systems help users estimate the trustworthiness of other parties in a decentralized system. By decentralized, we mean that entities are autonomous; there is no single centralized authority that asserts the trustworthiness of entities, or makes decisions on the appropriate actions of an entity. This concept is orthogonal to the underlying exchange structures of a system including centralized (e.g., e-commerce systems) or decentralized (e.g., a peer-to-peer file sharing systems). We assume that entities in a decentralized system interact with each other through transactions. Transactions are not limited to monetary interactions; they also include activities such as retrieving information from a website, downloading files from a peer, and etc. We assume that a transaction is uni-directional, i.e., given a transaction, there is a clear distinction between a service provider and a service consumer.

In general, reputation systems share a common structure[42], typically modeled as a 5-tuple \((C, P, R, F, A)\), where \(C\) is a set of service consumers, \(P\) is a set of service providers, \(R\) is a set of feedbacks, \(F\) is a trust function, and \(A\) is a set of actions. We describe the five components in detail below.

**Service consumers.** A service consumer is an entity who seeks services from a decentralized system, such as a buyer in an e-commerce market and a downloader in a file-sharing system. Each consumer is associated with a profile, which is a set of properties that are relevant to reputation management. For example, a profile may include the time a consumer joins the system and demographic information. A service consumer may have a set of services that it would like to get from the system at a certain time, but such information is not likely to be publicly known. Hence, we do not explicitly model it as a part of a consumer’s profile. Instead, it is modeled by a consumer behavior model, which is essential to reason about the evolution of a reputation system.

**Service providers.** A service provider is an entity who offers a set of services that may be requested by consumers (e.g., sellers in an e-commerce market and uploaders in file-sharing applications). Usually, the set of services offered by a provider is public (e.g., in ebay we can see all the items a seller is selling, but we do not know what items a buyer may
Therefore, the services offered by a provider is a part of its profile. Its profile may also have other similar properties to that of consumers. Note that the set of consumers and providers may not be disjoint; for instance, an entity may be a consumer in one transaction and a provider in another.

Feedbacks. A feedback $\gamma$ for a transaction takes the form $(c, p, i, r, t)$, meaning that consumer $c$ received service $i$ from provider $p$ at time $t$, and its rating is $r$. We leave the format of the rating opaque as it is application specific. In many systems, it is a single numerical/categorical value (e.g., 0 to 5 stars, or from poor to excellent). In others, it may instead be a vector that reflects a transaction’s quality from multiple aspects, e.g., price, product quality, and responsiveness of customer service.

Trust function. If an entity $a$ wants to evaluate the trustworthiness of another entity $b$, then we call $a$ and $b$ the source and destination of the trust evaluation, respectively. A trust function takes a source, a destination and a set of feedbacks, and returns the destination entity’s trust score. Similar to feedbacks, the format of a trust score is also opaque, and depends on the specific trust function. Most trust functions return a single numerical value as a trust score, while some others advocate returning a vector of numerical values [132], each corresponding to an aspect of the destination.

While most trust functions in the literature are subjective (i.e., from the point of view of different sources, the same destination may have different trust scores), some functions are objective (or global), meaning the trust score of an entity does not depend on the source. Nevertheless, the above modeling of trust functions is general enough to capture both types of functions.

Actions. A system allows a set of actions that entities can take. For example, a provider may list a set of services that are available to others; a consumer may choose to start a transaction with a provider; a consumer may post a positive or negative feedback regarding a transaction; a provider may provide good or poor transactions intentionally or unintentionally, etc.

The set of actions essentially defines the capabilities with which an attacker can manipulate a system, and is highly system-specific. For instance, many e-commerce systems have mechanisms to ensure that an entity cannot post a feedback unless it was indeed a consumer in the transaction. Other systems, such as online rating systems, cannot ensure whether or not an entity has direct experience with a service before rating it. As another example, some systems require a user to present some real-world credentials (e.g., credit cards) before creating an account, to circumvent Sybil attacks[16]. Many other systems
allow free entry, so that attackers may create multiple accounts and launch coordinated manipulation through these accounts.

As discussed earlier, an adaptive attacker does not stick with a fixed strategy to game a reputation system. Instead, it would evaluate the possible consequences of actions available at any given time, and decide which action is the best to achieve its goal. Note that the consequence considered might not be just the immediate ones. Instead, it is often desirable to consider an action’s long term impact in order to identify the best action at present. For example, cheating at a transaction at present might give the attacker an immediate payoff, but it may dramatically hurt its trustworthiness such that no other consumers will come to the attacker for service in the future, which may not be desirable for achieving its goal (e.g., get a certain amount of profits in the shortest time). Thus, in terms of the overall long-term payoff, it may not be the best action. To model this reasoning process, we introduce the concept of the state of a reputation system and its transition.

The state of a reputation system is a 3-tuple \((C, P, R)\), consisting of the set of consumers, the set of providers, and the set of feedbacks in the system. When an action occurs in a system, its state will change accordingly. For instance, when a new user joins the system, \(C\) is updated; when a provider lists a new service, its profile is updated; \(R\) is updated when a new feedback is issued. Let \(S_t\) be the system state at time \(t_0\). After an action \(a\) happens at time \(t_1\), the system state transitions to \(S_{t_1}\), denoted \(S_t \xrightarrow{(a,t_1)} S_{t_1}\). Given a sequence of actions \(X = (a_1,t_1), \ldots, (a_i,t_i)\), we denote the transition as \(S_{t_0} \xrightarrow{X} S_{t_e}\).
Chapter 3

Dataset

This section explains the dataset used in this study. Specifically, we describe how we collected data for our experiments, and present a few characteristics of this dataset. For our first and third work, we crawled data from one of the most popular e-commerce systems, Amazon, and had experiments over the data. To do so, we have implemented a data crawler with Perl. The crawler crawled information about items, reviews for items, and comments for reviews. Information about an item includes an item identifier and the category of the item as assigned by Amazon (e.g., Music, Movie). Information about a review includes its target item (i.e., the item the review is about), its reviewer (a user identifier assigned by Amazon), and its rating. Information about a comment to a review includes its commenter (a user identifier assigned by Amazon) and a target review about which the comment is.

To collect different categories of items, we employed a breadth-first search of Amazon starting from a specific item in each category. We first collected information about the item and every review for the item. We crawled the item identifier and a category that the item belongs to. And then, we got every review for the item. For each review, we got a reviewer’s user identifier and its rating that ranges from 1 to 5. After finishing crawling one item, the crawler proceeded to each reviewer’s user page. On the reviewer’s user page, we can find all of his reviews. The crawler thus grabbed all items about which the reviewer wrote reviews. Then, the crawler proceeded to each item page to perform the same breadth-first crawling. While doing so, if some commenters posted comments to a specific review, the crawler further proceeded to the review page. On the review page, we can find all comments about the review. For each comment, the crawler grabbed its commenter’s user identifier and proceeded to the commenter’s user page.

We will discuss different characteristics of each item category in the first work. To be specific, we will show that different characteristics of subjective and objective item categories. To do so, we performed experiments on four categories: two subjective categories (Books and Movie) and two objective categories (Electronics and Tools). We additionally use the integrated dataset of
four categories to investigate a broader impact across multiple categories. We will refer to the cross-category dataset as Across in this thesis. The details of the dataset are summarized in Table 3.1.

<table>
<thead>
<tr>
<th>Category</th>
<th>#items</th>
<th>#reviews</th>
<th>#comments</th>
<th>#reviewers</th>
<th>#commenters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Books</td>
<td>116,044</td>
<td>620,131</td>
<td>533,816</td>
<td>70,784</td>
<td>164,209</td>
</tr>
<tr>
<td>Movie</td>
<td>48,212</td>
<td>646,675</td>
<td>201,814</td>
<td>273,088</td>
<td>72,548</td>
</tr>
<tr>
<td>Electronics</td>
<td>35,992</td>
<td>542,085</td>
<td>128,876</td>
<td>424,519</td>
<td>72,636</td>
</tr>
<tr>
<td>Tools</td>
<td>22,019</td>
<td>229,794</td>
<td>32,489</td>
<td>151,642</td>
<td>21,977</td>
</tr>
<tr>
<td>Across</td>
<td>222,267</td>
<td>2,038,685</td>
<td>896,995</td>
<td>901,812</td>
<td>295,118</td>
</tr>
</tbody>
</table>

Table 3.1: Dataset

Generally speaking, it is hard to get real opinion spams datasets, because opinion spams look so similar to the real reviews. However, authors of [90] presented publicly available dataset for opinion spams. We thus additionally used the opinion spam datasets for our third work. We have obtained the 800 reviews from authors of [90]. In their dataset, 400 truthful reviews were mined from 20 most popular hotels on TripAdvisor in the Chicago area. 400 opinion spams are gathered for those same 20 hotels using Amazon Mechanical Turk (AMT) crowdsourcing. The details of this dataset can be found in their paper [90]. We will also generate ground truth opinion spammers datasets based on Arjun et al.[82]’s classifier.
Chapter 4

Revealing and Incorporating Implicit Communities to Improve Recommender Systems

4.1 Motivation

Online users are often overwhelmed by a deluge of available information and alternative choices. Recommender systems are designed to enable users to channel this flow and to make efficient choices, identifying personally-relevant items in an otherwise unending storm. These systems gather individual users’ preferences and then use that information to recommend appropriate items to them. These items can be any object that is consumed or evaluated by users such as objects for sale (e.g. Amazon or Ebay listings), or sources of information (e.g. news articles on Reddit or Boing Boing).

A number of recommender systems have been proposed in the literature [30, 66, 92, 96, 103, 104]. In recent years, researchers have begun to pay attention to the role that social connections can play in making recommendations [5, 11, 23, 38, 48, 71]. In general, people who have difficulty making a decision often ask friends or colleagues for advice which, in turn, plays an important role in their final decision. Social recommender systems have been shown to be more effective for some applications than collaborative filtering approaches that focus solely on user-item relationships. The key challenge when applying social systems, however, is to determine which social connections should be employed. Explicit social networks may often be ineffective because

---

This chapter has been published in Proceedings of the fifteenth ACM conference on Economics and computation, pp. 489-506. ACM, 2014. [20]
established friendships are frequently irrelevant to personal preferences. Two individuals may
friend one-another on Facebook because they have taken the same computer security class, but
that relationship has no bearing on their taste in film. By the same token, many influential
application platforms that would benefit from social recommendations such as Amazon, CNet,
and Urbanspoon, do not permit users to state explicit relationships and it is difficult if not
impossible to import relationships from other sources.

Our goal in this research is to address these limitations by determining how we can identify
meaningful social relationships through user connections on existing, non-social, platforms. We
have observed that many influential recommender systems such as Amazon and Yelp permit
users to both post item reviews and to respond to prior comments. Users may reply to a
review for a variety of reasons (e.g. signifying agreement or disagreement or seeking additional
information). From these connections we can extract implicit social relationships which can in
turn be extended to implicit communities. In the work we will focus on extracting relationships
of this type from the Amazon platform.

In this work we consider the act of replying to a user to be an interaction between two users:
the reviewer who authored the initial review; and the commenter who authors the reply. While
it is initially tempting to treat each such interaction as a sign of a meaningful relationship,
most such replies occur more by chance. Users will frequently browse for an item of interest,
read a review, and post a simple comment without considering the author of the original post.
Thus most such interactions can be modeled as a random process. It is necessary to filter out
these random interactions in order to provide high-quality, personalized, guidance. Our goal is
thus to develop a rigorous and systematic mechanism to identify significant and non-random
connections between users based upon their reviewer-commenter interactions which can then
be used to improve the quality of existing recommendation systems.

There are several fundamental questions that need to be investigated in this work: (1) Do
significant non-random connections exist in practice, and can we separate them from the
meaningless relationships? (2) Given a pair of users, how can we quantify the strength of their
relationship if any and identify any meaningful connections? (3) Once an implicit community
is discovered, will it have any relevance to recommender systems or, indeed, any relationship
to the individual members’ preferences?

We begin by proposing a probabilistic approach to measure the strength and randomness
of reviewer-commenter interactions. We will then apply this measure to show that strong non-
random relationships exist between Amazon users. We call these extracted connections implicit
social relationships as opposed to the explicit social relationships found in typical social networks
such as Facebook. We will then show that such implicit relationships can provide a direct
indication of mutual interests, and can thus be used to substantially improve recommendation
accuracy over existing collaborative filtering techniques that only take into account user reviews of common items.

4.2 Related Work

Many recommender systems have been proposed in the literature [2, 58, 92, 96, 99, 104]. They fall into two general categories: content-based recommendations, and collaborative filtering [2, 30, 40, 66, 103, 104].

Content-based approaches [30, 81, 92] recommend an item by comparing it to items in the user’s personal history. For example, Hulu, populates a list of “You may also like...” shows by identifying actors, directors, and genres that the user has seen before.

Wartena et al. proposed a method to recommend TV-broadcasts [123]. In particular, they represented items through keywords, and investigated different types of keywords including manually extracted ones and automatically extracted ones for better recommendation.

Mooney et al. proposed a book recommender system, LIBRA that utilizes a Bayesian learning algorithm and information extraction technique without any information about other users [81].

Collaborative filtering approaches, on the other hand, seek to identify users that share common items and then provide recommendations based upon these matching peers [40, 45, 104]. The key observation is that two users who have previously selected a number of items in common are likely to purchase similar items in the future. Amazon, for example, uses collaborative filtering to populate the list of items entitled: “Customers Who Bought Items in Your Recent History Also Bought”.

Collaborative filtering approaches can also be divided into two sub-categories: (1) memory-based and (2) model-based [45, 60, 103]. Memory-based approaches predict the unknown ratings by computing the similarity scores between users [95, 103, 130]. Model-based approaches, on the other hand, predict the unknown ratings by building probabilistic models [13, 43, 60].

While both content-based and collaborative filtering-based approaches have been used widely they both suffer from data sparsity and cold-start problems [38, 127]. The former arises from the fact that users have often selected and commented on a small number of items thus proving little information for matching algorithms to go on. The latter problem arises from the fact that new users have no history and thus we have a small amount of data on which to base our recommendations. Social recommender systems have been put forward as a candidate solution to these problems based upon the belief that advice from friends plays an important role in decisionmaking [5, 11, 23, 38, 70, 71]. For example, Liu et al. incorporated social network information into collaborative filtering methods to improve recommendation effectiveness [67]. They first collected data about users’ preference rating and their social relationships in online social
networks. Their approach first randomly chose 30% of each user’s ratings to generate a training dataset. Then, they computed similarity between each user and her friends in the training dataset. Finally, they recommended unknown ratings by aggregating the rating information of most similar friends.

Massa et al. proposed a trust model using a trust network in Epinions to predict the trust value [74, 75]. They inferred trust in unknown users by propagating over the trust network.

While social recommender systems have improved accuracy and better handle the data sparsity problem [2], explicit social information is not always available in recommender environments. Connecting recommender systems with social networks, however, raises privacy issues surrounding personal information that is neither relevant to recommendations nor should be disclosed [72]. Furthermore, there is no guarantee that friends always share similar interests [71] and the value of this information is degraded further in systems that combine near and distant relationships [38]. This may explain why it has been shown that employing explicit social information in recommender systems for news articles actually lowers the prediction accuracy [108].

With that in mind we propose a method to identify implicit social relationships in recommender systems by analyzing two different type of user actions: posting reviews, and replying to them. We will show that the use of these implicit relationships, both immediate and distant, improves the accuracy recommendation systems over existing collaborative-filtering approaches.

While the importance of social recommendations has been studied before, to the best of our knowledge no prior work has been done on the use of implicit relationships arising from direct interaction in recommender systems. Several papers take into account different types of interactions in news sites [61, 62, 105]. However, these studies focused primarily on extracting the discussion structure (e.g. word counts and sentence lengths) and location of replies [105]; or on evaluating the content of reviews and replies [61, 62, 105]. None of them have examined the relationships between users. Rather they focus on novel techniques to combine content-based and collaborative filtering approaches and consequently, they inherit the limitations of traditional approaches. Our implicit social structures, by contrast, help to improve upon existing systems and address these limitations. More importantly, the use of implicit social structures allows us to control the trade-offs between improving recommendation accuracy and sensitivity to these general limitations.

4.3 Revealing Implicit Communities in Recommender Systems

Our goal is to find implicit yet meaningful relationships among users in a recommender system. To do so, we first explain a typical model of user interactions in the system in Section 4.3.1. Then, based on the model, we present a global measure to study patterns of the interactions
in Amazon as an example of real applications in Section 4.3.2. To be specific, we will compare interactions among real users with the typical model of interactions to capture the difference between random/non-random interactions. Finally, in Section 4.3.3, we explain how we define a user relationship and a community, and present implicit communities found in Amazon.

4.3.1 Modeling Interactions in Recommender Systems

Let us present a typical scenario about a user’s activity in a system without explicit social relationships (e.g., Amazon). Let us assume that user $u$ has interest in item $p$, and $u$ found user $v$’s review about $p$ interesting. $u$ wants to have discussion with $v$ about $p$, so $u$ posts a comment to $v$’s review. Although $u$ left a comment on $v$’s review, from this instance it does not indicate that there is any special connection between $u$ and $v$. Instead, it is more likely that this interaction happens in a random manner: $u$ does not know $v$ before and does not specifically look for $v$’s reviews either. $u$ simply bumps into $v$’s review by chance during browsing. Therefore, a typical interaction between a commenter and a reviewer can be modeled as a random process. Then, intuitively, a special connection exists between a pair of users only if their interactions happen beyond random acts.

The next question is then how to determine whether two users have non-random interactions. A compelling measure is to look at the number of interactions between them. Intuitively, the more interactions between them, the less likely they happen randomly. However, purely looking at the number of interactions is not sufficient as it affected by a lot of factors. For example, some user $u$ may have a tendency to engage in discussions with others. So the mere fact that $u$ has many (say 10) interactions with a reviewer does not necessary mean the reviewer is anything special to $u$ (as it is possible $u$ may have similar number of interactions with many other reviewers in a random fashion). On the other hand, if $u$ rarely issues comments to others, then it may indicate strong and some unusual connections between $u$ and $v$ if $u$ comments on 10 of $v$’s reviews.

To distinguish random/non-random interactions, we first present a model to capture random interactions of users. We represent users and their interactions as a directed multigraph $G = (U, E)$ in which $U$ represents users (vertices) and $E$ represents interactions (edges). Since an interaction is defined as an action for a commenter to post a comment on a reviewer’s specific review, an interaction has a direction from a commenter to a reviewer.

Each edge $\vec{e}_{uv}$ is a 2-tuple $(u, v)$, where $u$ is a commenter and $v$ is a reviewer. Since we count multiple comments from the same user $u$ on the same review of user $v$ as a single interaction from $u$ to $v$, the number of edges from $u$ to $v$ is in fact the number of different reviews of $v$ to which $u$ post comments. A user as a commenter in a graph has outgoing edges and a user as a reviewer has incoming edges. An out-degree of commenter $u$ is the total number of edges
from $u$ to other users and an in-degree of reviewer $v$ is the total number of edges from other users to $v$. A total-degree of a user is the sum of out-degree and in-degree of the user. Since the in-degree of $v$ is the number of comments from other users to $v$, it essentially indicates how popular $v$ is to get comments (i.e., how interesting $v$’s reviews tend to be). On the other hand, the out-degree of $u$ is the number of comments from $u$ to other users, which indicates the tendency of $u$ to write a comment (i.e., how much $u$ is willing to comment). Fig.4.1 shows the in/out-degree distributions of users in each category. The X-axis represents in/out degree and the Y-axis represents corresponding probabilities. The in/out-degree distributions in different categories were similar, so we only present the distributions in two categories.

In general, the in-degree and the out-degree represent a user’s personal tendencies as a commenter and a reviewer, respectively. For example, some user may be an influential reviewer whose reviews are read by many because of good review qualities; while some user seldom write reviews, but read a lot or sometimes post comments on the existing reviews. To characterize those different nature of individuals, it is needed to examine a user’s both tendencies as a commenter and a reviewer separately. We model such user’s different tendency as an outgoing probability and an incoming probability. The outgoing probability of $u$ is the probability that $u$ generates outgoing edges and an incoming probability of $v$ is the probability that $v$ gets incoming edges. If all the interactions in a recommender system are not guided by any special relationship.
between users, interactions from $u$ to $v$ happen randomly depending on $u$’s outgoing probability and $v$’s incoming probability. Then, we can represent all users’ interactions as a random graph in which edges (i.e., interactions) are created following the incoming/outgoing probability of each user.

