Trust in the Social Web

Applications in Recommender Systems and Online Auctions

John O'Donovan, BSc, MSc

A thesis submitted to University College Dublin for
the degree of Doctor of Philosophy

April 2008

School of Computer Science and Informatics,
University College Dublin
Belfield,Dublin 4.

Head of Department: Joe Carthy

Supervisor: Professor Barry Smyth

Abstract

The portion of daily business and socialising conducted on the Web is increasing at a rapid rate. In February 2006, online auction giant eBay inc. reported annual growth rate to be in the region of 29% [27] with net revenues of $1.7 billion for the last quarter of 2006 alone. eBay has over 200 million registered users and receives over a billion page views per day [27]. In 2006, the online store and recommender system Amazon.com was ranked 272 on the Fortune 500, with a revenue of 8,490 million dollars [1]. From July 2006 to July 2007 the social networking site Facebook grew from 7.5 million users to 30 million. [25]

Despite these huge figures, the technology behind Web applications is still very much in its infancy. Every day, people are forced to make important trust decisions about strangers on the Web with only a limited amount of information by which to assess and evaluate the person they are transacting with. The rapid growth of "Social" Web applications, in which people can perform transactions, author content, or otherwise contribute, has brought about a need for new and better ways to model trust and reputation on the Web.

This thesis explores issues of trust on the Social Web, focusing specifically on the domains of recommender systems such as Amazon.com and online auctions such as eBay. This work proposes novel techniques for assessing trust and reputation in recommender systems and online auctions, drawing on the wealth of historical data from existing systems as well as new data as it arrives. Trust information can be used to benefit online systems in a variety of

ways, mainly as a tool for making the vast and impersonal online world more like the familiar "local-store" business culture or a "social scene" in which people can assess trust based on word of mouth and the opinions of friends. Experiments and evaluations performed during the course of this research show that trust information can be harnessed from historical data and used to increase predictive accuracy, robustness and overall user satisfaction in recommender systems and online auctions.

Acknowledgements

Thanks to my supervisor Barry Smyth for his excellent and patient guidance, without which this work would not have been possible. Thanks to my parents for everything they've done to get me this far, to my sister Anne and brothers Paul and Daniel. Thanks to everyone in the office at UCD, and to all my friends who made this time so memorable. I would like to give a special dedication to my good friend Paul Caffrey (RIP). We will not forget you. Thanks to professor Paddy Nixon for getting me connected at the University Southern California, and my collaborators, professor Dennis McLeod and Vesile Evrim. Thanks also to Brynjar Gretarsson and professor Tobias Höllerer at the University of California Santa Barbara. I would also like to gratefully acknowledge Science Foundation Ireland who funded this research. Most of all I want to thank my girlfriend Jenny for being so caring and for putting up with me while I was writing this work. Thank you!

List of Publications

- O'Donovan, John. Gretarsson, Brynjar. Smyth, Barry and Höllerer, Tobias. "PeerChooser: Visual Interactive Recommendation" *CHI 2008, The ACM International Conference on Computers and Human Interaction.*, Florence, Italy, April 2008. ACM Press. (accepted)

- O'Donovan, John. Evrim, Vesile. McLeod, Dennis, and Smyth, Barry "Extracting and Visualising Trust Relations from Online Auction Feedback Comments." *In proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI) Hyderabad, India 2007.* ACM Press.

- O'Donovan, John. Evrim, Vesile. McLeod, Dennis, Smyth, Barry and Nixon, Paddy "Personalising Trust in Online Auctions" *In proceedings of the Starting Artificial Intelligence Research Symposium (STAIRS)*, Riva Del Garda, Italy 2006. ACM Press.

- O'Donovan, John and Smyth, Barry "Trust No-One: Evaluating Trust-Based Filtering for Recommenders" *In proceedings of the Nineteenth International Joint Conference on Artificial Intelligence Edinburgh, Scotland 2005.* ACM Press.

- O'Donovan, John and Smyth, Barry "Is Trust Robust?" *In proceedings of the international conference on Intelligent User Interfaces (IUI)*, Sydney, Australia 2006. ACM Press.

- O'Donovan, John and Smyth, Barry "Mining Trust from Recommendation Error" *International Journal of Artificial Intelligence Tools*, 2006. ACM Press.

- O'Donovan, John and Smyth, Barry "Eliciting Trust Values from Recommendation Errors" *In proceedings of the Florida Artificial Intelligence Researchers Society Conference*, Clearwater, Florida, 2005.

- O'Donovan, John and Smyth, Barry "Trust in Recommender Systems" *In proceedings of the international conference on Intelligent User Interfaces (IUI)*, San Diego, 2005. ACM Press.

Book Chapter

- O'Donovan, John and Smyth, Barry "Using Trust in Social Web Applications" *In Computing with Social Trust.* Edited by Jennifer Golbeck, Springer-Verlag 2008. (in press)

Other Relevant Publications

- O'Donovan, John and Dunnion, John (2003) "A Comparison of Collaborative Recommendation Algorithms over Diverse Data Types" *14th Irish Conference on Artificial Intelligence and Cognitive Science*, Dublin 2003.

- O'Donovan, John 2005 "AdRec: An Adaptive Recommender System" *University College Dublin, 2005*, Masters Thesis, University Press

- O'Donovan, John and Dunnion, John (2004) "A Framework for the Evaluation of Collaborative Recommendation Algorithms in an Adaptive Recommender System" *In Proceedings of the International Conference on Computational Linguistics (CICLing-04), Seoul, Korea.*

- O'Donovan, John and Dunnion, John (2004) "Adaptive Recommendation: Putting The Best Foot Forward" *In Proceedings of 3rd International Symposium on Information and Communication Technologies (ISICT)*, Las Vegas, Nevada. USA. June 16-18 2004.

- O'Donovan, John and Dunnion, John (2004) "Evaluating Information Filtering Techniques in an Adaptive Recommender System" *In Proceedings of 3rd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, Eindhoven, The Netherlands. August 23-26 2004.

- O'Donovan, John and Dunnion, John (2004) Mheitifhoglaim do mholadh dhearadh innill. *In Sruth Gaeilge sa 15u Comhdhail ar an Intelacht Shaorga agus ar na hEolaiochtai Cognaiochta (AICS 04)* Caislean an Bharraigh, Contae Mhaigh Eo, Eireann., pages 18-20, September 9-12 2004.

Table of Contents

Chapter 1

Introduction

This thesis presents the results of research into trust and the Social Web, using applications in recommender systems and online auctions. These applications act as a test bed for new ideas on how to harness trust and reputation in the Social Web. Long before the World Wide Web, in the 1960s, Harvard professor Stanley Milgram proposed his Small World Experiments [77]. These consisted of a range of experiments designed to test Kuchen and Pool's Small World Problem [96]: given a set N of