Let $x_{uv}$ denote an event for an edge $\overrightarrow{e_{uv}}$ to form and let $P(x_{uv})$ be a probability that $x_{uv}$ occurs. We assume that a random event (one interaction) occurs or does not occur with an equal probability, and thus we can model it as a Bernoulli trial [78].

Given a graph $G = (U, E)$, let us assume that each user $u \in U$ has its in-degree $I_u$ and out-degree $O_u$, and the total number of all edges is $n$. We generate $n$ events (i.e., $n$ Bernoulli trials) to build a random graph $G_r$ drawn from the given degree distribution (i.e., incoming and outgoing probabilities). At each trial, we randomly choose and connect a pair of users $(u, v)$ from $U$ with a probability $P(x_{uv}) = (O_u/n) \times (I_v/n)$. Therefore, one edge is added at each trial. After $n$ trials, we have a random graph $G_r = (U, E')$ in which the total number of all edges is $n$, each user’s degree distribution is the same as $G$; but edges are randomly generated and the number of edges between each pair of users will be different from $G$.

4.3.2 Randomness of Interactions in Amazon

To study interactions among real users, we present our observations on Amazon dataset and compare it with the random model presented in Section 4.3.1.

We construct a random graph by generating edges randomly with the given incoming/outgoing probabilities of users in the real dataset. Then, we compare a graph in the real dataset with the corresponding random graph to see how interactions are different from random interactions. By doing so, we will show that there exist implicit communities in Amazon.

In information theory, entropy is a measure of unpredictability (randomness) [22]. We thus employ entropy as a measure to assess the randomness of interactions among users.

Let $X_u$ denote the event that an edge is formed from user $u$ to other users and $x_{uv}$ denote the event that an edge is formed from commenter $u$ to reviewer $v$. Then, $X_u$’s entropy $H(X_u)$ for outgoing edges, so called outgoing entropy, is given as follows:

$$H(X_u) = - \sum_{v \in U_u} P(x_{uv}) \log_2(P(x_{uv})),$$

(4.1)

where $H(X_u)$ is the entropy value, $P(x_{uv})$ is the probability that $x_{uv}$ occurs, and $U_u$ is the set of users who have interacted with user $u$.

Let $Y_v$ denote the event that an edge is formed from other users to $v$ and $y_{uv}$ denote the event that an edge is formed from commenter $u$ to reviewer $v$. Then, $Y_v$’s entropy $H(Y_v)$ for
incoming edges, so called *incoming entropy*, is given as follows:

\[ H(Y_v) = - \sum_{u \in U_v} P(y_{uv}) \log_2(P(y_{uv})), \quad (4.2) \]

where \( H(Y_v) \) is the entropy value, \( P(y_{uv}) \) is the probability that \( y_{uv} \) occurs, and \( U_v \) is the set of users who have interacted with user \( v \).

Fig.4.2 and Fig.4.3 show users’ incoming/outgoing entropy and in-degree/out-degree in real/random dataset of four categories - Books, Movie, Electronics, and Tools. Random dataset was built based on the random model in Section 4.3.1. In order to visualize a relationship between entropies and corresponding degrees clearly, values were sorted by entropy. The X-axis represents sorted incoming/outgoing entropies and the Y-axis represents corresponding in/out degrees.

Usually, if an event (i.e., edge) is randomly generated (i.e., interactions are randomly distributed), a user’s entropy and degree should be proportional by definition of the entropy (i.e., Eq.4.3.2 and Eq.4.3.2). Indeed, we can see that in the random model dataset, the in-degrees and the out-degrees were almost proportional (i.e., smooth line graph) to the corresponding incoming entropies and outgoing entropies, as shown in Fig.4.2 and Fig.4.3.

In the real dataset, on the other hand, the graph of each category presents a distinguishing characteristic. To be specific, in/out degree was out of proportion (i.e., a lot of spikes) to the corresponding entropy values in two subjective categories: Books and Movie. Those spikes
mean that the entropy is low, even when the degree is high. In other words, a large portion of interactions of those users are focused with a few users, which results in low entropies given high degree. Hence, it suggests that the interactions of such users with spikes are not random. Note that the general shape of the graph without spikes are similar to that of random model. This lend support to our assumption that user interactions occur randomly in general, but some users form implicit communities.

On the other hand, the figures of two objective categories, Electronics and Tools, are very close to those of the random model dataset as shown in Fig.4.2 and Fig.4.3. One possible explanation is that people usually share opinions and interact a lot throughout various items in subjective categories (e.g., books and movie), whereas people interact to grab information about specific sets of items in objective categories (e.g., electronics and tools). In other words, people’s interactions about objective items are rather limited to a small set of items of interest than various items; therefore, their interactions are similar to a random model.

From these results, we can see that relationships between users in different categories may show different characteristics. For example, many relationships in certain categories are random; while there may exist social relationships in certain categories, so that interactions between users occur because of the social relationships, instead of purely randomly. We will show these different characteristics actually have influence on the prediction accuracy of different recommendation algorithms in Section. 4.4.2.

In the following sections, we will describe how we define a user relationship and a community.
in recommender systems, and implicit communities found based on our definitions in Amazon.

4.3.3 User Relationship

To define a user relationship, let us first give an example scenario. Suppose that $u$’s interest is similar to $v$’s, $u$ thinks $v$’s reviews are always interesting, and $u$ posts comments to many of $v$’s reviews. Then, we may say that $u$ recognizes $v$ in a certain way, so that $u$ is likely to be influenced by $v$.

With the consideration of a random model discussed in Section 4.3.1 and a scenario above, let us define a user relationship. Intuitively, if there are a large number of interactions between two users compared with the number of expected interactions in the random model, we may assume that those two users are somewhat related. Based on the assumption, we denote $u$ is related to $v$, when the number of $u$’s comments on $v$’s different reviews lies outside the range of the expected number in the random model.

Suppose that there is a graph $G$ with the total number of all edges $n$, and $u$ and $v$ are vertices (users) on $G$. There are $\omega$ interactions from $u$ to $v$, which means the number of edges from $u$ to $v$ is $\omega$. Meanwhile, $u$’s out-degree is $O$ and $v$’s in-degree is $I$. So, if they are not related, the chance for an edge from $u$ to $v$ to form is $\rho_{uv} = (O/n) \times (I/n)$.

Let $X_{uv}$ denote an event for $\omega$ edges $\overrightarrow{e_{uv}}$ to form in a random model. As discussed in Section 4.3.1, $X_{uv}$ follows Bernoulli distribution and therefore the mean $X_{uv}$ is $\omega/n$ and the standard deviation is $\sqrt{X_{uv} \times (1 - X_{uv})}$. Given the mean and the standard deviation, confidence intervals are often computed using z scores [112]. To compute $\tau$% confidence interval, $\alpha = 1 - \frac{x}{100}$ and $z$ is defined. For example, $\alpha$ is 0.05 and $z$ is 1.96 for the 95% confidence interval.

For each $\tau$, the upper and lower bounds of $\tau$ are $X_{uv} \pm z_{\tau} \times \sqrt{X_{uv} \times (1 - X_{uv})}/n$, where $z_{\tau}$ is a z-score for $\tau$ confidence interval.

If the probability $\rho_{uv}$ is larger than the upper bound, it indicates that $u$ and $v$ have less interactions than the expected one, which is also common in a random case; because if $u$ and $v$ do not share any interest, they will not interact with each other. Accordingly, we only take into consideration the case that the probability is less than the lower bound.

A formal definition of user relationship is given as follows.

**Definition 1.** $u$ is related to $v$ with $\tau$ confidence interval, if the following condition holds:

$$\rho_{uv} < \theta_{uv},$$

where $\rho_{uv}$ is the probability for an edge $\overrightarrow{e_{uv}}$ to form in a graph $G$ and $\theta_{uv}$ is the lower bound of given confidence interval $\tau$ (i.e., $X_{uv} - z_{\tau} \times \sqrt{X_{uv} \times (1 - X_{uv})}/n$).

Given Definition 1, we quantify the strength of relationship in two ways. The larger difference between $\rho_{uv}$ and the bound, the less likely the interactions between $u$ and $v$ happen randomly,
and thus the more likely a strong social relationship exists between \( u \) and \( v \), which, in turn, the more influences \( v \) has on \( u \). The strength can thus be defined with the difference between \( \rho_{uv} \) and the bound as follows.

**Definition 2.** The strength \( \Delta_{uv}^1 \) of user relationship between \( u \) and \( v \) is

\[
\Delta_{uv}^1 = |\rho_{uv} - \theta_{uv}|
\]

where \( \rho_{uv} \) is the probability for an edge \( \overrightarrow{e}_{uv} \) to form in a graph \( G \) and \( \theta_{uv} \) is the bound, \( \bar{X}_{uv} - z_\tau \sqrt{\bar{X}_{uv} (1 - \bar{X}_{uv})} / n \)

On the other hand, the wider confidence interval \( \tau \) with which the relationship is defined, the more likely a strong social relationship exists between \( u \) and \( v \) by Definition 1. The strength can thus be defined with the confidence interval \( \tau \) as follows.

**Definition 3.** The strength \( \Delta_{uv}^2 \) of user relationship between \( u \) and \( v \) is

\[
\Delta_{uv}^2 = \tau_{uv}
\]

where \( \tau \) is confidence interval with which the relationship between \( u \) and \( v \) is defined.

The user relationships can be extended to communities in which users are related to another while constructing social graphs. The communities are defined with the confidence interval used in the definition of user relationships as follows.

**Definition 4.** edge \( \overrightarrow{e}_{uv} \) (in turn, user \( u \) and \( v \)) belongs to \( \tau\% \) community, if the following condition holds:

\[
\rho_{uv} < \theta_{uv}
\]

where \( \rho_{uv} \) is the probability for edge \( \overrightarrow{e}_{uv} \) to form, and \( \theta_{uv} \) is the bound of \( \tau\% \) confidence interval.

### 4.3.4 Revealing Implicit Communities in Recommender Systems

In this section, we present implicit communities revealed in our dataset from Amazon.

Fig.4.4 represents discovered \( \tau\% \) communities from two categories - Books and Movie. The thickness of an edge represents the number of interactions. Accordingly, the thicker edge \( \overrightarrow{e}_{uv} \) is, the more interactions from \( u \) to \( v \) exist. Relationships with thicker edges are thus stronger than relationships with thinner edges. The darkness of vertices represents the total degree of the vertices.

As shown in Fig.4.4, there are two patterns of sub-communities. One is a so called following community in which there are a few influential users (reviewers) who get a lot of comments from many commenters, but do not leave many comments to the commenters. We refer to the
relationship between influential users and commenters as a *following relationship*, because it is similar to the relationship between followers and followees on Twitter. Note that the influential user (followee) does not leave many comments to followers. Consequently, it is not necessary that an influential user’s opinion depends on the followers’ ones. Another pattern of sub-communities is a so called *strongly correlated community*. Compared with the one-directional interactions in the following communities, interactions in strongly correlated communities are bi-directional. In this case, users are mutually related to each other, so that the users have an influence on each other.

Recall that the strength of a user relationship is defined by $\tau$ (Definition 3). Hence, relationships in Fig.4.4(c) and Fig. 4.4(d) are stronger than those only in Fig.4.4(a) and Fig. 4.4(b).
As delineated in Section 4.3.3, a stronger relationship between users means that there are more interactions between the users than the expected. Note that multiple comments to the identical review are counted as a single interaction. Also, a commenter’s intention behind comments on a review is essentially to discuss the item of the interest. Many interactions between a commenter and a reviewer thus indicate that they share interests in many items. Hence, we argue that users who have stronger relationships would have more influences on each other than weaker relationships and those whose opinions will be more helpful to improve recommendation accuracy, which will be shown in Section 4.4.2.

Also, following communities tend to remain with larger \( \tau \% \) community as shown in Fig. 4.4. These results mean that there is a tendency for following communities to include stronger relationships than a strongly correlated community. As mentioned above, it is not necessary that followee’s opinion depends on the followers’ ones. We thus argue that considering directions of discovered social relationships will be more helpful than ignoring directions, which will be also shown in Section 4.4.2.

4.4 Recommendation

In this section, we delineate how to utilize discovered implicit communities for the purpose of recommendation and evaluate the significance of the communities. The main idea is to treat users in the community as social relationships, i.e., friends.

4.4.1 Recommendation with Discovered Communities

Our goal is to show that discovered implicit communities are relevant to item preference and they can be incorporated into social recommender systems to improve recommendation accuracy. To do so, we employed commonly used social recommender models \([38, 71, 108]\), details of which are given below.

In previous social recommender models, a predicted rating for an item is often computed as an aggregation of friends’ ratings to the item \([71]\). One way to aggregate is to use the average of friends’ ratings, referred as \textit{an average-based approach}, based on assumption that a user’s taste should be close to the average tastes of its friends. The formal definition of user \( u \)'s predicted rating for item \( i \) \((r_{u,i})\) by averaging friends’ ratings is as follows.

\[
r_{u,i} = \frac{1}{n} \sum_{v \in F} r_{v,i},
\]

where \( n \) is the number of \( u \)'s friend, \( v \) is a user in the set \( F \) of \( u \)'s friends, and \( r_{v,i} \) is \( v \)'s rating for the item \( i \).
However, user $u$ may have a lot of friends with different tastes, so that their ratings are not meaningful to predict $u$’s ratings. Another way is to treat each friend differently according to how similar $u$ and $v$ are, referred as a similarity-based approach. The formal definition of user $u$’s predicted rating for item $i$ ($r_{u,i}$) by employing friends’ similarity is as follows.

$$r_{u,i} = k \sum_{v \in F} \text{sim}(u, v) \cdot r_{v,i},$$

where sim($u,v$) is the similarity between $u$ and $v$ and $k$ is a normalizing factor defined as

$$k = \frac{1}{\sum_{v \in F} |\text{sim}(u, v)|}$$

To compute the similarity, there are two common methods we can borrow in the literature: Vector Cosine Similarity (VCS) and Pearson Correlation Coefficient (PCC) [71]. VCS is defined as

$$\text{sim}(u, v) = \frac{\sum_{i \in I_{uv}} r_{u,i} \cdot r_{v,i}}{\sqrt{\sum_{i \in I_{uv}} r_{u,i}^2} \sqrt{\sum_{i \in I_{uv}} r_{v,i}^2}},$$

and PCC is defined as

$$\text{sim}(u, v) = \frac{\sum_{i \in I_{uv}} (r_{u,i} - \overline{r_u}) \cdot (r_{v,i} - \overline{r_v})}{\sqrt{\sum_{i \in I_{uv}} (r_{u,i} - \overline{r_u})^2} \sqrt{\sum_{i \in I_{uv}} (r_{v,i} - \overline{r_v})^2}},$$

where $\overline{r_u}$ and $\overline{r_v}$ are the average ratings of user $u$ and $v$ for all the items rated by $u$ and $v$, respectively and $I_{uv}$ is the set of items $u$ and $v$ have in common.

There are a variety of ways to define who are one’s “friends” for the purpose of recommendation [38]. In other words, we need to determine whose ratings should be included when predicting a user’s rating. One option is to only consider immediate users (i.e., one hop away in a social graph) and the other is to consider both immediate and distant users (i.e., multiple hops away in a social graph). Relationships in our approach also have directions from commenter $u$ to reviewer $v$ as mentioned in Section 4.1 and Section 4.3. Thus, we have the options to either use directional information or to ignore it. When considering the direction, we include reviewer $v$’s ratings to predict commenter $u$’s rating but not vice versa. This is based on our assumption that while the reviewer $v$’s opinions are more likely to have influence on commenter $u$, it is less clear that commenter $u$’s opinion would have influence on reviewer $v$. Alternatively, we can ignore directional information and when predicting a reviewer $v$’s rating, we also include commenter $u$’s ratings as well.

When considering distant users, we may further apply community detection techniques to $\tau\%$ community. For example, we may use users’ ratings as long as there is a path in a social
graph from/to a user; we may consider \( k \)-hop distant users, so that we use users’ ratings, if they are \( k \) hops away from/to a user; or, we may apply clustering techniques to filter out users who are too far away in a social graph. The techniques used in this work will be discussed in Section 4.4.2.

### 4.4.2 Comparison with Collaborative Filtering Methods

In this section, we evaluate the significance of the discovered communities. The significance will be validated by analyzing recommendation accuracy and how many users can get benefit from the proposed approach. As discussed before, there are a variety of ways to determine whose ratings will be used to predict a user’s ratings. Let \( F_u \) denote the set of users whose ratings will be used to predict user \( u \)’s ratings. We introduce 3 approaches - directed, undirected, and connected - to determine \( F_u \). We compared those approaches and two representative existing popular collaborative filtering methods: the memory-based nearest neighbor (NN) [103], and the latent factor model-based (LFM) [60] algorithms. NN finds similar users, whereas LFM discovers factors to predict.

For the directed, we considered the direction of interaction. When there is an edge from commenter \( u \) to reviewer \( v \) in a social graph, we used \( v \)’s ratings to predict \( u \)’s ratings but not the other way. In other words, \( v \) belongs to \( F_u \) but \( u \) does not belong to \( F_v \). For the undirected, we ignored directions so that we used a reviewer’s rating to predict a commenter’s ratings as well as a commenter’s ratings to predict a reviewer’s ratings. In other words, \( v \) belongs to \( F_u \) and \( u \) belongs to \( F_v \).

While for both directed and undirected approaches, we considered only immediate users, for connected approaches, we considered both immediate and distant users as long as they are connected in our implicit social community. Furthermore, in this approach, we ignored directional information. In other words, all users who have a path in the social graph from/to \( u \) belong to \( F_u \).

When we combine the three categories on how to define \( F_u \) with two aggregation methods: average-based or similarity-based, we have our three average-based approaches: directed average (DA), undirected average (UA), connected average (CA) and three similarity-based approaches: directed similarity, undirected similarity, and connected similarity. Here we used vector cosine similarity to compute a similarity for our 3 similarity-based approaches.

First, we compared our three average-based approaches with three similarity-based approaches. Our results showed that the average-based approaches out-performed the similarity-based ones. We argue it is because similarity with other users cannot be measured without previous rating. That is, if user \( u \) does not have any item in common with other users, similarity-based approaches can neither use \( u \)’s rating to predict other’s nor give recommendations to \( u \).
On the other hand, the average-based approaches can take $u$’s rating to predict all related users, that is, the average-based approaches can use more information than the similarity-based ones, which resulted in the better accuracy and coverage. Also, commenters without previous ratings can not get recommendation by a similarity-based approach, because similarity cannot be measured; but the average-based approach can give recommendation to commenters without previous rating using related users’ ratings. Therefore, in the following, we only present the results of three average-based approaches. More specifically, we compared five approaches: our three average-based approaches, directed average (DA), undirected average (UA), and connected average (CA) and two state of the art collaborative methods: NN and LFM, in terms of both prediction accuracy and coverage.

**Prediction Accuracy**

A recommender system predicts each user’s rating for an item to judge whether the user likes/dislikes the item. We thus predicted a user’s rating and compared the predicted rating with the real rating of the user. The rating ranges from 1 to 5. The prediction accuracy is measured by mean absolute error (MAE), which is defined as follows.

$$MAE = \frac{\sum_{u,i} |r_{u,i} - r'_{u,i}|}{n},$$

where $r_{u,i}$ is user $u$’s rating about item $i$, $r'_{u,i}$ is an estimated value for $u$’s rating about item $i$, and $n$ is the number of items. The smaller MAE is, the more accurate a prediction is.

We compared the prediction accuracy of our 3 approaches using implicit social communities and two collaborative filtering methods across four categories: Books, Movie, Electronics, and Tools. Fig.4.5 represents MAE of those 5 approaches across various $\tau$% confidence communities. The X-axis represents each confidence community as defined in Section 4.3 and the Y-axis represents the average of MAE of predicted ratings for the community users. Recall that interactions between users in Electronics and Tools are more similar to the random model. Hence, there existed only a few users in a larger $\tau$% community so that users who are in larger $\tau$% communities (i.e., 99.5%, 98% in Electronics, 99.5%, 98%, 95% in Tools) could not get predictions with a limited amount of information.

As shown in Fig.4.5, the directed approach (DA) performed better (lower MAE) than the two undirected ones: undirected (UA) and connected (CA) approaches. This result suggests that a commenter’s opinion may not have much influence on a reviewer’s opinion; thus when predicting a reviewer’s ratings, it is more beneficial to exclude commenters’ ratings.

Since it might limit the number of users who could get predictions when considering only immediate users, we employed the connected approach. Although more users’ ratings can be
used to predict a user’s ratings, it increases the possibility to include more weakly related users. Consequently, the predicted ratings with CA were less accurate (larger MAE) than DA and UA. Note that MAE dropped dramatically between 60% and 70% communities with the connected approach. It occurred because many weak relationships started to be included when we used the $\tau$ smaller than or equal to 60%; Note that weak relationships mean that two users’ would have less influences on each other. Thus, a user’s rating may not be close to the average of his/her weakly related users’ ratings. Hence, MAE will be larger as we include more weakly related users.

Also, there was a big drop between 60% and 70% communities with the directed or undirected approaches in Electronics and Tools, while there was no such a big drop in Books and Movie. It is because interactions in Electronics and Tools were more similar to the random model and relationships were thus relatively weak; while relationships in Books and Movie were relatively strong. Consequently, the accuracy will be worse as many weak relations are considered. As shown in Fig.4.5, using 0 % community (i.e., users are considered, once they have at least one interaction) make prediction accuracy worse than NN or LFM.

In many papers, LFM was shown to be more accurate than NN [44, 59, 71]. Indeed, LFM
performed better for community users in Electronics and Tools where interactions follow the random model more closely as shown in Fig.4.5. However, for community users in Books and Movie where interactions do not follow the random model, NN provides almost similar or better prediction accuracy than LFM as shown in Fig.4.5. We argue this is because NN considers user similarity relationships and the taste of related users in Books and Movie are more similar to each other. In other words, considering user similarity relationships (NN) performed better than considering latent factors (LFM).

In short, we showed that when incorporating our implicit social communities, all three average-based approaches out-performed the two state of the art approaches: NN and LFM across all four categories on predicting users’ ratings.

**Coverage**

While we have shown that recommendation using discovered implicit communities can improve prediction accuracy on a recommender system, it is also important for us to show how many users can potentially benefit from those communities. Therefore, we measured the percentage of users in each category who can get predictions from 5 approaches.

Fig.4.6 shows the coverage of 5 approaches from Books, Movie, Electronics, and Tools. Recall that relationships in a larger $\tau$% community (i.e., 99% > 98% >..) are stronger than those only in a smaller $\tau$% community; and there were fewer users in a larger $\tau$% community as shown in Fig.4.4, which means many users in the system may not get predictions; whereas there are more users in a smaller $\tau$% community, which means more users in the system can get predictions.

As mentioned before, there were more strongly related users in Books and Movie, compared to Electronics and Tools. In other words, more users could get predictions in Books and Movie, compared to Electronics and Tools. As shown in Fig.4.6, the coverages of Books and Movie were higher than that of Electronics and Tools. Note that nearly 90 % of users in Books could get predictions by DA, UA and CA using 20 % ~ 60% communities, whereas only 40 % or less users could get predictions by NN and LFM. In Books, many users have strong relationships to each other (i.e., a lot of interactions throughout various items) so that many commenters without previous ratings can get predictions by average based approaches, whereas users without previous ratings can not get predictions by NN.

As discussed before, interactions in Electronics and Tools are similar to random interactions; which means users do not have many interactions throughout various Electronics and Tools items (i.e., lack of implicit social connections). Hence, many commenters without previous ratings could not benefit from communities much using average-based approaches (worse coverage).

Overall, Fig.4.6 shows that the coverage decreases as $\tau$ increases in general while as shown
in Fig.4.5, prediction accuracy improves as $\tau\%$ increases. Although we get the highest accuracy using stronger relations, the coverage may be too small, close to 0%. To address this issue, we combined our implicit social approaches with existing collaborative filtering methods, which will be described in the following section.

4.4.3 Combining A Social Recommender System with Collaborative Filtering Methods

Note that our goal is not to propose a new recommender system, but to suggest incorporating discovered implicit communities into existing recommender systems. As mentioned above, there may not exist a large community and a lot of users may not get predictions in certain categories such as Electronics and Tools. To balance accuracy and coverage moderately, we combined our social recommender systems using discovered communities with existing collaborative methods, called a hybrid approach. That is, if a user belongs to a community, a social recommendation algorithm is applied, otherwise a collaborative algorithm is applied. As shown in Fig.4.5 and Fig.4.6, there was no big difference between NN and LFM in terms of MAE and coverage. Therefore, for the space reason, we only present results with one of collaborative
filtering methods, NN, for our hybrid approach in this section.

Fig. 4.7 shows MAE and coverage when we use hybrid approaches. Overall, our results showed that our hybrid approaches beat the collaborative method NN in both prediction accuracy and coverage across all four categories. As mentioned above, there are fewer users in a larger $\tau\%$ community so that many users got predictions by NN. Consequently, MAE of the hybrid approach converged to MAE of NN, as $\tau$ increases as shown in Fig. 4.7. Note that communities in Tools were the smallest among 4 categories. Hence, MAE of the hybrid approach was similar to MAE of NN even for 70% or 80% communities; whereas MAE of the hybrid approach greatly improved over NN in Books, Movie, and Electronics.

As shown in Fig. 4.5 and Fig. 4.6, an undirected approach covers more users than a directed approach and there is no significant difference between MAE of those. In other words, more users will get predictions by social recommender algorithms; and fewer users will get predictions by NN, when an undirected approach is applied. Hence, MAE of the undirected approach with NN is lower than the directed approach with NN.

On the other hand, the coverage of the hybrid approach with DA, UA, and CA showed an increase over NN as shown in Fig. 4.7. This is because commenters without previous ratings can get predictions by average-based approaches, whereas NN can only cover users with previous ratings.

As discussed before, each user tended to have many interactions as well as to write many reviews in Books and Movie. In other words, many of covered users by social recommender algorithms and NN were overlapped. Hence, the coverage of the hybrid approach was the same as or similar to the larger one of two (i.e., either coverage of the social recommendation or NN) as shown in figures 4.6 and 4.7.

On the other hand, many reviewers in Electronics and Tools did not belong to discovered communities. That is, most users covered by the social recommender algorithms were commenters without previous ratings. Accordingly, users covered by social recommender algorithms and users covered by NN were almost mutually exclusive. Hence, covered users by the hybrid approach were union of two nearly disjoint users. In other words, the coverage of the hybrid approach was similar to the sum of two coverages (i.e., coverages of social recommender algorithms and NN) as shown in figures 4.6 and 4.7.

In short, Fig. 4.7 shows that we could adjust the trade-off between the data sparsity problem and prediction accuracy with different $\tau$. Indeed, it is shown that the hybrid approach using 20% ~ 60% communities greatly improve both accuracy and coverage over NN. In fact, the same results holds for LFM.
4.5 Summary

Recent research has pointed out the importance of social relationships between users in recommender systems based on an intuition that friends are likely to have similar interests to each other. However, social relations are not always available in recommender systems and it is not guaranteed that friends always share similar interests in different categories of items.

In this work we proposed a method to extract implicit relationships from a recommender system and to utilize discovered implicit communities for recommendation. Based on the assumption that if a few users have a large number of interactions with each other, when compared to the expected number of interactions in a random network, in a recommender system, they are likely to have similar interests, we exposed implicit communities in recommender systems. In order to corroborate that our approach works in real recommender systems, we crawled Amazon and performed analysis on the dataset. With experimental results, we showed that there exists implicit communities in recommender systems in practice, and that we can improve the accuracy of recommender systems by utilizing discovered communities.
Figure 4.7: MAE and Coverage of a hybrid approach in each category

(a) Books MAE

(b) Books coverage

(c) Movie MAE

(d) Movie coverage

(e) Electronics MAE

(f) Electronics coverage

(g) Tools MAE

(h) Tools coverage

34
Chapter 5

COMPARS: Toward An Empirical Approach for Comparing the Resilience of Reputation Systems

5.1 Motivation

Large-scale decentralized systems, such as peer-to-peer systems, online auction communities and ad hoc mobile networks, often involve transactions between strangers from different security domains with no pre-existing knowledge of each other. Although such systems offer great benefits in terms of service diversity, flexibility and scalability, they also give malicious parties the opportunity to cheat during transactions without being identified or punished [124, 131]. Inspired by social interactions between human beings, reputation mechanisms have emerged as a major technique for trust establishment in decentralized systems. In a reputation system, upon completion of a transaction, the involved parties issue feedback to evaluate one another’s service or behavior during the transaction. Before a new transaction starts, one may first assess a party’s trustworthiness based on the feedback for its previous transactions. This process can be viewed as the application of a trust function, which conceptually takes as input the feedback for a party’s past transactions (or that of other parties, when necessary), and outputs a trust value to indicate its trustworthiness.

Many trust functions have been proposed in the literature (e.g., [6, 16, 52, 54, 126, 128, 134]). However, when building a decentralized system, it is non-trivial to pick the right trust function.

This chapter has been published in Proceedings of the fourth ACM conference on Data and application security and privacy, pp. 87-98. ACM, 2014 [18]
One has to consider and balance a variety of factors, including computation and communication costs, scalability and resilience to manipulation. While many other factors are relatively straightforward to evaluate, the evaluation of resilience can be challenging. Resilience refers to how accurately a computed trust value reflects a party’s true trustworthiness in the presence of malicious manipulation. Malicious manipulation includes selectively and strategically providing good or bad transactions, or issuing dishonest positive or negative feedback. We emphasize that the evaluation of resilience has to consider adaptive attackers who are aware of how a trust function works and adapt their behavior accordingly to maximize their gain in a system.

Unfortunately, though a large amount of work has been done on the design of trust functions, very few studies are devoted to systematic evaluation of their resilience [51, 55, 85]. Most existing efforts focus on performance evaluation [42, 124], while efforts on resilience evaluation are rather limited and are often built on the assumption that attackers are likely to behave within a fixed set of strategies that can be described by static models [16, 42, 55, 85, 116, 124]. This assumption is unrealistic, as it does not consider the adaptive nature of adversaries. Furthermore, evaluations based on fixed strategies do not offer a fair comparison of different trust functions. A particular trust function may be resilient to one type of attack yet vulnerable to another. Hence, comparing two trust functions using a fixed attack strategy may only offer a biased and incomplete view of their strength against manipulation. To be fair, one has to compare the worst cases of two trust functions. That is, it is necessary to compare their resilience against the most effective attacks, which are likely to be different for different trust functions.

In this work we propose an evaluation platform for the COMPARison of Reputation Systems (COMPARS) that specifically takes into consideration the adaptiveness of adversaries. Essentially, an adversary tries to game a reputation system to achieve maximum profits, while normal users behave more or less consistently. Therefore, from an attacker’s point of view, a reputation system sets up a single-player game. The trust function along with the behavior of normal users forms the environment and rules of the game (i.e., how the system will evolve when certain actions are taken), and the goal of the attacker is to carefully choose its actions to maximize certain profit measures (e.g., to meet a profit goal in the shortest time, or obtain the maximum profit in a given period of time).

Theoretically, given a trust function and a model of normal user behavior, there exists an optimal strategy for an attacker to game the system (i.e., the most effective attack). However, in practice, it is often hard to derive such optimal strategies. A system’s state is determined by the behavior of a large number of users. Further, although normal users follow some behavior models more or less consistently, they are non-deterministic in nature (e.g., even if an honest user strives to provide good service, unsatisfactory transactions may happen from time to time due to uncontrollable factors such as network delays or interruption of delivery services, which have
to be captured through probabilistic models). Therefore, it is very difficult, if not impossible, to purely rely on theoretical analysis to reason about the ultimate future state of a system and derive the optimal strategy directly.

In COMPARS, we adopt an empirical approach that approximates the optimal strategy of an attacker. Specifically, COMPARS explores the future system states after an attacker takes up to \( k \) actions, similar to the idea of MiniMax [86]. Among these future states, COMPARS picks the one that is most beneficial to the attacker, and uses it to determine the next action that the attacker should take. When doing so, we use a probabilistic reasoning method to deal with the non-deterministic nature of other users, which we will detail in Section 5.3 and Section 5.4.

### 5.2 Related Work

A number of reputation systems[6, 16, 24, 52, 54, 106, 126, 128, 134] have been proposed with the goals of ensuring trustworthy transactions between participants and protecting peers from malicious users. Most research efforts have been conducted primarily on the development of trust functions.

EigenTrust[54] supposes that there exists pretrusted peers in peer to peer file sharing applications, and utilizes normalized local trust values to compute a global trust value.

PeerTrust[126] employs similar ideas, but PeerTrust is built in a decentralized system. PeerTrust introduces five factors - i) feedbacks from other peers, ii) the total number of transactions a peer performs, iii) the credibility of the feedback sources, iv) transaction context factor, and v) the community context factor - and quantifies the trustworthiness of peers with the weighted average of five factors; since we cannot guarantee the existence or availability of pretrusted peers in decentralized networks.

Some systems[52] utilize additional information such as a relation between nodes (e.g., transitive chains/paths). TNA-SL[52] utilizes subjective logic as a measure of trust computation so that TNA-SL simplifies transitive trust relationships into a directed graph. Given the directed graph, TNA-SL calculates a single opinion, i.e., a consumer’s general impression on all transactions about a service provider. An opinion is derived by applying two logical operators, discount and consensus. Discount is an operator to evaluate transitive chains and consensus is an operator to compute the average of two different opinions about one peer.

P2PRep system[6] is proposed to deter selfish and malicious activities in completely decentralized systems. P2PRep collects and merges opinions into distinct trust values using fuzzy techniques.

PowerTrust[134] employs Bayesian based approaches to calculate local trust values and picks small number of power nodes who are the most reputable to aggregate global values.
TrustGuard\cite{113} presents a reputation framework to cope with three vulnerabilities of reputation systems - strategic oscillation, fake transactions, and dishonest feedbacks.

Beta Reputation System\cite{53} utilizes the beta probability density function, which models binary events. Based on each peer’s own experience as well as other peer’s one, the distribution helps to estimate the probability of a service provider’s future behaviors.

These reputation systems have shown their feasibility in different application domains and their effectiveness has been analyzed under a few scenarios in which malicious users attempt to exploit the systems. However, the proposed systems are designed for specific application domains and evaluated only on those targeted domains, making general comparison difficult.

In order to facilitate a systematic analysis, a few papers discuss design issues of reputation systems\cite{42, 73, 116} and classify trust functions into a few categories\cite{17, 64, 101, 117, 132}. Some of them\cite{42} further discuss attacks and defenses related to design issues. Papers that deal with design issues\cite{42, 73, 101, 116, 117, 132} can offer a clear view on different dimensions of reputation systems with a common criteria. Also, existing reputation systems can be theoretically analyzed with the proposed criteria\cite{15}. A theoretic analysis, however, has the limitation that it does not take a practical perspective into account. Indeed, Josang \textit{et al.}\cite{51} address that a reputation system, which is considered theoretically robust, can be nevertheless vulnerable in realistic environments. Unlike theoretical approaches, we construct an evaluation framework that can generate quantitative values so that the different systems can be compared by a fair, empirical approach.

There have been a few studies\cite{15, 17, 29, 56, 124} that have attempted to address and evaluate the robustness of trust functions. \cite{42, 51} discuss a set of properties to evaluate reputation systems. Although they mention robustness, they do not provide an analysis of robustness, instead focusing on algorithm analysis including how to compute trustworthiness. Zuping \textit{et al.}\cite{17} classify trust-based recommender systems into three categories and pick a representative system in each category to analyze their robustness. Through experiments, the authors prove vulnerabilities in trust-based recommender systems. However, their evaluations are limited to a few specific recommender models and specific attacks.

Fullam \textit{et al.}\cite{29} propose a testbed to evaluate reputation systems, called \textit{ART}. Unfortunately, ART can only be used in a few specific applications and it only allows a small number of participants. Consequently, ART can deal with only simple user/attack behavior scenarios, which is unrealistic. For example, every participant is assumed to behave in the same way, attackers can not change their behavior, and no participant can enter/leave during evaluation. Instead, we employ a probabilistic reasoning method to handle a large number of users’ behavior.

Kerret \textit{et al.}\cite{56} propose a more general platform, TREET, to analyze reputation systems.
Although TREET is more flexible than ART, supporting both centralized and decentralized systems, it can only handle marketplace scenarios and specific attacks in the marketplace. Our COMPARS framework, on the other hand, is general and domain-independent.

West et al. [124] and Irissappane et al. [46] propose evaluation frameworks based on an empirical approach. [124] defines possible user/attacker behavior models and implements a simulator with a static trace. [46] simulates reputation systems with static user behavior models and fixed attack models. Static behavior models and traces, however, make systems vulnerable by nature, since attackers may be well-aware of how a trust function works and intentionally change their behavior to exploit a reputation system. Further, both [124] and [46] do not take users’ non-deterministic nature into consideration. Hence, it is hard to handle real scenarios where each user may have their own behavioral pattern in [124] and [46]. On the contrary, we employ a probabilistic user behavior model to capture the non-deterministic nature of other users.

Considerable research on evaluation frameworks [15, 42, 46, 77, 116, 124] lies in the classification of reputation systems and attacks/defenses in reputation systems. These works try to classify reputation systems considering their robustness against classified static attack models [15, 17, 26, 46, 51, 55, 106, 115, 124]. In contrast, we believe an evaluation based on static attack strategies is not sufficient to reflect the true robustness of a reputation system. Instead, we propose an evaluation framework that captures the adaptive nature of attackers who can exploit the properties of specific trust functions and behave accordingly to maximize their profits.

5.3 Optimal Attack Strategies

The attackers’ goal is to manipulate a reputation system without being detected. To do so, attackers often change their behavior according to the change of system states. Consequently, it is important to model attackers’ adaptive behavior to evaluate the robustness of trust functions. Attackers will essentially try to find and perform the most effective attack for a specific trust function. Clearly, the most effective attack will be different for different trust functions and attackers should have an optimal strategy to perform the most effective attack for a given trust function. We model a reputation system as a game, in which players are either normal users or attackers. In Section 5.3.1, we first explain our model of a game in reputation system. Then, we describe how to find the optimal strategy in Section 5.3.2 and Section 5.3.3.

5.3.1 Games in Reputation Systems

There are multiple players in a game whose type can be normal users and attackers in a reputation system. A new player may join during the game or a player may leave during the game. Each player has multiple choices of actions depending on its strategy in each turn. To
choose an action, each player considers the consequence of the action. In other words, the player needs not only to examine a current system state, but also to estimate a future system state. For example, a player will choose a certain action, if it estimates the action will bring a profit; a player will not choose a certain action, if it estimates the action will punish itself. A system state transitions to the next system state depending on each player’s action.

For a reputation system to reflect a user’s true trustworthiness, users are expected to behave in a way that the system wants them to. That is, a reputation system requires users to give a high(low) rating to a good(bad) service[131]. Consequently, normal users behave in a predictable manner; however, each user may behave in its own way, not the same way as other users. For instance, one user may tend to give relatively low ratings habitually, even when it is very satisfied; even if an honest user strives to provide good service, unsatisfactory transactions may occur due to faulty third-party delivery. Consequently, normal users can be assumed to behave in the predictable manner, but their behavior is still non-deterministic, and each normal user has its own behavior model.

An attacker, on the other hand, tries to game a reputation system to achieve its goal. Therefore, from an attacker’s point of view, a reputation system sets up a single-player game. The trust function along with the behavior models of normal users forms the environment and rules of the game. The goal of an attacker is to carefully choose its actions to maximize its profit.

Theoretically, given a trust function and models of normal user behavior, there exists an optimal strategy for an attacker to game the system. Unlike classical games such as chess, however, there are a large number of users whose behavior is non-deterministic in a reputation system. Also, the design of reputation systems is more complicated. Therefore, purely using a theoretical analysis to pick the optimal attack strategy can be prohibitively expensive. We thus employ an empirical approach to explore future system states after an attacker takes a sequence of actions and approximate the optimal strategy of the attacker. Similar to the idea of MiniMax[86], we represent possible system states as a tree, called an attack tree, but only in an attacker’s point of view. In the following subsections, we delineate how we generate an attack tree.

5.3.2 Attack Tree

A conceptual view of an attack tree is shown in Fig.5.1 and Fig.5.3. The formal definition of an attack tree is given as follows.

**Definition 5.** An attack tree is a rooted tree given by $\mathcal{S}$, $\mathcal{X}$, $\mathcal{A}$, where

- $\mathcal{S}$ is the set of nodes, each of which $(S_i = (C_i, P_i, R_i))$ represents a system state at a certain time point.
Figure 5.1: An attack tree

- $X$ is the set of edges, each of which $(X_i = \{(a_1, t_1), \ldots, (a_m, t_m)\})$ represents a sequence of normal users’ actions.

- $A$ is the set of edges, each of which $(A_i = (a_i, t_i))$ represents one of an attacker’s possible actions.

As discussed in Section 2.2, a system state transitions into the next system state, whenever an action occurs. The action may be performed by attackers or normal users. The attack tree, however, is employed to derive the attackers’ optimal strategy, while estimating future system states from an attacker’s point of view. Accordingly, we do not represent every possible system state that evolves depending on a normal user’s action as a node. Instead, we represent system states as nodes only when an attacker is involved in the transactions. In order to differentiate a system state right after an attacker’s single action $A_i$ from a system state after a sequence of normal users’ actions $X_i$, we represent the former as a square node and the latter as a round node. Each round node (except the root node) has $n$ square child-nodes, each of which corresponds to a system state after an attacker’s action $A_i$ occurs. Each square node and the root node have one round child-node corresponding to a system state after a sequence of normal
users’ actions $X_i$ occurs.

We assume that an attacker is involved in a transaction at time $t_j' (j = 0, \ldots, n)$. And $S_j' (j = 0, \ldots, n)$ are the system states at time $t_j' (j = 0, \ldots, n)$. As shown in Fig. 5.1, whenever an attacker tries to estimate a system state, a sequence of normal users’ actions is generated first. A sequence of normal users’ actions is generated by randomly choosing users and their actions according to the users’ behavior models. Each action results in a transition into the next system state. A node $S_0$ transitions to a node $S_0'$ after a sequence of normal users’ actions $X_0$ happens, as shown in Fig. 5.1. At time $t_0'$ when an attacker is involved in a transaction, the attacker has a set of choices $A_i$’s, i.e. the attacker’s possible actions. After taking one of the actions, an attacker will estimate its next system state. Depending on which action is taken by the attacker, a node $S_0'$ transitions to its $n$ child-nodes, $S_{0i} (i = 1, \ldots, n)$.

An attacker repeats this reasoning process, until the number of the attacker’s transactions exceeds a certain number, which is defined by the attacker. We call the number of the attacker’s transactions in an attack tree as the height of the attack tree, which is the same as the depth of a tree only considering round nodes. For example, the height of the attack tree in Fig. 5.1 is two. Ideally, if we can explore all possible system states, we can get the most accurate estimation for an attacker’s optimal strategy. In practice, it would be computationally infeasible to explore whole spaces. We thus limit the maximum height of an attack tree so that attackers can predict future system states with a reasonable computation. Fig. 5.2 represents the pseudocode for the generation of an attack tree.

A system state is determined by the behavior of a large number of users, which is likely to be probabilistic, as discussed earlier. Since the generation of an attack tree is an estimation process of the attackers, only one sequence of normal users’ actions cannot be a representative to reflect a large number of users’ probabilistic behavior. We thus sample the users’ actions to make better estimation of system states. That is, we generate sample sequences of normal users’ actions, instead of one single sequence of normal users’ actions. In the following subsection, we describe how we generate an attack tree with sample sequences of normal users’ actions.

### 5.3.3 Attack Tree with Sample Sequences

Fig. 5.3 depicts an example of an attack tree with sample sequences of normal users’ actions. Similar to Fig. 5.1, we assume that an attacker is involved in a transaction at time $t_j' (j = 0, \ldots, n)$ and $S_j' (j = 0, \ldots, n)$ are system states at time $t_j' (j = 0, \ldots, n)$. Whenever an attacker tries to estimate a system state, $m$ sample sequences of normal users’ actions $X_{jk} (k = 0, \ldots, m)$ are generated first. Each sequence $X_{jk}$ is generated by randomly choosing users and their actions according to the users’ behavior models. Although each action results in a transition into the next system state, we only represent system states after the last action in the sequence $X_{jk}$. 
AttackBehavior()
1: $S \leftarrow S_0$
2: $r \leftarrow 0$
3: Gen_AttackTree($S, r$)

Normal_Sequence($S_p$)
1: for $k=0$ to $L_S$ do
2: Randomly pick a user and its action, $\omega$.
3: $S_p \leftarrow Get_SystemState(S_p, \omega)$
4: end for
5: return $S_p$;

Add_Node(Parent $S_p$, Child $S_c$)
1: Add $S_c$ as one of $S_p$’s children into an attack tree.

Gen_AttackTree(System state $S_p$, $r$)
1: $r \leftarrow r + 1$
2: $S_c \leftarrow Normal_Sequence(S_p)$
3: Add_Node($S_p, S_c$)
4: for $k = 0$ to $N_A$ do
5: // Do an action $a_k$
6: $S_k \leftarrow Get_SystemState(S_c, a_k)$
7: Add_Node($S_c, S_k$)
8: if $r < M_T$ then
9: Gen_AttackTree($S_k, r$)
10: end if
11: end for

Get_SystemState(System state $S_p$, Action $a$)
1: $S_p \xrightarrow{a} S_c$
2: $S_p$ transitions to $S_c$ after $a$ happens.
3: return $S_c$;

$S_0$: an initial system state
$S_i$: System states at time $i$
$M_T$: Maximum height of an attack tree
$r$: A counter for the height of an attack tree
$A$: The set of attackers’ possible actions to achieve their goals.
$a_k$: An attacker’s one possible action at a certain time point. ($\in A$)
$N_A$: The number of attackers’ possible actions at a certain time point.
$\Omega$: The set of normal users’ possible actions.
$\omega$: One of normal users’ actions. ($\in \Omega$)
$L_S$: Chosen length of one sequence of normal users’ actions.

Figure 5.2: The pseudocode for generation of an attack tree
occurs as nodes in an attack tree. Hence, a node $S_0$ transitions to nodes $S_{0k}' (i = 0, \ldots, m)$ after sequences of normal users’ actions $X_{0k} (k = 0, \ldots, m)$ happen, as shown in Fig. 5.3.

To estimate a system state at $t_{j'} (j = 0, \ldots, n)$, we need to analyze the distribution of system states $S_{jk}' (k = 0, \ldots, m)$. Then, the attacker picks a representative system state, $S_j'$, among $S_{jk}' (k = 0, \ldots, m)$’s for the time, $t_{j'} (j = 0, \ldots, n)$. The representative system state can be defined by the attacker. For example, let $v_{jk}' (k = 0, \ldots, m)$ be the trust value of an attacker at each system state, $S_{jk}' (i = 0, \ldots, m)$. Then, the attacker may choose a system state $S_{j, \text{avg}}'$, at which an attacker’s trust value $v_{j, \text{avg}}'$ is the average of trust values $v_{jk}' (i = 0, \ldots, m)$, as a representative system state for the time, $t_{j'} (j = 0, \ldots, n)$.

Similar to Fig. 5.1, the attacker has a set of choices $A_j$’s, i.e. attackers’ possible actions, at time $t_{j'}$ when an attacker is involved in a transaction. After taking one of the actions, an attacker will estimate its next system state. Depending on which action is taken by the attacker, a node $S_j'$ transitions to its $n$ child-nodes, $S_{ji} (j = 1, \ldots, n)$. An attacker repeats this reasoning process, until the number of the attacker’s transactions exceeds a certain number, which is defined by the attacker.

Figure 5.3: An attack tree with sample sequences
5.4 COMPARS: A Framework for Comparison of Reputation Systems

The goal of the proposed approach is to evaluate trust functions in the presence of adaptive attack behavior. In subsection 5.4.1, we first give a brief overview of the proposed approach, called COMPARS (COMPArison of Reputation Systems). In subsections 5.4.2, 5.4.3, 5.4.4, and 5.4.5, we discuss four functional components of COMPARS and the evaluation criteria of COMPARS.

5.4.1 Overview of COMPARS

COMPARS simulates the evolution of a reputation system. The framework is built with basic components common to all reputation systems so that any reputation system can be easily integrated into COMPARS and so that different trust functions can be evaluated with COMPARS. Fig.5.4 describes the architecture of COMPARS, which consists of four functional components: initial state generator, transaction manager, reasoning manager, and evaluator.

Note that attackers often change their behavior, depending on specific properties of trust functions and of normal users. In order to reflect an attacker’s strategic behavior, COMPARS
considers a reputation system from the attacker’s point of view. Essentially, given the initial system state, the goal of an attacker is to carefully choose its behavior to maximize its profits. COMPARS thus derives an attacker’s optimal strategy to achieve its goal. First, the initial state generator generates an initial system state, which is defined by basic user information (e.g., the list of consumers, providers, and items each user has). Given the initial system state, the transaction manager controls who will be involved in each transaction. In general, normal users will not change their behavior much. Along with this observation, the transaction manager takes user behavior models as input. The transaction manager consists of the event generator and the transaction generator. The event generator controls who will be a consumer, \( c \), based on a current system state and user behavior models. Depending on the users behavior models, the transaction generator manages who will be chosen as a provider, \( p \) by a consumer, \( c \). In subsection 5.4.3, we will discuss how we define behavior models for normal users in detail.

Given a trust function and a system state, attackers attempt to game a reputation system to achieve their goals. Different from normal users, attackers are adaptive so that they are able to choose the optimal strategy under a specific system state. For attackers to derive their optimal strategy, COMPARS explores the future system states after an attacker takes up to \( k \) actions. COMPARS represents possible system states with different attacking actions as an attack tree, as illustrated in Section 5.3.

The reasoning manager handles the reasoning process of attackers, generating attack trees to reason about the attackers’ future actions. By monitoring a generated tree, COMPARS picks the most beneficial action to the attacker and uses it to determine the next action that the attacker should take. Given the optimal strategy, the evaluator carries out the optimal strategy and evaluates the robustness of the reputation systems.

In the following subsections, we will describe each component in detail.

### 5.4.2 The Initial State Generator

An initial system state is the system state before an attacker’s action occurs. Note that COMPARS simulates the evolution of a reputation system in an attacker’s point of view. Therefore, the initial state does not mean no transactions ever happen in the system. An attacker may join in the middle of a system; or, normal users can be compromised by attackers and start to behave as attackers. In order to generate the initial system state, the initial state generator takes initial parameters as input, including basic user information (e.g., the list of consumers, providers, and services offered by providers). With the given parameters, the initial state generator generates the initial system state \( S_0 = (C, P, R) \), where \( C \) is a set of service consumers, \( P \) is a set of service providers, and \( R \) is a set of feedbacks.
c: A consumer who wants to get a service
$S_t, S_{t_1}, S_{t_2}$: System states at given time $t$, $t_1$, and $t_2$, respectively
$\Sigma_c$: The set of consumers’ strategies to pick service providers
$\sigma_c$: c’s strategy to pick service providers ($\in \Sigma_c$)
$\gamma$: A feedback from a single transaction

**ConsumerBehavior()**
1: Pick a service provider p who meets the requirements of $\sigma_c$.
2: Do a transaction with p.
3: $S_t \xrightarrow{c \text{ gets a service}} S_{t_1}$
4: Issue a feedback, $\gamma$.
5: $S_{t_1} \xrightarrow{c \text{ issues a feedback}} S_{t_2}$

Figure 5.5: A consumer’s behavior and the evolution of system states

### 5.4.3 The Transaction Manager

Given the initial system state, the *transaction manager* controls who will be involved in each transaction. As mentioned earlier, normal users in reputation systems are likely to behave consistently. The *transaction manager* takes user behavior models as input. Our abstract model of normal user behavior is as follows. Note that the abstract model is not fixed, but is flexible and able to accommodate any user behavior model.

**A. A Consumer Behavior Model:** A consumer seeks services from a decentralized system. While doing so, a consumer needs to choose a set of services it would like to get as well as a provider from whom it would like to get the services. Given a trust function, a consumer’s strategy for choosing a provider may vary depending on a current system state. For example, one consumer may choose a provider whose trust value is over certain threshold, whereas another may only choose the top-ranked providers in the system. Based on its own strategy at a given system state, consumer c chooses provider p and starts a transaction with p. After c gets service i, c issues feedback $\gamma$. Fig.5.5 illustrates how a system state evolves depending on a consumer’s action.

**B. A Provider Behavior Model:** A provider offers a set of services, which is usually publicly known. Therefore, provider p’s set of services should be a part of its profile, as discussed previously. However, even if a provider offers the same services at different time, the quality of transactions may vary due to uncontrollable factors such as network delays or interruption of delivery services. Also, a provider may want to change the quality of transactions for the same services depending on the trust function and consumers’ strategies. For example, uploaders in file-sharing applications may publish normal quality files rather than high quality files to save
p: A provider who wants to provide a service
c: A consumer who wants to get a service
$S_t, S_{t_1}, S_{t_2}$: System states at given time $t$, $t_1$, and $t_2$
$\Sigma_p$: The set of providers’ strategies
$\sigma_p$: $p$’s strategy to provide a service ($\in \Sigma_p$)

**Figure 5.6:** A provider’s behavior and the evolution of system states

```
ProviderBehavior()
1: while $p$ is not chosen by any consumer do
2: $p$ checks a current system state.
3: $p$ picks a strategy $\sigma_p$.
4: $p$ lists a set of services, which meet requirements of $\sigma_p$.
5: $S_t \xrightarrow{\text{p lists a set of services}} S_{t_1}$
6: end while
7: if $p$ is chosen by a consumer $c$ then
8: $S_{t_1} \xrightarrow{\text{p offers a service to } c} S_{t_2}$
9: end if
```

their resources. As another example, sellers in e-commerce markets who do not have a good
transaction history may try to post only best items to repair their reputations. Note that this
is different from the attackers’ behavior, in which an attacker intentionally provides good or
bad services to maximize its profits. To capture such factors, we use a provider behavior model
in addition to a provider’s profile. Fig. 5.6 illustrates a provider’s behavior and how a system
state evolves depending on a provider’s action.

Provider $p$ lists a set of services and waits until it is chosen by a consumer. If $p$ is not chosen
within a certain time, $p$ checks the current system state and changes its strategy to offer the
services. For instance, $p$ may offer the same quality of services at a lower price or $p$ may offer
a better quality of services at the same price.

Given the initial system state and user behavior models, the transaction manager controls
who will be involved in each transaction. The transaction manager consists of the event genera-
tor and the transaction generator. Based on the current system state and user behavior model,
the event generator picks a consumer who will be involved in each transaction. For example,
the event generator may pick a user as a consumer only when its trust value is over a certain
threshold. Once the event generator picks consumer $c$, the transaction generator selects provider
$p$ depending on $c$’s strategy to pick a service provider. The quality of each transaction is de-
cided depending on a specific system configuration. In some reputation systems, one provider
may explicitly mention (advertise) a service quality; or, a reputation system may have its own
measure to judge a service quality [124].

5.4.4 The Reasoning Manager

As described in Section 5.3, sophisticated attackers often change their behavioral patterns intentionally based on the trust functions. If a framework evaluates trust functions based on a set of pre-defined strategies, this always leaves opportunities for attackers to exploit trust functions by using different strategies that are not included in the set of strategies. COMPARS thus employs an empirical approach to model attackers’ adaptive behavior, which is delineated in Section 5.3.

A reasoning process begins when an attacker is chosen for a transaction, generated by the transaction manager. An attacker can be a consumer or a provider. The reasoning manager takes an attacker’s goal as input. The attacking goal can be defined with a few parameters, which may include, but are not limited to a trust value, a profit, and a time period. For example, an attacker may want to achieve its profit goal within a certain time period, while maintaining a trust value above a certain value; or, an attacker may want to demote a competing provider’s trust value so as to prevent the provider from being chosen by consumers. To compute profits from a given action, the reasoning manager takes a profit model as input. A profit model is a method to compute profits by a single action at a specific system state.

Depending on a specific reputation system, the attacker may have a different set of choices of actions at each system state to achieve its goal. For example, the attacker may create a new account, so that it can control multiple accounts in one reputation system; whereas, multiple accounts are not allowed in another, so that the attacker may need to choose another action to achieve the same goal. Accordingly, the reasoning manager takes into account the attacker’s capabilities in a given reputation system and handles the attacker’s reasoning process in the system, so that COMPARS predicts the optimal action in the given reputation system.

The reasoning process continues until the maximum reasoning step (i.e., the maximum height of an attack tree) is reached. More reasoning will lead to better estimation. The amount of reasoning performed, however, will impact the computational overhead. Therefore, we limit the maximum number of reasoning steps so that COMPARS can predict future system states with a reasonable amount of computation.

5.4.5 The Evaluator

A good trust function will restrict the chances for an attacker to exploit a system. In other words, if an attacker can achieve its goal early, it is expected that the trust function is vulnerable to manipulation. Hence, the evaluator observes how many transactions are required for an attacker
to achieve its goal with its optimal strategy so that COMPARS estimates the robustness of trust functions.

5.5 Experimental Results and Analysis

This section describes an analysis of reputation systems using COMPARS. Many reputation systems have been developed in different application domains [132]. To show the validity of COMPARS, three influential reputation systems—EigenTrust [54], PeerTrust [126], and TNA-SL [52]—have been integrated into the COMPARS framework. Note that COMPARS is general and domain-independent, so that there are many ways to materialize COMPARS depending on which reputation system will be evaluated. For example, different user behavior models and profit models can be used to evaluate different reputation systems. Here, we provide a few case studies with the three reputation systems in an eBay-like e-commerce system; and show how COMPARS can be used to observe the resilience of different reputation systems with adaptive attackers.

We implemented EigenTrust and PeerTrust ourselves, and modified and adjusted the TNA-SL code by Andrew G. West et al [124]. As discussed in Section 5.2, EigenTrust is designed for peer to peer file sharing applications with the assumption that there are peers who always behave in an honest way and can thus be pre-trusted. PeerTrust is implemented in a decentralized P2P environment without any pre-trusted users. TNA-SL utilizes a theoretical approach with a greater emphasis on prior direct interaction. Details of the three reputation systems can be found in their respective papers [52, 54, 126]. Considering the fundamental differences between these three systems, we believe other reputation systems can be integrated easily into COMPARS and evaluated as well. We first present the parameters used for our experiments in Section 5.5.1 and we present experimental results in Section 5.5.

5.5.1 Materialization

Table 5.1 summarizes the notation for the parameters used in our experiments. Except for experiments where we needed to change some parameter values, we used the default values listed in the table.

*Normal user behavior model:* As mentioned earlier, the behavior of normal users in e-commerce markets (e.g., eBay) is typically predictable, but non-deterministic. Such nature can be captured by probabilistic models [124, 131]. We thus employed two parameters—*feedback reliability rate* $H_i$ and *service quality rate* $Q_i$, each of which was defined for consumers and providers, respectively. Feedback reliability rate $H_i$ is the probability that consumer $i$ gives feedbacks consistent with true service quality; service quality rate $Q_i$ is the probability that
<table>
<thead>
<tr>
<th>Parameter Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_U$</td>
<td>the number of users in our network</td>
</tr>
<tr>
<td>$N_S$</td>
<td>the number of samples for reasoning</td>
</tr>
<tr>
<td>$H_i$</td>
<td>feedback reliability rate of a consumer i</td>
</tr>
<tr>
<td>$Q_i$</td>
<td>service quality rate of a provider i</td>
</tr>
<tr>
<td>$\sigma_c$</td>
<td>consumers’ strategy to choose a provider</td>
</tr>
<tr>
<td>$G$</td>
<td>an attacking goal (the amount of profit that an attacker wants to get)</td>
</tr>
<tr>
<td>$M_T$</td>
<td>the maximum height of an attack tree for reasoning</td>
</tr>
<tr>
<td>$T_{avg}$</td>
<td>the average of trust values</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>a range value to calculate profits</td>
</tr>
<tr>
<td>$\delta_T$</td>
<td>threshold to be selected as a provider with “OVER TRUST” strategy</td>
</tr>
<tr>
<td>$\delta_R$</td>
<td>threshold to be selected as a provider with “OVER RANK” strategy</td>
</tr>
</tbody>
</table>

Table 5.1: List of parameters for experiments

provider $i$ offers good services.

The value for $H_i$ ranges from 0.0 to 1.0, where consumers whose $H_i$ is close to 0.0 are untrustworthy and those whose $H_i$ is close to 1.0 are trustworthy. Although normal consumers’ $H_i$ should usually be close to 1.0, 0.0 does not necessarily mean the consumer is an attacker. This is because it is possible that one consumer may keep offering bad feedbacks for good services unintentionally, because of uncontrollable factors (e.g., a consumer is under a bad network condition or a bad delivery service); or, a consumer may accidentally give good feedbacks for bad services. We used a default value of 1.0 for $H_i$.

The value for $Q_i$ also ranges from 0.0 to 1.0, where providers whose $Q_i$ is close to 0.0 are untrustworthy and those whose $Q_i$ is close to 1.0 are trustworthy. Similar to $H_i$, 0.0 does not necessarily mean the provider is an attacker, because uncontrollable factors (e.g., a provider is under a bad network condition or a bad delivery service) may affect the provider’s service quality. We used a default value of 1.0 for $Q_i$.

Each trust function in three reputation systems (i.e., EigenTrust, PeerTrust, TNA-SL) returns a single trust score for each user reflecting their trustworthiness. In many reputation systems, the trust score is often represented as either a trust value that ranges from 0.0 to 1.0 (i.e., a trust value-based approach) or a rank among users (i.e., a rank-based approach) [42, 132]. We thus defined two strategies for a consumer to choose a provider. One is “OVER TRUST”
with which a consumer will choose a provider whose trust value is over a certain threshold. A consumer can choose a threshold $\delta_T$ to pick a provider when the “OVER TRUST” is selected. Another is “OVER RANK” with which a consumer will choose a provider who is in the set of top-ranked providers. A consumer can choose a threshold $\delta_R$ to define top-ranked providers so that a user will be chosen as a provider if its rank is higher than $\delta_R$. If a consumer does not have any strategy, a provider will be chosen randomly. Most reputation systems employ a trust value-based approach. We thus assumed that users choose “OVER TRUST”, except when we compare trust value-based with rank-based approaches.

Although each user may behave in its own way, we assumed in this experiment that every user follows the same behavior model for simplicity. That is, every user was assumed to share the same feedback reliability rate, the same service quality rate, and the same strategy to choose a provider. Also, normal users are often expected to behave consistently regardless of reputation systems. Accordingly, we plugged in the same normal consumer/provider behavior models to COMPARS for a fair evaluation of three reputation systems. That is, if a consumer whose feedback reliability rate is 1.0 has a transaction with a provider whose service quality rate is 1.0, the consumer’s feedback about the transaction is the highest value in each system.

**Attacker:** An attacker can be a consumer or a provider. In this experiment, however, we assumed that the attacker is a malicious provider who wants to increase its profit in e-commerce markets so as to clearly show the attacker’s profits while it carries out the best actions. For simplicity, we assumed that there is a single attacker in the system who has two possible actions at each system state, i.e. provide a bad service (cheating) and provide a good service (not cheating). Note that we can add more choices of actions to deal with various types of attacks. For example, we may allow creating a new account as one of an attacker’s possible actions to capture Sybil attacks.

**Reasoning:** When an attacker reasons, COMPARS should decide the maximum height $M_T$ of an attack tree for a reasoning process, i.e. the maximum number of reasoning steps. We used 1 as default value of $M_T$.

While generating an attack tree, we produced 15 sample sequences of normal users’ actions based on their behavior models, resulting in transitions to 15 system states at each timepoint. Although an attacker is not involved in the transactions, the attacker’s trust score at each system state may or may not be different, because of global aggregations in some reputation systems [54]. For simplicity, we picked a system state as representative for the given time at which an attacker’s trust value is the average of an attacker’s trust values at 15 system states.

**Attacking goal and profit model:** We assumed the goal of the attacker is to get an amount of profits $G$ with a default value 60 by gaming the system. Table 5.2 shows how we calculated an attacker’s profits at each system state, whose rationale is explained below.
Although it is hard to define a definite relation, a number of studies have found that in e-commerce markets, the provider’s reputation has impacts on the profits it can get in each transaction [79, 97]. Also, Hazard et al. [37] and Melnik et al. [79] have found that the reputation and profits have a multiplicative relationship on e-commerce markets such as eBay. Along with previous studies, we allocated different amounts of profits that an attacker gains from a single action, depending on an attacker’s reputation. Since we used “OVER TRUST” as the default strategy for normal users to pick providers, we assumed that an attacker gains profits depending on its trust value.

An attacker will essentially try to look like normal users so that it will try to maintain its trust value similar to that of normal users. If the attacker’s trust value is too low compared to that of normal users, it would be easily identified as untrustworthy and avoided by others; therefore, the attacker will not be able to make profits [79, 97]. We thus assumed that an attacker’s profit will depend on how far its trust value is from the the average trust value of users. We divided users into 10 groups according to how far a user’s trust value is from the average trust value. Depending on the range of users’ trust values, we set a value $\alpha$. Let $T_{avg}$ and $T_{max}$ denote the average and maximum trust value of users, respectively. Then,

$$\alpha = \frac{T_{max} - T_{avg}}{5}$$

We generated 100 sequences of normal users’ actions to compute $T_{avg}$ and $T_{max}$ for the three trust functions.

An attacker in e-commerce markets attempts to make more profits by cheating [42, 55]. Clearly, if an attacker can gain a large profits without cheating, it does not need to cheat.

<table>
<thead>
<tr>
<th>Reputation</th>
<th>Profit (Cheating)</th>
<th>Profit (No cheating)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$&gt; T_{avg} + 5 \times \alpha$</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>$&gt; T_{avg} + 4 \times \alpha$</td>
<td>18</td>
<td>9</td>
</tr>
<tr>
<td>$&gt; T_{avg} + 3 \times \alpha$</td>
<td>16</td>
<td>8</td>
</tr>
<tr>
<td>$&gt; T_{avg} + 2 \times \alpha$</td>
<td>14</td>
<td>7</td>
</tr>
<tr>
<td>$&gt; T_{avg} + \alpha$</td>
<td>12</td>
<td>6</td>
</tr>
<tr>
<td>$&gt; T_{avg}$</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>$&gt; T_{avg} - \alpha$</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>$&gt; T_{avg} - 2 \times \alpha$</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>$&gt; T_{avg} - 3 \times \alpha$</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>$&gt; T_{avg} - 4 \times \alpha$</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 5.2: Profit setting
Therefore, we assumed that an attacker makes more profits by cheating, and set a larger value of profit when an attacker is cheating at a specific state.

The entries of Table 5.2 indicate that if an attacker’s trust value is over a defined trust value in the column, Reputation, it will get profits depending on its action, cheating or no cheating. The first entry, for instance, means that if an attacker’s trust value is larger than $T_{avg} + 5 * \alpha$, it will get 20 profits if it cheats; 10 profits if it does not cheat.

Even though an attacker may gain more profits by cheating at a specific system state, it does not mean that the attacker’s best strategy is to keep cheating all the time. That is because its trust value will change depending on its actions and the profit at a given system state is affected by its trust value. The relationship between attack goals and optimal strategies will be discussed in Section 5.5.

Since an attacker is a malicious provider in our experiments, we assumed for simplicity that normal providers’ $Q_i$ is 1.0. Note that a consumer chooses a provider whose trust value is over a certain threshold. Therefore, the attacker can still be chosen by a consumer as long as it maintains its trust value over the threshold.

### 5.5.2 Analysis of Existing Reputation Systems with COMPARS

In this section, we present our experimental results and analysis of the three reputation systems by plugging in parameters discussed in Section 5.5.1 to COMPARS. Note that different conclusions can be drawn from evaluations with different user behavior models, profit models and attack goals. In other words, our results do not mean an absolute conclusion that one reputation system is more or less resilient than another regardless of parameters.

Clearly, an attacker will try to choose the optimal strategy with which it obtains large profits within a small amount of time. That is, an attacker’s goal is to reduce the number of transactions to satisfy a given profit goal $G$; whereas, the goal of trust functions is to increase the number of the attacker’s transactions. We thus evaluated the number of the attacker’s transactions $N_{trans}$ to satisfy a given profit goal with different normal user behavior models (Section 5.5.2) and different number of reasoning steps (Section 5.5.2).

#### Analysis with Normal User Behavior Models

As mentioned before, we used consumers’ feedback reliability rate and provider’s service quality rate to handle non-deterministic nature. Since an attacker in our experiments is assumed to be a malicious provider, we assumed service quality rates of normal providers are 1.0 for simplicity.

Fig. 5.7 shows the required number of an attacker’s transactions $N_{trans}$ (Y-axis) with varying consumers’ feedback reliability rates $H_i$ (X-axis). Even though each user can have different
Figure 5.7: The required number of an attacker’s transactions with change in consumers’ feedback reliability rate

feedback reliability rates, we assumed that every consumer shares the same feedback reliability rate for simplicity.

A low feedback reliability rate of a consumer means that the consumer’s feedback is not consistent with true service quality. Therefore, a consumer with a low feedback reliability rate has a high probability of giving a good feedback for an attacker’s bad service (i.e., cheating). In other words, the attacker’s misbehavior will not be punished under a given trust function, if most users have low feedback reliability rates. In such a case, the attacker’s optimal strategy is cheating continuously to achieve its profit goal early, because the attacker obtains more profits by cheating at each system state and cheating will not damage the attacker’s reputation much. Consequently, the required number of an attacker’s transactions $N_{\text{Trans}}$ decreases as consumers’ feedback reliability rate $H_i$ decreases, as shown in Fig. 5.7.

EigenTrust uses the weighted sum of each user’s local trust value to compute global trust values and pre-trusted peers are responsible for a big part of the computation because of their large weight. Although pre-trusted peers should have a high $H_i$, it is possible for pre-trusted peers to issue bad feedbacks because of uncontrollable factors as discussed in Section 2.2. We thus assumed even pre-trusted peers share the same $H_i$ with other users. Hence, a low $H_i$ of pre-trusted peers allows an attacker to manipulate a system easily. That is, the resilience of EigenTrust greatly depends on the feedback reliability rate of normal users (especially that of pre-trusted peers), compared with PeerTrust and TNA-SL. Accordingly, $N_{\text{Trans}}$ under EigenTrust greatly
decreases (i.e., less resilient than PeerTrust and TNA-SL), as the feedback reliability rate gets lower.

We considered two typical strategies (i.e., OVER TRUST and OVER RANK) for consumers to choose a provider. To compare the resilience of trust functions under different consumers’ strategies, we thus assessed the number of an attacker’s transactions $N_{\text{Trans}}$ with those two strategies as shown in Fig. 5.8.

Figure 5.8: The number of required transactions to reach a goal with two strategies

(a) A trust value-based approach (OVER TRUST)

(b) A rank-based approach (OVER RANK)
PeerTrust takes 1.0 as a trust value for every user at an initial state. An attacker thus has a high trust value for a while from the beginning of transactions under PeerTrust, because of its initial parameters. An attacker under PeerTrust can thus reach a goal early (i.e., less resilient) with a high trust value compared to EigenTrust and TNA-SL; because the trust value of an attacker is still relatively high, even though it continues to cheat in the first several steps. Since an attacker can get more profits by cheating at one specific state, an attacker’s optimal strategy under PeerTrust is to cheat continuously for a while from a initial state. However, the attacker will not have a high trust value if a lot of continuous cheating actions are performed. Hence, $N_{Trans}$ increases gradually, as $G$ increases as shown in Fig. 5.8.

Similar to PeerTrust, the curve of $N_{Trans}$ shows a gradual increase under TNA-SL, as the goal $G$ increases as shown in Fig. 5.8; because TNA-SL weighs information from direct interaction. As the number of transactions increases, more direct information will be collected. Hence, the trust values under TNA-SL reflect more accurate information with more direct information, as the number of transactions increases. However, EigenTrust utilizes normalized global trust values so that the difference between an attacker’s trust values with more or fewer transactions is relatively small [54]. Therefore, $N_{Trans}$ increases in almost direct proportion to the $G$ under EigenTrust as shown in Fig. 5.8.

When “OVER TRUST” is employed, a consumer should choose $\delta_T$ that defines the minimum trust value required for a provider to be selected. If “OVER RANK” is employed, a consumer should choose $\delta_R$ to define the maximum rank for a provider to be selected. To compare the resilience of trust functions under different thresholds, we evaluated the required number of an attacker’s transactions $N_{Trans}$ with different values for $\delta_T$ and $\delta_R$.

Fig. 5.9 represents the required number of an attacker’s transactions with different values for a threshold $\delta_T$ in trust value-based reputation systems. Initially, we used the average trust value $T_{avg}$ of users for $\delta_T$ and increased by $p\%$ with the following equation until an attacker does not satisfy the required threshold.

$$\delta_T = T_{avg} + (1 - T_{avg}) \times p$$

A large value of $\delta_T$ implies that consumers set high standards for providers’ reputations. As $\delta_T$ increases, the attacker should have more honest transactions (more actions with no cheating) to build a reputation and to be selected as a provider. At each system state, an attacker’s profits from not cheating are smaller than the profits from cheating. Hence, the required number $N_{Trans}$ of the attacker’s transactions increases gradually, as $\delta_T$ increases. And, $N_{Trans}$ increases dramatically, when $\delta_T$ increases by 40 % over $T_{avg}$.

Fig. 5.10 illustrates the required number of an attacker’s transactions with different values for a threshold $\delta_R$ in rank-based reputation systems. Initially, we used the top 50 % for $\delta_R$.
and decrease by p % with following equation until an attacker does not satisfy the required threshold.

\[ \delta_R = N_U \times \frac{50 - p}{100} \]

A small value of \( \delta_R \) means that consumers set high standards of providers’ reputations. As \( \delta_R \) decreases, an attacker should behave honestly in more transactions (more actions with no cheating) to build a reputation and to meet consumers’ requirements. As mentioned above, an attacker’s profits from not cheating are smaller than the attacker’s profits from cheating at each system state. Hence, \( N_{Trans} \) increases incrementally, as \( \delta_R \) decreases.

Figure 5.9: The required number of an attacker’s transactions with different \( \delta_T \)
Figure 5.10: The required number of an attacker’s transactions with different $\delta_R$

**Effect of Different Number of Reasoning Steps**

COMPARS can choose different number of reasoning steps to make more or less accurate estimation of the attacker’s optimal strategy. Therefore, we evaluated the number of the attacker’s transactions $N_{trans}$ to satisfy a given profit goal with changes in the number of reasoning steps as shown in Fig. 5.11.

As shown in Fig. 5.11, $N_{trans}$ (Y-axis) decreased as the attacker performs more reasoning steps (X-axis). This indicates that the attacker can achieve its goal much more efficiently with more reasoning steps. The number of reasoning steps essentially mean that the attacker behaves more or less adaptively (i.e., 0 reasoning step means static behavior and more reasoning steps mean more adaptive behavior). Accordingly, Fig. 5.11 shows that highly adaptive attackers can indeed better game the system, compared to less adaptive attackers.
EigenTrust assumes that there exist pre-trusted peers who are the most trustworthy and computes trust values with a weighted sum of each user’s local trust value. Pre-trusted peers are thus assumed to follow static behavior model and the weight of them is much greater than that of other normal users. If an attacker acts badly to pre-trusted peers, it will greatly damage the attacker’s trust value, because of their weight. Consequently, it is relatively easy (i.e., less reasoning is needed) for an attacker to guess how a system works and to estimate its optimal strategy (i.e., no cheating, if a pre-trusted peer is involved in a transaction; otherwise, behave depending on each user’s behavior). In other words, the attacker can save its time in satisfying a given profit goal $G$ with a small number of reasoning steps. As shown in Fig. 5.11, an attacker can greatly reduce $N_{\text{trans}}$ with only three reasoning steps under EigenTrust.

PeerTrust assumes that every user has the highest trust value at an initial state, so that an
attacker’s first few actions will not have significant impacts on the attacker’s trust value. That is, it will be hard for the attacker to guess how a system works within a few reasoning steps. Hence, more reasoning is needed to reach a goal sooner. As shown in Fig. 5.11, an attacker will begin to reduce the number of transactions greatly with reasoning steps more than three.

TNA-SL weighs information from direct interaction. As more transactions have done in the system, users who have relatively many direct interactions may appear under TNA-SL. In such a case, the feedback of those users will have bigger impact on others. In other words, those users who had a lot of direct interactions under TNA-SL can be considered to play a similar role to pre-trusted peers under EigenTrust. Similar to EigenTrust, it will greatly damage the attacker’s trust value for an attacker to act badly to those users. It is thus relatively easy (i.e. needs less reasoning) for an attacker to guess how a system works. As shown in Fig. 5.11, an attacker can greatly reduce the number of transactions with only three reasoning steps under TNA-SL.

An interesting finding in Fig. 5.11 was that with 1 reasoning step, EigenTrust was much more resilient than PeerTrust and TNA-SL; but with 5 reasoning steps, EigenTrust and TNA-SL offer similar resilience. This corroborates that the evaluation of resilience with less reasoning (i.e., static or less adaptive attack) do not reflect the true resilience of a trust function.

5.6 Summary

The adaptive nature of sophisticated attackers presents challenging issues for the evaluation of robustness of trust functions. Specifically, reputation systems based on static user models leave opportunities for malicious parties to exploit systems easily by changing behavior arbitrarily with knowledge of trust functions. This work presents an evaluation framework for the COM- PArison of Reputation Systems (COMPARS), which models adaptive attackers. COMPARS simulates attackers’ optimal strategies with an attack tree to estimate the possible outcomes of an attacker’s action sequences up to certain points in the future. Assuming that a good trust function will restrict the opportunities for attackers to exploit a system, we estimated the robustness of trust functions against attacks by observing how many transactions are required for attackers to achieve their goal. We have shown the validity of the attack tree with experimental results, which indicated that the number of an attacker’s transactions decreases with more reasoning steps. This finding indicates that the attack tree effectively captures the changing behavior of attackers and with more reasoning steps, we can get a more accurate estimation for an attacker’s optimal strategy.
Chapter 6

Detecting Opinion Spammer Groups through Community Discovery and Sentiment Analysis

6.1 Motivation

Review systems are indisputably important resource for users when making various decisions on products or services. Consequently, they become increasingly targeted by attackers who deliberately inject opinion spams [69, 82, 83, 90, 107, 120, 129]. Opinion spams refer to malicious and biased contents that aim to influence normal users’ decisionmaking for profit.

While a number of methods have been proposed to detect opinion spam [25, 27, 68, 83, 90, 91, 120, 129], most of them focus primarily on developing pure content-based classifiers. The basic idea behind these approaches is to detect opinion spam through the analysis of review content. More specifically, they include supervised approaches that compare unlabelled content with ground truth data [49, 90], and unsupervised ones that exploit abnormal features in review content [82]. Such pure content-based classifiers, however, are limited for several reasons. First, spammers can easily manipulate review content to avoid detection [83, 114, 121]. For example, if duplicated text reviews are considered to be spams, spammers may simply paraphrase the content [110]. Second, they are often designed for specific application domains such as travel reviews, and cannot be applied easily to different domains such as movie reviews [90]. Third, while supervised methods generally require ground truth labels, it is often hard to obtain for...
real datasets. Some previous researchers have hired human experts to manually label data. The high cost of such a human-driven approach, however, makes it impossible to do so reliably for large-scale datasets [82].

In this work we explore an alternative approach by examining what we call promotional opinion spammers through the analysis of user relationships rather than review content. Promotional opinion spammers refer to attackers who try to improve the influence of their opinions by malicious artificial boosting. For example, many review systems employ some sort of reviewer/review ranking systems e.g., a top reviewer list on Amazon, most helpful reviews on Amazon, or most recommended reviews on Yelp. Spammers may thus artificially boost the rank of their reviews to attract more attention; spammers may also artificially demote the rank of competitors’ reviews.

To obtain high ranking, spammers need to collect significantly more positive responses than negative ones. For example, review and reviewer ranks on Amazon are based primarily on the number of helpful votes received. Similarly, spammers need to inject a significant amount of negative responses to competitors’ reviews to demote competitors’ ranking.

Since multiple votes from the same user on one review are often counted as one vote, spammers need to boost their ranks by gathering positive votes from different users (i.e., colluders). One possible way to do this is for spammers to collaborate to vote high. We thus hypothesize that such malicious artificial boosting activities would eventually lead to construct spammer communities in which spammers are strongly positively connected with each other through review-response interactions (e.g., votes and text replies on the review). Similarly, spammers can collaborate to vote down for their competitors. As spammers will not be likely to vote down for each other but for competitors, we hypothesize that malicious demoting activities would eventually lead to build one-directional negative connections from spammer groups to competitors. Our first goal is thus to find these strongly or even abnormally positively connected communities among users and we argue that it is more likely to detect collusive spamming behavior among these users than those who are not part of these communities. Then, we investigate negative connections from discovered communities to others.

Our work is grounded in the context of a review ecosystem on Amazon. In our first work we identified the existence of implicit communities of different strengths through review/response activities on Amazon. In this work we further explore positively and negatively connected communities through review and response activities via sentiment analysis. The intuition behind this approach is that: if a user has an unusual positive or negative relationship with another, they may be posting fraudulent positive and negative responses to each other’s items and/or reviews to boost or demote the reputation of specific reviews or reviewers. We investigate community structures and their distinguishing characteristics that can be used to detect spammers.
In our approach, we first build general user relationship graphs representing how users interact with one-another; Then, we derive the sentiment of each relationship by aggregating sentiments of all responses between any two users. We then extract positive relationship graphs from the general user relationship graphs to capture boosting behavior. More specifically, motivated by link-based web spam detection, we focus on strongly connected communities in positive relationship graphs. Finally, we analyze extracted strongly positively connected communities and their negative connections to find opinion spammer groups and their competitors.

Note that non-spammers may also form natural communities based upon their genuine similar interests [20]. However, we argue that spammer communities have distinguishing characteristics in terms of structures and the strength of their relationships. Concretely, we show that the stronger a community the user appears in, the more likely the user is involved in spamming-like activities. In this work we employ two existing content-based classifiers to measure users’ spammicity [82, 90]. Our experiments over a dataset collected from Amazon suggest that our community-based scheme achieves the same effectiveness in terms of spam detection as the state-of-art content-based approach targeting at Amazon spams.

6.2 Related Work

There has been a lot of research on combating spams in the context of Web and Email, which can be classified into two categories: content-based and link-based approaches [1, 14, 28, 33–35, 87, 102, 111].

Content-based approaches analyze the content of the webpages including irrelevant contents, malicious urls, and unsolicited commercial advertisements [8, 34, 39, 87, 111].

Ntoulas et al. introduced a few heuristic methods that exploit content features of webpages including word distributions [87]. Then they proposed a supervised spam detection method using those features. They first manually generated a training dataset, proposed a few heuristics, and classified unknown dataset using a few traditional classification techniques such as decision-tree based, rule-based, and support vector machines.

Ribeiro et al. proposed a method to extract pages linked by URLs in spam messages and characterize the relationships between those pages and the messages [98]. Then, they employed a lazy associative classifier to extract classification rules from the web pages relevant to spam detections.

Link-based approaches, on the other hand, specifically target to detect link-based web spams with which spammers try to boost their page rankings and get popularity by building strongly connected page links. To detect link-based web spams, link-based approaches leverage the properties of the link structure [14, 102, 133].

Broder et al. investigated the in/out-degree distributions [12]. They showed that the in/out
degree distributions of webpages follows Zipf distributions, and outliers in the distributions are likely to be web spams.

Fetterly et al. investigated statistical distribution of content (e.g., word counts) and link features (e.g., the in/out degree distributions) to identify machine-generated web spams. They showed their approach using statistical analysis successfully captures the characteristics of web spams with 14% false positives.

Another literature of spam research has been conducted on various applications including social networks [9, 47, 110, 119] and blog space [10, 57, 80]. Many of them focused on characterizing spamming behavior with suspended accounts in Twitter. For example, Grier et al. studied Twitter spams such as phishing attacks and targeted spams by analyzing click-through behavior, and identified two types of accounts: actual spam accounts and compromised accounts [32].

Thomas et al. analyzed tweets from suspended users, and characterize a variety of spamming behavior including creating new Twitter accounts, generating spam URLs, and distribution spams [118].

Ghosh et al. investigated link farming activities in Twitter [31], and proposed methods to deter link farming. To be specific, Ghosh et al. proposed a ranking algorithm penalizing spammers.

Mishne et al. compared linguistic features of blog posts and blog comments [80]. With consideration of a random nature of spam comments, they built language models for blog posts and comments, and measured the discrepancy between language models using KL-divergence.

Kolari et al. first introduced a few features including bag-of-anchors, bag-of-urls [57]. Then they employed support vector machines to classify blogs as spam/non-spam.

In recent years, since first introduced by Jindal et al. [49], there is a growing body of research about a new type of spams, called opinion spams [36, 68, 69, 82, 83, 90, 91, 107, 120, 129]. In fact, a few researchers and news medias have pointed out the importance of opinion spam detection, as they showed how prevalent opinion spams are in real review-systems such as TripAdvisor and Yelp [7, 76, 84, 88, 122]. For example, Ott et al. reported 15% of reviews in TripAdvisor are spams [88], and Dellarocas reported many book reviews in Amazon were written by book authors and publishers [21]. Also, Yelp admitted a quarter of submitted reviews might be spams [7].

Despite the importance, opinion spam research is still widely open; as it is often hard to get ground truth for opinion spam, unlike traditional spam analysis in the context of Web and email. Previous research thus employed different mechanisms to obtain ground truth data. Early work including [27, 49, 50, 65, 125] manually inspected reviews and extracted simple features. Jindal et al. classified reviews as spam/non-spam by detecting duplicate/near-duplicate reviews.
Li et al employed 10 college students to manually label randomly chosen reviews as spam/non-spam given a few intuitive features [63]. Then they introduced two-view semi-supervised methods to classify unknown dataset given the labeled dataset. The authors observed four spam review features and two spammer features based on which they classify unlabelled reviews and reviewers.

Rubin et al. characterize sellers according to statistical model with anomaly detection technique for the purpose of detecting price-inflated behavior in online auction communities [100].

Liu et al. proposed an algorithm combining a temporal analysis and a user correlation analysis to identify products under malicious ratings in e-commerce [68]. To do so, Liu et al. investigated whether rating changes dramatically or accumulatively over time.

Similarly, there has been several research to capture unexpected rating patterns [25, 27, 50, 65, 125]. Whereas these approaches can capture reviewers’ unexpected behavior, they depend largely on heuristics that cannot be applied to different domains, which is limited [82, 83].

More recently, a few researchers generated ground truth set by hiring human experts to manually label reviews [82, 83] or by hiring online workers, called Mechanical Turkers, to write spams [89, 90].

In [89, 90], researchers have generated ground truth data containing 400 truthful reviews and 400 opinion spams by hiring Turkers to write spams. They then developed content-based classifiers that compare the linguistic features of truthful and spam reviews. While these classifiers have been shown to be successful, it is not clear whether they can be applied reliably in other domains because they are very content specific. For example, linguistic features of hotel reviews may be different from those of electronics reviews. More importantly, there have been unresolved debates on whether datasets generated by Turkers for research purpose can be representative of actual spams in practice [82, 84].

NetProbe leverages Markov Random Field in modeling the auction users to search the suspicious pattern of fraudster, and uses a Belief Propagation mechanism to detect fraudulent behavior [91]. While doing so, the authors manually label the propagation matrix from available data.

Mukherjee et al. generated ground truth by hiring domain experts who manually detected spams given a few intuitive features [82, 83]. The authors observed some abnormal behavior regarding spam, and they classified the typical behavioral features of opinion spam and spammers into nine indicators.

While existing efforts discussed above present promising results, it is often easy for spammers to avoid content-based spamming detection by making superficial alterations to their reviews [83, 114, 121]. Also, such pure content-based detection methods often need to develop different classifiers for each purpose and domain [89, 90]. Furthermore, it is often hard, if not impossible,
to manually generate ground truth set reliably for large-scale datasets [82]. By contrast, our approach detects spammers by analyzing user relationships and communities built through unusual interactions; which is much harder to fake than to reword their review content, as we shall describe later in this chapter.

6.3 Discovering Opinion Spammer Groups and Competitors

Our proposed approach aims to detect opinion spammer groups who artificially form communities through the coordinated positive interactions, and to find their competitors.

Our approach can be briefly divided into four stages and Fig.6.1 depicts the general four stages through four sub-graphs, one sub-graph per stage. The four stages are: 1) building a general user relationship graph, Fig. 6.1(a); 2) annotating the general graph through sentiment analysis, Fig. 6.1(b); 3) pruning the general graph to positive relationship graph, Fig. 6.1(c); and finally 4) identifying strongly positively connected communities within the positive relationship graph, Fig. 6.1(d). In the following, we will describe each stage in more details.

6.3.1 Stage 1: Building a General User Relationship Graph

We extend the definitions of a user relationship and a community proposed in the first work, which we briefly describe in this section.
We represent users and their interactions on a review system as a directed multigraph $G = (U, E)$ in which $U$ represents users (vertices) and $E$ represents interactions (edges). Each edge $e_{uv}$ is a 2-tuple $(u, v)$ having direction from a commenter $u$ to a reviewer $v$. A commenter has outgoing edges, and a reviewer has incoming edges in a graph. An out-degree of commenter $u$ is the total number of edges from $u$ to other users and an in-degree of reviewer $v$ is the total number of edges from other users to $v$. Note that multiple comments of user $u$ on the same review of user $v$ will count as one interaction and thus the number of edges from $u$ to $v$ equals to the number of different $v$’s reviews which $u$ commented on.

Generally, the in-degree of $v$ essentially reflects $v$’s tendency as a reviewer (i.e., how popular $v$ is to get comments); while the out-degree of $u$ reflects $u$’s tendency as a commenter (i.e., how much $u$ is willing to comment). We further model a users tendencies as a reviewer and a commenter using incoming and outgoing probabilities defined as a reviewer’s probability to get incoming edges and a commenter’s probability to generate outgoing edges respectively.

Generally speaking, if we assume that there is no external relationship between users $u$ and $v$, the typical interaction between a commenter and a reviewer can be modeled as a random process. User $u$ simply stumbles upon $v$’s review by chance when browsing the system. He does not know $v$ and seek out $v$’s review deliberately. In other words, if there was no prior relationship from $u$ to $v$, interactions from $u$ to $v$ should happen randomly depending on $u$’s tendency as a commenter and $v$’s tendency as a reviewer. Accordingly, we can represent all users’ interactions as a random graph in which edges (i.e., interactions) are randomly created following the incoming/outgoing probability of each user. As a result, we get a random graph $G_r = (U, E')$ in which the total number of all edges and each user’s degree distribution are the same as the original interaction graph. The random graph thereby preserves the same nature of each individual as a commenter or a reviewer, which is independent of any prior relationships between users. The only main difference between the two graphs is that: all edges are randomly generated in the random graph and so the number of edges between each pair of users will be different from original graph.

Given the random graph model, we examine the real interaction patterns in a review system and see how much they are deviated from the random graph. We define users’ relationship and its strength based upon the distance between users’ original interaction graph and its corresponding random graph. Intuitively, the larger the difference between the real interaction pattern and the random model is, the more likely the relationships are to have been artificially orchestrated. We measure the distance by building confidence intervals based on the random graph. We denote that $u$ has a relationship with $v$ with $\tau$ strength, when the probability for edge $e_{uv}$ to form in the real graph is outside of the given confidence interval $\tau$. Then, the larger $\tau$ a relationship has, the farther the real interaction pattern is from the random graph and thus
the higher strength the relationship has.

The concept of strength can be naturally extended to communities. Concretely, edge $e_{uv}$ (in turn, user $u$ and $v$) belongs to $\tau\%\text{community}$, if the strength of a relationship from $u$ to $v$ is $\tau$. The larger $\tau$ is, the higher strength relationships in a community have and thus the higher strength the community has.

For this work it is important to note that relationships belonging to higher strength of communities are excluded from lower ones. For instance, if a relationship is in 99.5% community, it is excluded from all lower strength of communities such as 98%.

Given the definitions above, we extract separate user relationship graphs for each $\tau$ community in which vertices are users and edges are their relationships defined by interactions, as illustrated in Fig. 6.1(a). Fig. 6.2 presents examples of user relationship graphs in Amazon. As we exclude higher strength of relationships from lower ones, relationships in Fig. 6.2(a) do not appear in Fig. 6.2(b).

6.3.2 Stage 2: Sentiment Analysis on User Relationships

Given user relationships graphs, we further analyze the sentiment of relationships. To do so, we aggregate the sentiments of all comments between any pair of users from which we derive the sentiment of each relationship.

If comments are in the form of explicit votes, it is straightforward to obtain sentiment values. However, in many systems including Amazon and Yelp, it is often unknown who made
the vote, while only aggregated information is publicly available. For example, we may know a
certain review got 50 positive votes total, but we cannot know who made those votes. We thus
focus specifically on the commenting text in order to define the sentiment of a relationship.
We chose to employ a publicly available tool, AlchemyAPI [3], for this purpose. AlchemyAPI
is often used to generate ground truth sentiment data and is known to present high accuracy
on identification of sentiments in various applications including reviews and tweets [94, 109],
which gives us good confidence in using the API.
AlchemyAPI takes text contents as input, identifies a sentiment of the text contents, and
output sentiment score. The score ranges from -1 to 1, where positive/negative scores represent
the strength of positivity/negativity, and 0 means neutral.
There are many possible ways to derive the sentiment of a relationship from the sentiment
of each comment. In this paper we employ a straightforward approach where the sentiment
of a relationship from commenter $u$ to reviewer $v$ is the average of the sentiments of all $u$’s
comments on $v$’s reviews. Specifically, to decide whether a relationship between users $u$ and $v$
is positive or negative, we first analyze the sentiments of all comments between $u$ and $v$, and
aggregate them. We then build relationship graphs in which sentiments of all relationships are
analyzed, as illustrated in Fig. 6.1(b). We consider the relationship is positive if the average
sentiment score is bigger than 0. Fig.6.3 shows examples of positive interactions between two
users who are defined to have a positive relationship by our approach in Amazon.

6.3.3 Stage 3: Positive Relationship Graphs
Once sentiments are analyzed, we prune the user relationship graphs to build positive relation-
ship graphs by extracting or keeping only positive relationships (Fig. 6.1(c)).

6.3.4 Stage 4: Identify Spammer Candidates by Decomposing Positive Re-
relationship Graphs
We propose to identify spammers by analyzing community structures and the strength of re-
relationships. Note that we are interested in spammer groups who work together, not individual
spammers. As mentioned before, to boost their opinions, each spammer needs to collect a sig-
ificant amount of positive interactions from others, usually her colluders; as it is expected that
non-spammers rarely post positive comments to spam in general, whereas groups of spammers
post positive comments to each other so that most of them can obtain a dominant position
(i.e., reviewers whose opinions are believed to be trustworthy) in a system. In other words,
interaction from a non-spammer to a spammer are not likely to appear in positive relationship
graphs; and spammers will have strong interconnections through positive interactions. This
motivates us to extract strongly connected components from positive relationship graphs. In
other words, we believe that deliberate positive interactions among spammers will lead to the formation of strongly connected communities in a positive relationship graph, as illustrated in Fig. 6.1(d). Accordingly, we cast the problem of detecting opinion spammers as the problem of finding strongly positively connected communities.

A strongly connected component $G' = (V, E')$ is a subgraph of given graph $G = (U, E)$ such that there is a directed path in each direction between every pair of vertices $u, v \in V \subset U$. In our context, we define a strongly positively connected community $G' = (V, E')$ as follows.

**Definition 6.** $G'$ is a strongly positively connected community, if $G'$ is a subgraph of positive relationship graph $G$ such that
i) $\exists$ at least two vertices in $G'$
ii) $G'$ is a strongly connected component of $G$
iii) $G'$ is maximal, i.e., $\nexists$ strongly connected component $H \subset G$ containing $G'$.

We find all strongly positively connected communities in each positive relationship graph.
and consider them as possible spammer candidates.

As noted before, non-spammers may form natural communities due to their similar interest on items. For example, in a random fashion, $u$ has a few positive interactions with $v$ through multiple items. This is likely to happen because $v$ may have reviewed similar items, and $u$ may look at those items to buy so that multiple interactions from $u$ to $v$ occur. And natural communities can arise from such random relationships. On the other hand, spammers construct artificial non-random communities. We thus argue that we can differentiate natural and spammer communities by measuring the level of randomness within the relationships. By our definition in Section 6.3.1, the strength of the relationships captures such a level of randomness, which we have in fact shown in Chapter 4. We show that how the strength and spammicity are correlated in Section 6.4.

6.3.5 Stage 5: Building a Negative Relationship Graph with Discovered Communities and Their Negative Neighborhood

Once we identify all strongly positively connected communities, we build negative relationships graphs by extracting negative relationships of discovered communities, appearing in the sentiment annotated-general relationship graphs. Fig.6.4 shows examples of negative interactions between two users who are defined to have a negative relationship by our approach in Amazon.
6.4 Experimental Results and Analysis

In this section we will present the characteristics of the discovered reviewers who appear in the strongly positively connected communities identified by our approach. We will present correlations between the strength of relationships among the discovered reviewers and spammnicity.

In our experiment, we investigated the characteristics of the discovered reviewers in each category individually as well as Across dataset. We report results with two individual category dataset - Books and Movie datasets. We also report results with Across dataset to investigate the behavior of spammers who launch attacks across categories. As the same patterns and observations were found for other categories, we primarily discuss results for the Across dataset.

We will compare three groups of reviewers: the discovered reviewers identified by our approach, the top reviewers recognized by Amazon, and the total reviewers which includes all reviewers appearing in our dataset. Amazon provides a list of 10,000 top ranked reviewers and to be on the list, a reviewer needs to demonstrate credible resources of high-quality reviews. Since Amazon is a well-known system, we assume that most top reviewers are trustworthy. Based on this assumption, we focus primarily on comparing our discovered reviewers with Amazon’s top reviewers to show that although the discovered reviewers can appear to be as “helpful” as top reviewers, they are strong spammer candidates.

We begin by presenting statistical characteristics of discovered reviewers in Section 6.4.1. Then, we compare the behavior of three groups of reviewers in terms of verified purchase ratio (Section 6.4.2), and spammnicity level introduced in previous work including [90] (Section 6.4.3) and [82] (Section 6.4.4). In sections 6.4.3 and 6.4.4, we further compare our community-based detection to the state-of-the art content-based approaches. Finally, in Section 6.4.5, we compare two classifiers.

6.4.1 User Statistics

Fig. 6.5 shows the number of reviewers in each discovered community. The x-axis demonstrates the strengths of the discovered communities; the y-axis presents the number of discovered reviewers in each strength of communities. In the Across dataset, no communities were found with strengths 10% ∼ 40% and 70%, and in the Movie dataset, no communities were found with strengths 10% ∼ 30% and 70%. So they are not presented in the following graphics.

First and most importantly, we measured the average number of reviews of three groups of reviewers as shown in Fig. 6.6. The x-axis represents the strengths of the discovered communities; the y-axis represents the average number of reviews reviewers in each strength of communities submitted. We have one line per reviewer group.

Fig. 6.6 shows that both discovered and top reviewers have reviews more than 100 on average while the average number of reviews of total reviewers are relatively low, with less than
10 reviews on average. This result agrees with the observations of prior researchers who found that the majority of reviewers writes only a few reviews [122, 125]. One important observation from Fig. 6.6 is that both the discovered reviewers identified by our approach and the top reviewers ranked by Amazon are active reviewers (i.e., who actively participate in discussion on items) and more importantly, our discovered reviewers are much more active on average than the top reviewers regardless of the strength of communities they are in. For instance, in the Across dataset, > 300 reviews on average for discovered reviewers vs. 150 for the top reviewers. Additionally, the higher strength of communities (e.g., 99.5 % and 98 %) had more reviews on average than those in the lower strength of communities (e.g., 0 %). For example,
Figure 6.6: The average number of reviews (RNUM) in each category

in the Across dataset, reviewers in 98% ~ 99.5% had reviews > 450 on average, but reviewers in a 0% community had reviews in 300 – 400 range on average.

As our goal is to find opinion spammer groups who maliciously and artificially boost their reviews, we first need to know whether their opinions are actually to be boosted in the system (i.e., whether their opinions are marked as worth reading and helpful). In a common sense, reviews marked as helpful will have more influence on others. We thus calculated the positive vote ratio (PVR), ranging from 0 to 1, of the three groups of reviewers. We calculated PVR for each reviewer as the percentage of positive votes over the total number of votes the reviewer got and then averaged it over the groups. The higher PVR is, the more helpful their reviews are appeared to be in general.
Fig. 6.7 shows the average PVRs of the three groups of reviewers. The x-axis demonstrates the strengths of the discovered communities, and the y-axis presents the PVR. We have one line per reviewer group.

As shown in Fig. 6.7, the PVRs of the discovered reviewers are relatively high and in fact, they are close to the PVR of the top reviewers, nearly 80%. Both groups are much higher than PVR of the total reviewers whose value is closer to 55%. This indicates that the opinions of discovered reviewers do indeed appear to be quite helpful in general, as much as that of the top reviewers. Additionally, PVRs of discovered reviewers vary across different strengths and 60% community has the lowest PVR ratio: close to 70%. In following sections we show that although the PVR analysis indicates that opinions of discovered reviewers may have a similar
level of influence on others as that of top reviewers, they are more likely to be spammers.

6.4.2 Verified Purchase Analysis

Amazon tags each review with Verified Purchase to indicate whether or not the reviewer made a purchase through Amazon. Although it is not the case that every non-spammer reviewer made a purchase through Amazon, reviewers who purchased the item in question are less likely to be spammers than those who submitted a review without doing so. We therefore defined the verified purchase ratio (VPR) as the percentage of verified reviews over the number of total reviews of each user and believe that VPR is good indicator for spammicity.

Figure 6.8: Verified purchase ratio (VPR) in each category
Fig. 6.8 shows the average VPRs of the three groups of reviewers. The x-axis demonstrates the strengths of the discovered communities, and the y-axis presents the VPR. We have one line per reviewer group.

Interestingly, it shows that there was no difference between the top and the total reviewers in terms of their VPRs. In other words, the top reviewers were no more likely to purchase the reviewed item than normal users. As we expected, our discovered reviewers have lower VPRs than the other two groups in general except for the 60% communities. In fact, the VPRs for the 80% ~ 99.5% communities are substantially lower than those for the top and the total reviewers.

For the reviewers in the 0% ~ 60% communities we see that the stronger the community the higher the VPRs observed. However, as shown in Fig. 6.8, the trend is different for reviewers in the 80% ~ 99.5% communities. In that case the strength of the community is negatively correlated with VPR. We believe that this occurs because the members of those communities are more likely to be spammers as we will show in the following sections.

6.4.3 Analysis on Spammer Classification with a Linguistic Model-based Classifier

6.4.3.1. Spammicity Analysis

In this subsection we will measure the spammicity of the discovered reviewers across the various community strengths using a linguistic model-based classifier proposed in [90]. Ott et al. trained nine machine learning models based on ground truth set including truthful reviews extracted from TripAdvisor and spam reviews written by Turkers. We hypothesize that the models built from ground truth set used by [90] can be directly applied to estimate the spammicity of reviews on Amazon, and we applied the models to our dataset.

Among nine machine learning models, we employed unigram Naive Bayes Classifier. Following [90], a 5-fold cross validation procedure was performed on the ground truth set, the details of which can be found in [90]. Based on the model, we measured the spammicity by estimating the fake probability of each review of discovered reviewers. The fake probability is the likelihood of each review being spams, which is computed by the Naive Bayes Classifier. Fig. 6.9 shows an excerpt from a review classified as 99.99999% fake probability.

We define the fake probability of a reviewer as the average fake probability of all of her reviews. Fig. 6.10 shows the average fake probabilities of the three groups of reviewers. The x-axis demonstrates the strengths of the discovered communities, and the y-axis presents the average fake probability. We have one line per reviewer group.

Unexpectedly, in Books and Movie datasets, we could not find any clear patterns in terms of the fake probability when varying the strength of communities, as shown in Fig. 6.10. Moreover, the fake probabilities of the three groups of reviewers are not much different from each other.
Although in Across dataset, we see the fake probability increases as the strength of communities increases, the average fake probability of top reviewers was also relatively high, compared to that of total reviewers. This result is inconsistent with our assumption that top reviewers are the most trustworthy reviewers. We discuss possible reasons behind the inconsistency in Section 6.4.3.3 and Section 6.4.5.

6.4.3.2. Analysis on Spammer Classification

In this section we further report results on whether discovered reviewers were classified as spammers according to the linguistic model-based classifier. Since Ott et al. focused on review contents not the reviewers, we were not able to get ground-truth labels for reviewers in TripAdvisor dataset. We instead sort total reviewers based on their fake probability in descending order, and assume that top ranked reviewers are spammers and bottom ranked reviewers are non-spammers. We believe that it is a plausible approach as a reviewer will be more likely to be spammers if most of her reviews has high probability to be spams, in turn, resulting the high average fake probability. As previous work reported 15% spam ratio in TripAdvisor, we assume that top and bottom 15% ranked reviewers can be classified as spammers and non-spammers respectively [88, 90]; which is used for ground-truth labels for this analysis. We then measure accuracy to find correlation between discovered reviewers and the classification results by the linguistic model-based classifier.

Fig. 6.11 shows ROC curve by varying different strengths as thresholds to define spammers. The x-axis represents the false positive rate and the y-axis represents true positive rate. Each point represents true positive rate against false positive rate given $\tau$ strength as a threshold.

We assume that reviewers in communities with strengths greater than or equal to $\tau$ are
spammers; those in communities with strengths less than $\tau$ are non-spammers. For example, a point labelled as 90% represents that we assume reviewers in 90% $\sim$ 99.5% communities are spammers and those in 0% $\sim$ 80% communities are non-spammers. Note that in Fig. 6.32, we present results regarding whether or not discovered reviewers in different strengths of communities are spammers. We thus do not plot when 0% is used as a threshold, as we could not get false or true negative results.

As shown in Fig. 6.11, our community-based detection yields all points in the bottom right corner. This indicates that reviewers discovered by our approach seem to be in discord with reviewers detected by the linguistic model-based classifier. Accordingly, these results might lead to the conclusion that our approach fails to detect spammers involving reviews with linguistic

Figure 6.10: Fake probability in each category
features of spams. However, we argue that the unexpected results were due to an unfair comparison between linguistic features of Amazon reviews and TripAdvisor reviews, which we will discuss further in Section 6.4.3.3 and Section 6.4.5.

6.4.3.3. Discussion

In this section we discuss why results in Section 6.4.3.1 and Section 6.4.3 go against our hypothesis. In particular, we delineate that the linguistic model-based classifier proposed in [90] greatly depends on the ground truth set.

Fig. 6.12 lists examples of most significant features in TripAdvisor ground truth set learned by the Naive Bayes Classifier, showing that the relative frequency of each feature in truthful
contains(block) truthful : spam = 15.7 : 1.0
contains(indoor) spam : truthful = 15.0 : 1.0
contains(reviews) truthful : spam = 13.8 : 1.0
contains(vacation) spam : truthful = 13.0 : 1.0
contains(street) truthful : spam = 12.2 : 1.0
contains(noise) truthful : spam = 10.3 : 1.0
contains(pet) spam : truthful = 10.3 : 1.0
contains(hear) truthful : spam = 9.7 : 1.0
contains(train) truthful : spam = 9.7 : 1.0
contains(luxury) spam : truthful = 9.6 : 1.0
contains(accommodations) spam : truthful = 9.0 : 1.0
contains(floor) truthful : spam = 8.5 : 1.0
contains(cost) truthful : spam = 8.3 : 1.0
contains(speed) spam : truthful = 8.3 : 1.0
contains(small) truthful : spam = 7.9 : 1.0
contains(public) truthful : spam = 7.7 : 1.0
contains(elegant) spam : truthful = 7.3 : 1.0
contains(parking) truthful : spam = 7.2 : 1.0
contains(expensive) truthful : spam = 7.0 : 1.0
contains(upgraded) truthful : spam = 7.0 : 1.0
contains(we) truthful : spam = 7.0 : 1.0
contains(luxurious) spam : truthful = 5.2 : 1.0

Figure 6.12: Most significant features learned by Naive Bayes Classifier

and spam reviews. For example, Fig. 6.12 indicates that the word “block” tends to appear in truthful reviews 15.7 times more often than in spam reviews.

The Naive Bayes Classifier proposed in [90] basically extracts those significant features from the ground truth set, and check their frequencies in unknown reviews. In other words, the accuracy of the classifier often depends largely on the reliability of features of ground truth set. Those features, however, may not be applied to different domains of dataset. For example, adjectives such as “expensive” and “elegant” might be used through multiple domains; whereas words such as “accommodations” and “parking” might be applicable only to hotel reviews, not to book reviews. Moreover, the words themselves such as “expensive” are not suspicious at all that can appear in any truthful reviews.

Moreover, a few researchers suggested the linguistic model-based classifier does not work
well in some applications including Yelp [82, 84]. We discuss this inconsistency through different applications in Section 6.4.5.

6.4.4 Analysis on Spammer Classification with a Spam Indicators-based Approach

6.4.4.1 Spammicity Analysis with Spam Indicators

In this subsection we will measure the spammicity (i.e., how likely users are to be spammers) of the discovered reviewers across the various community strengths with a spam indicators-based approach. We used nine content-based spam indicators suggested by existing research to measure the level of spammicity of reviewers [82–84]. Each value for the spam indicators ranges from 0 (non-spammers) to 1 (spammers).

In the following we will describe each of the measures used for our study (i.e., nine spam indicators) and will present the results in a series of plots where: the x-axis demonstrates the strengths of the discovered communities; the y-axis presents the strength of the appropriate measure; and we have one line per reviewer group.

**Content similarity (CS):** measures how similar the user’s reviews are, as spammers often copy their own reviews across items. Following [82–84], we measure the maximum of pairwise content similarities of two reviews by the same reviewer to capture the worst case. Fig. 6.13 presents the average CSs of the three groups of reviewers. Mukherjee et al. stated that the expected value of CS of spammers was 0.7 [82]. As shown in Fig. 6.13, we observe that the CSs of reviewers in 80% ∼ 99.5% communities are over 0.7 in the Across and the Books dataset. Note that there is a big drop between the 80% community and 60% community, and the CSs of 0% community is very close the CSs of total reviewers. This result suggests that 80% ∼ 99.5% communities are more likely to be spammers with much higher CSs than the lower strength of communities.

**Rating abused item ratio (RA):** checks whether a user posted multiple reviews with similar ratings on the same item, as non-spammers post multiple reviews usually when her opinion changes. Following [82, 83], we measured the similarity by computing the difference between the maximum and minimum ratings of each reviewer for an item; and we assumed a reviewer abused ratings, if she posted the similar ratings more than twice on the same item. We then measured how many items were involved in rating abuse for each user. Fig. 6.14 presents the average RAs of the three groups of reviewers. In general, non-spammers are not likely to involve in rating abuse. Indeed, RAs of reviewers in 0% ∼ 60% communities and top reviewers are close to zero, whereas RAs of reviewers in 80% ∼ 99.5% communities range from 0.2 to 0.4.

**Review duplicated items ratio (DUP):** checks whether a user posted similar multiple reviews on the same item. Although DUP is similar to RA, it focuses on review contents, not
ratings. DUP can thus capture similar multiple reviews by a spammer with multiple identifiers. Following [82, 83], we assume that reviews are spams if the content similarity between two reviews on the same item is over 0.72, and measured how many items a user is involved in review duplication. Fig. 6.15 presents the average DUPs of the three groups of reviewers. In general, non-spammers are not likely to involve in such activities. Indeed, DUP of reviewers in 0% ~ 60% communities and top reviewers are close to zero, whereas that of reviewers between 95% and 99.5% ranges from 0.1 to 0.4.

**Maximum one day review ratio (MOR):** measures how many reviews a user posted in one day compared with the maximum across all reviewers, as a massive amount of reviews in one day often looks suspicious. In our dataset, the maximum number of reviews per day were

Figure 6.13: Contents similarity (CS) in each category
60 (Movie), 95 (Books), and 96 (Across), which we can undoubtedly say is suspicious amounts of reviews for a single day. Fig. 6.16 shows the average MORs of the three groups of reviewers. Mukherjee et al. stated the maximum number of reviews per day was 21 in their dataset, and the expected MOR of spammers was 0.28 and that of non-spammers was 0.11 (i.e., the expected number of reviews in a day of spammers was $0.28 \times 21 \approx 5$ and that of non-spammers was $0.11 \times 21 \approx 2$) [82]. The maximum number of reviews per day was higher in our dataset than that used in [82] and this produced a correspondingly different MOR. However, we found that the number of maximum reviews in a day ranged from 7 ($\approx 0.07 \times 96$) to 17 ($\approx 0.18 \times 96$) for reviewers in the $80\% \sim 99.5\%$ communities, which is more than the expected number of reviews per a day for spammers; whereas it was 3 ($\approx 0.03 \times 96$) for those in $0\% \sim 60\%$ communities,
which is similar to the expected number of reviews per a day for non-spammers. It is interesting to see that the MOR of the top reviewers was also relatively high, compared to the MOR of the total reviewers. One possible reason might be that Amazon invites some top reviewers to get advance access to not-yet-released items and to write reviews [4].

**First review ratio (FRR):** measures how many of user’s reviews are the first review for the target item, as spammers often post reviews early in order to maximize the impact of their reviews. Fig. 6.17 presents the average FRRs of the three groups of reviewers. As shown in Fig. 6.17, the top and the total reviewers have very close FRRs overall but for our discovered reviewers, we observe that FRR increases, as the strength of a community increases. Note that this result may simply reflect the fact that reviewers in the higher strength of communities

Figure 6.15: Review duplicated items ratio (DUP) in each category
are more active and thus are more likely to author the first review. This may explain that top reviewers also have relatively high value, compared to total reviewers in Books and Movie category. However, the high FRRs for reviewers in 80% ~ 99.5% communities still reflect their spamnicity, when combined with other spam indicators.

**Early time frame ratio (ETF):** measures how early a user reviewed the item. The intuition behind ETF is the same as for the FRR, because if not the first review, earlier reviews may have a bigger impact. Mukherjee *et al.* estimated the appropriate threshold to decide whether the review is written early [82, 83]. We employed the same threshold (0.69), and measured the percentage of a user’s reviews that were written early. Fig. 6.18 shows the average ETFs of the three groups of reviewers. As shown in Fig. 6.18, we observe similar results to FRR
so that ETF increases, as the strength of a community increases.

**Deviated rating ratio (DEV):** checks the difference between a user’s rating and the average rating of other users for the same item, as spammers often try to inflict incorrect projections which deviate from the common consensus. We employed a threshold (of 0.63) estimated in [82, 83] to decide whether a rating is deviated, and measured the percentage of a user’s reviews that are deviated. Fig. 6.19 shows the average DEVs of the three groups of reviewers. Note that DEV of the top reviewers is the lowest. This suggests that top reviewers’ reviews are actually reliable or consistent with others’ perceptions, whereas most reviews by reviewers in the 80% ~ 99.5% communities deviate greatly from the common consensus. This deviance reaches as high as 0.8 deviation for the 99.5% community.
Extreme rating ratio (EXT): measures whether a user’s rating is extremely high or low, as spammers are likely to post extreme ratings while non-spammers post more moderate product-specific ratings. Since ratings range from 1 to 5 on Amazon, we consider 1 and 5 as extreme rating, following [82, 83], and measured the percentage of a user’s reviews having extreme ratings. Fig. 6.20 shows the average of EXT of the three groups of reviewers in each category. We observe that the EXTs of reviewers in 80% ~ 99.5% communities are relatively high ranging from 0.5 to 0.8, whereas that of reviewers in 0% ~ 50% communities, total reviewers, and top reviewers ranges from 0.4 to 0.6.

Review burstiness (BST): measures the interval between a user’s first and last reviews, as spammers often post reviews in a short period of time. Mukherjee et al. compared each
reviewer’s history with an estimated threshold of 28 days [82, 83]. The shorter the interval, the larger the burstiness. Burstiness was 0 if a reviewer has a history equal to or longer than 28 days. Fig. 6.21 shows the average BSTs of the three groups of reviewers. Note that top reviewers are expected to be valued customers who have a relatively long history with high-quality reviews. Indeed, top reviewers have the lowest BSTs (close to zero) as shown in Fig. 6.21. By contrast, we observe that reviewers in the 80% and 99.5% communities have rather high BST scores. Recall that both the top reviewers and the discovered reviewers in the 80% and 99.5% communities have high PVRs, but the BST score analysis suggests that the latter are likely to be spammers since do not have a long history but collect many positive comments in a short period of time to appear to be very “helpful.”

Figure 6.19: Deviated rating ratio (DEV) in each category
Other than nine spam indicators proposed by Mukherjee et al. [82, 83], we further report a measure, so called **Interaction burstiness (IBST)**. IBST measures an interval between first and last interactions of two users in each community. Note that spammers we are interested in are trying to boost their colluders’ influence. To do so, bursted interactions will occur mainly for the colluders’ reviews. We thus measure IBST between two users in discovered communities. Although we could not estimate the threshold, we used the same threshold (28 days) to compute IBST; because the intuition behind IBST is the same as BST. Since no specific relationships are defined for top reviewers and total reviewers, we computed the average IBST of each reviewer for them. Fig. 6.22 shows the average IBSTs of the three groups of reviewers. We observe IBSTs of reviewers in 80% ~ 99.5% communities are extremely different from IBSTs of total reviewers,
Figure 6.21: Review Burstiness (BST) in each category

top reviewers, and reviewers in 0% ∼ 50% communities as shown in Fig. 6.22. Specifically, we find that reviewers in 80% ∼ 99.5% communities had a massive amount of positive interactions with each other in a short period of time from Fig. 6.22.

Summary In short, our findings from spammicity analysis can be summarized as follows. First, we find a clear distinction in terms of each spammicity value between reviewers in the 80% ∼ 99.5% communities and reviewers in the 0% ∼ 60% communities. Concretely, the behavior of the former groups tends to exhibit strong spamming behavior (high spammicity) although their positive vote ratio is high. The behavior of the latter groups by contrast tend to be similar to that of the total reviewers and that of top reviewers (low spammicity). This result suggests that there exist reviewers whose reviews are maliciously endorsed to make more them influential.
Indeed, prior researchers have argued that votes from users are not reliable and easy to abuse [65, 82].

Second, we see that the spammicity increases, as the strength increases for reviewers in the $80\% \sim 99.5\%$ communities. In other words, reviewers in the higher strength communities (e.g., 99.5%) have a higher probability of being spammers; whereas reviewers in $0\% \sim 60\%$ communities tend to have low spammicity in general, although the spammicity values vary.
6.4.4.2 Spammicity Analysis of Reviewers Grouped by the Number of Reviews

In Section 6.4.4.1, we mentioned discovered reviewers tended to be users who reviewed relatively many items compared to top reviewers and total reviewers. To determine whether the spammicity differences of discovered reviewers reported in Section 6.4.4.1 were due to the number of reviews they submitted, we grouped the reviewers by the number of reviews each of them submitted. In this section, we report nine spammicity values of five groups of reviewers - reviewers with 100-200, 200-300, 300-400, 400-500, and 500-600 reviews, since the number of reviews of discovered reviewers ranges from 100 to 600. By doing so, we will show reviewers who have the corresponding number of reviews to each strength of communities do not show specific behavior patterns like discovered reviewers. In fact, we will show that those five groups of reviewers behave similar to either top reviewers or total reviewers.
In the following we will present the results in a series of plots where: the x-axis demonstrates the number of reviews they each submitted; the y-axis presents the strength of the appropriate measure; and we have one point per reviewer group.

**Content similarity (CS):** Fig. 6.23 presents the average CSs of the five groups of reviewers. As shown in Fig. 6.23, we do not observe a specific correlation between CS and the number of reviews. Also, the CSs of the five groups of reviewers range from 0.3 to 0.5, which is similar to the CS of total population (0.4).

**Rating abused item ratio (RA):** Fig. 6.24 presents the average RAs of the five groups of reviewers. As mentioned before, non-spammers are not likely to involve in rating abuse in general. As shown in Fig. 6.24, we observe the RAs of the five groups of reviewers are close to zero, which is similar to that of top reviewers and that of total reviewers. It is also interesting to see, while the difference between the maximum and the minimum values is not significant,
there exists a negative correlation between the number of reviews and RAs in Across and Book categories as shown in Fig. 6.24(a) and Fig. 6.24(c). One possible reason might be that those active users who reviewed more are often more likely to be reliable similar to top reviewers.

**Review duplicated items ratio (DUP):** Fig. 6.25 presents the average DUPs of the five groups of reviewers. As shown in Fig. 6.25, DUPs of the five groups of reviewers are close to zero regardless of the number of reviews they submitted.

**Maximum one day review ratio (MOR):** Fig. 6.26 shows the average MORs of the five groups of reviewers. It is also interesting to see that MORs of the five groups of reviewers are relatively high, compared to that of total reviewers. This result may suggest that users who submitted a lot of reviews in a system are actually active users who submit many reviews in one day as well in general.

**First review ratio (FRR):** Fig. 6.27 presents the average FRRs of the five groups of
reviewers. As shown in Fig. 6.27, the FRRs of the five groups of reviewers are similar to that of total reviewers (0.3).

**Early time frame ratio (ETF):** Fig. 6.28 shows the average ETFs of the five groups of reviewers. Interestingly, the ETFs (nearly 0.5) of the five groups of reviewers are relatively high compared to that of total reviewers (nearly 0.4), while their ETFs are similar to that of top reviewers. This result may indicate that those more active reviewers often tend to review an item earlier than others.

**Deviated rating ratio (DEV):** Fig. 6.29 shows the average DEVs of the five groups of reviewers. It is interesting to see a negative correlation between the number of reviews and DEV in Across category as shown in Fig. 6.29(a). Also, the DEV of a group of reviewers who submitted 100-200 reviews is the highest in Across and Book categories as shown in Fig. 6.29(a) and Fig. 6.29(c). This result may indicate that more active users who submitted more reviews
show more similar rating behavior to top reviewers. In other words, opinions of those active users are often likely to be consistent with others’ perceptions.

**Extreme rating ratio (EXT):** Fig. 6.30 shows the average of EXTs of the five groups of reviewers in each category. As shown in Fig. 6.30, EXTs of the five groups of reviewers are similar to that of total reviewers, regardless of the number of reviews they submitted.

**Review burstiness (BST):** Fig. 6.31 shows the average BSTs of the five groups of reviewers. In general, active non-spammers do not *burstly* review items, but they are more likely to have a relatively long history. Indeed, the BSTs of the five groups of reviewers are zero as shown in Fig. 6.31; whereas the BST of total reviewers is higher than zero.

**Summary** In short, our findings from spammicity analysis of reviewers grouped by the number of reviews can be summarized as follows. First, in general, we could not find specific patterns depending on the number of reviews they submitted. Also, the spammicity values of
the five reviewer groups were similar to those of total reviewers (low spammicity). This lends support to our claim that distinguishing characteristics of discovered reviewers presented in Section 6.4.4.1 were not simply due to the reason that the discovered reviewers reviewed more.

Second, users who reviewed more tended to show similar behavior to top reviewers in some features such as ETF, MOR, FRR, and DEV. This also agrees with the common sense that users who submitted many reviews are actually active users who review more and earlier. And opinions of such more active reviewers may be considered as reliable.

6.4.4.3 Analysis on Spammer Classification

In Section 6.4.4.1 and Section 6.4.4.2, we have discussed discovered reviewers have distinctive characteristics in terms of spammicity values. In this section we show the correlation between

Figure 6.28: Early time frame ratio (ETF) in each category
the strength of each community and the probability of being spammers. Our goal is to suggest a way to incorporate distinctive characteristics of different strengths of communities for spammer detection, and to show the effectiveness of our community-based scheme.

The most direct way to evaluate our approach is to compare our detection process to the state of the art content-based classifier with Amazon ground-truth dataset. However, after several attempts, we were unable to obtain access to Amazon datasets with ground-truth labels used in previous research such as [82, 83]. Therefore we opted to generate a “pseudo ground truth set” by applying Mukherjee et al.’s approach to our dataset, as it is the state-of-art classifier shown to have high accuracy over Amazon dataset with ground truth [82]. In particular, following [82, 83], we assume that when reviewers are ranked in descending order based on the sum of nine spamminicinity values, top and bottom 5% ranked reviewers can be classified as spammers and non-spammers respectively in an unsupervised setting. We then show that our
community-based approach reliably identifies opinion spammers even when pure content-based classifiers fail, and achieves the same level of accuracy as the state-of-art pure content-based classifier.

To measure the accuracy, we plot ROC curve by varying different strengths as thresholds to define spammers. Fig. 6.32 shows the ROC curve where the x-axis represents the false positive rate and the y-axis represents true positive rate. Each point represents true positive rate against false positive rate given $\tau$% strength as a threshold. Similar to the analysis in Section 6.4.3, we assume that reviewers in communities with strengths greater than or equal to $\tau$ are spammers; those in communities with strengths less than $\tau$ are non-spammers.

When $80\% \sim 99.5\%$ are used as thresholds, we observed there were no false positives, as shown in Fig. 6.32. This means that all the reviewers in $80\% \sim 99.5\%$ communities appeared in the top 5% of ranked reviewers (i.e., spammers); which is expected, as their spamminess values
were high on average as discussed in Section 6.4.4.1. Note that the larger threshold \( \tau \) is, the lower true positive rate is. For example, when 99.5% is used as threshold, true positive rate is 0.2 in Across dataset. This is because there were many false negative results including reviewers in 80\% \sim 98\%.

On the other hand, when we use 60\% or less strengths as a threshold, the false positive rate dramatically increased (over 0.7), meaning that 0\% \sim 60\% communities are likely to be non-spammers. The number of false positive results thus increased, as more reviewers in 0\% \sim 60\% communities are classified into spammers by using 60\% or less strengths as a threshold. In such a case, the number of false negative results would be small, resulting in higher true positive rate.

Note that we get the best result (i.e., 0\% false positive rate and high (close to 100\%) true positive rate), when 80\% is used as a threshold; and the classifying results get worse with
thresholds lower than 80%. This implies a clear distinction between reviewers in 80% ~ 99.5% communities and those in 0% ~ 60% communities. This lends support to our claim that such distinctive characteristics of different strengths of communities can be used to distinguish spam communities from non-spam communities.

**Summary** In short, our findings from ROC analysis can be summarized as follows. First, ROC analysis suggests that while strongly positively connected communities may be naturally constructed with different strengths, communities with a strength higher than 80% are strong spammer candidates.

Second, we have shown that there exists a great correlation between the strength of communities and content spammicity through ROC analysis. In fact, we have shown that we could achieve 0 % false positive rate and nearly 100 % true positive rates using 80% as a threshold.
However, by no means do we claim that our classification using 80% as threshold is almost perfect with 0% false positive rate and nearly 100% true positive rates; as the accuracy was measured by comparing to the pseudo ground truth set. The correctness of accuracy measurement in our ROC analysis may thus depend on the accuracy of the pseudo ground truth set. However, we showed that the strength can be used an indicator to distinguish spam communities from non-spam communities, while achieving the similar level of accuracy to the state-of-art content based classifier.

We note that it is hard to evade our community-based scheme as spammers essentially need to build such high strength of communities to make their opinions influential; whereas spammers can easily fake their content features (e.g., reword their contents to lower content similarity value) to evade detection by content-based classifiers.

It is also important to note that discovered communities not only include reviewers but also commenters who may not write any spam reviews. Existing pure content-based approaches will not be able to discover such supporting commenters, though they are also suspicious and indirectly contribute to opinion spams. In other words, our approach can discover both spam reviewers and suspicious commenters, which is a great advantage over pure content-based approaches.

6.4.4.4 The Effect of Sentiment Analysis on Spammer Classification

Recall that non-spammers may form natural communities because of their genuine common interest on items as mentioned in Section 6.1. To show that only high strengths of positive communities, not high strengths of neutral or negative communities, are spammers, we present ROC curve analysis without sentiment analysis in this section. In particular, we ignored sentiments of relationships and found strongly connected communities in each strength of general user relationship graphs.

Fig. 6.33 shows ROC curve without sentiment analysis. For this analysis, we vary strengths of strongly connected communities as thresholds to define spammers. The x-axis represents the false positive rate and the y-axis represents true positive rate. Each point represents true positive rate against false positive rate given \( \tau \) strength as a threshold.

Similar to the analysis in Section 6.4.3.2 and Section 6.4.4.3, we assume that reviewers in communities with strengths greater than or equal to \( \tau \) are spammers; those in communities with strengths less than \( \tau \) are non-spammers.

As shown in Fig. 6.33, we observe that all points are yielded in the bottom right corner. Unlike the results in Fig. 6.32 where we observed 0% false positive rate with 80% as a threshold, Fig. 6.33 suggests that false positive rates are close to 50% with 80% as a threshold; unlike the results in Fig. 6.32 where we observed high (close to 100%) true positive rate with 80% as
a threshold, Fig. 6.33 suggests that true positive rates range from 10 % to 20 % with 80% as a threshold. This indicates that reviewers in strongly connected communities whose strengths are greater than 60% are not necessarily to be spammers unlike those who appear in the same strengths of strongly positively connected communities. This agrees with our assumption that spammers are strongly correlated with each other through positive interactions, and both analysis including the sentiment analysis of relationships and the strength analysis of relationships is needed to define spammers.
### Table 6.1: Spearman’s rank correlation coefficient in each category

<table>
<thead>
<tr>
<th>Category</th>
<th>Spearman’s ρ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Across</td>
<td>-0.00541366846337238</td>
</tr>
<tr>
<td>Movie</td>
<td>-0.29488221271928</td>
</tr>
<tr>
<td>Books</td>
<td>-0.00413571909952635</td>
</tr>
</tbody>
</table>

6.4.5 A Comparison of Linguistic Model-based and Spam Indicators-based Classifiers

As we discussed in Section 6.4.3 and Section 6.4.4, we have observed inconsistency in two classification results (i.e., results from a linguistic model-based classifier and that from a spam indicators-based classifier). While we have mentioned one possible reason (i.e., the linguistic model with TripAdvisor ground truth set can only be applied to travel-related or even only TripAdvisor reviews), in this section we further investigate how different two classification results are. To this end, we apply Spearman’s rank correlation coefficient.

Spearman’s rank correlation coefficient $\rho$ is a measure of testing statistical dependence between two ranked lists [93], which is defined as follows.

$$\rho = 1 - \frac{6 \sum_{i=1}^{n} d_i^2}{n(n^2 - 1)},$$

where $d_i$ is the difference between ranks of reviewer $i$ and $n$ is the number of reviewers.

The sign of the Spearman correlation indicates the direction of association between two ranked lists $R_1$ and $R_2$. If $R_2$ increases when $R_1$ increases, the coefficient $\rho$ is positive. If $R_2$ decreases when $R_1$ increases, $\rho$ is negative. As $\rho$ becomes closer to zero, there are less correlation between $R_1$ and $R_2$, and $\rho$ of zero means that there is no correlation between $R_1$ and $R_2$. Table 6.1 summarizes the Spearman’s rank correlation coefficient between ranks in Section 6.4.3.2 and ranks Section 6.4.4.2 in each category.

As shown in Table 6.1, $\rho$ in each category is close to zero, meaning that there are no correlation between two ranked lists (one from Section 6.4.3.2 and another from Section 6.4.4.2).

6.4.6 Analysis on Negative Relationships of Discovered Reviewers

We now explore characteristics of negative relationships of discovered reviewers. Note that we want to find the competitors of spammers. As we have observed communities with a strength higher than 80% are strong spammer candidates, we investigated negative relationships of only those users in communities with a strength higher than or equal to 80%. Unfortunately, we do not find many negative relationships of those in our dataset as shown in Fig. 6.34. One possible...
Figure 6.34: The number of negative relationships of spam reviewers in each category

(a) The number of negative relationships in Across
(b) The number of negative relationships in Movie

(c) The number of negative relationships in Books

conclusion might be that the primary goal of spammers is rather promoting their own reviews than demoting competitors’.

6.5 Summary

In this work we proposed a novel approach to find opinion spammer groups by analyzing user interactions coupled with sentiment analysis rather than analyzing the review content. This methodology is based on the intuition that spammers need to form artificial communities in order to make their opinions influential, we identified spam communities by exploiting community structures built through abnormally non-random positive interactions. We thereby exposed two
types of spammers: spam reviewers who post spam reviews and supporting commenters who extensively endorse those reviews. Through extensive experimental analysis, we demonstrated the effectiveness of our community-based approach in terms of accuracy and reliability. We showed that our approach can successfully identify without relying on specific review contents, while achieving the same level of accuracy as the state-of-art pure content-based classifier.
Chapter 7

Concluding Remarks

7.1 Summary

In recent years we have witnessed the rapid growth of e-commerce markets. In this thesis, we address three major concerns in e-commerce markets including guaranteeing the quality of information, building a trusted manager resilient to attacks, and detecting attackers.

First of all, our first work aimed to better understand users’ true interest to help users make efficient choices given an extensive range of unfiltered information resources. Specifically, in the first work, we have investigated on whether the fact that users reply to each others reviews could help us discover meaningful relationships relevant to item recommendation in e-commerce markets. We first showed through a global measure that different degrees of relationships do exist in different item communities. We then designed a probabilistic mechanism to distinguish meaningful relationships from random interactions. Based on the mechanism, user communities with strong relationships were discovered. To demonstrate the relevance and usefulness of the discovered communities, we applied existing social recommendation algorithms over them, and showed that they can help improve significantly the quality of item recommendation when compared with collaborative filtering approaches that only consider user-item relationships.

Increasingly sophisticated groups continue to create tremendous amount of emerging threats as a new technological paradigm evolves. Accordingly, there is a growing need of building security solutions to make a system resilient to attacks. Although many solutions including reputation systems have been proposed in the literature, it is often difficult to pick the right solution for a given system. To help each system administrator pick a right reputation system for their own purpose, in our second work, we proposed a highly configurable adaptive framework that models the attackers’ realistic behavior and that evaluates the true resilience of each system considering adaptive nature of attackers. We proposed a model, called an attack tree, to capture attackers’ adaptive behavior and find their optimal strategy for different reputation systems.
With a case study of three reputation systems, we showed that the proposed framework was thereby able to provide a fair evaluation with the consideration of each system’s worst case scenario.

With the growing influence of user-generated opinions on a wide range of daily decisions (e.g., purchasing products), adversaries started to deliberately inject fraudulent opinions into various online communication media (e.g., social networks, message boards, and review systems). In our third work, we thus proposed a detection scheme that identifies spammer groups in one of e-commerce platforms, a review system. We first introduced a new angle of opinion spamming behavior, what we call *promotional opinion spammers*, referring to attackers who try to improve the influence of their opinions by artificial boosting behavior. We explored community structure built through users’ replying activities coupled with analyzing sentiments of user relationships. Then, we showed that our community-based approach achieves both accuracy and reliability. In other words, the proposed approach is reliable against content manipulation, while achieving a comparable level of accuracy to traditional content-based approaches.

7.2 Future Work

I look forward to pursuing challenging problems across the board, particularly those involving security and data analytic issues. In the following, I discuss possible extension of current work as well as long-term research direction.

7.2.1 Extension of Recommender System Research

Several possible extensions of my first research include (1) expanding it into different domains, (2) finding other forms of implicit communities for recommendation purpose, and (3) improving recommendation further with sentiment analysis, and (4) building real-time solutions. First of all, my research is specifically done in the context of item recommendation in Amazon without explicit social networks. We may encounter research problems when applying the proposed method to other domains. For example, explicit social networks, if available, may be more helpful in certain contexts of recommendation. Whereas implicit communities I found is helpful to identify shared interest on items to purchase, it may not be helpful for friends recommendation. To investigate such cases, one immediate next step may include investigating on difference between recommendation with explicit social networks from that with implicit communities. In certain applications including Urbanspoon, there may not exist relationships built through replying actions, I would love to explore other forms of implicit communities that may help recommendations.

Second, as we worked on users’ replies, the sentiments of such replies may help further to
recommend items. For instance, if certain users interact with each other because they have shared interest on item categories yet have totally opposite opinions about certain items. In such cases, it would be helpful to recommend “you may like” or “you may not like” items to analyze sentiments of their interactions,

Finally, my research is based on the assumption that we have sufficient amount of interaction history. However, it remains challenging to offer a comprehensive real-time solution to recommend certain categories of items such as events that often require finishing analysis in a timely manner. I believe it would be interesting research direction to discover patterns given limited amount of time and resources.

7.2.2 Finding Optimal Attack Strategies in Complex Systems

The most important yet difficult part of research on resilience evaluation is how accurately we can reason about attackers’ realistic behavior, and estimate the optimal attack strategy specific for each reputation system. In my second work I proposed a framework exploring possible future states to estimate the optimal attack strategy. Ideally, if we can explore all possible states, we can get the most accurate estimate of an attacker’s optimal strategy. In practice, it would be computationally infeasible to explore the whole space of future states. To balance between computational costs and better estimation, I thus limited the number of future states to explore in my research. However, as systems are becoming increasingly complex, excogitating how to efficiently derive the optimal strategy remains an interesting avenue for future research.

Moreover, although my research on designing a resilience evaluation framework was done in the context of reputation systems in an e-commerce environment, I anticipate the idea of the proposed framework can be applied in diverse applications such as sensor networks and real-time communication systems, as it is a highly configurable and domain-independent framework. In fact, better estimation of the optimal attack strategy (i.e., system vulnerabilities analysis) in any applications is the core part of any security solutions to guarantee resilience to manipulations.

7.2.3 Studying Other Types of Spamming Activities

My third research opened up a new research direction in opinion spams (i.e., analysis of sentiments and structures of user relationships). As my research targeted spammers who build strongly positively connected communities to make their own reviews influential, many other forms of spamming activities may be possible through other structures. For example, individual non-group spammers who employed by the company but do not know each other, may not build communities through direct interactions. One of my next research steps will be thus exploration of different spam community structures. We may also combine our approach with content-based classifiers (e.g., [82, 125]) to detect such non-group spammers.
Moreover, while we have discussed the effectiveness of our approach in terms of detection accuracy, it would also be useful to develop a model to measure the effect of various spamming strategies (e.g., manipulate contents and build artificial communities). I thereby plan to investigate the robustness of our approach (i.e., to what degree attackers can manipulate their behavior to avoid detection).

7.2.4 Studying Dynamics of Implicit Communities

As discussed above, my research on community discovery was done with static historical data. However, it is also known that people’s opinions evolve over time. When analyzing their typical or suspicious behavior for both of my first and third research, recent interaction information in a short time window might be more meaningful than old historical data. For example, one may change her interest on items over time (e.g., from music players to phones). Then, interactions about music players may not be helpful to recommend phones. Also, one may be compromised by spammers at some timepoint, while she was a normal customer once. In such a case, we can analyze temporal information of her behavior to find out which point she was compromised. I thus believe analyzing temporal and long-term history would be an interesting avenue for my future research.

7.2.5 Analyzing Economic Effects of Opinion Spams

Another interesting research direction is to investigate economic effects of spams from the viewpoint of attackers, forum industries such as Amazon, and normal users. By doing so, I expect to answer various questions. (1) What is the real benefit of spamming? Is it economically worth for attacker to consume their resources for spams? By understanding their actual need for spamming, we may be able to build a more generalized and realistic model to detect spam communities. (2) From a forum industrial point of view, is it critically needed to build detection techniques on their systems? How much do normal users care about the quality of opinions in the forum? Does the quality have influence on the profit of forum industries and purchasing decisions of normal users?
REFERENCES


[19] Bimal Viswanath, M Ahmad Bashir, Mark Crovella, Saikat Guha, Krishna P Gummadi, Balachander Krishnamurthy, and Alan Mislove. Towards detecting anomalous user be-


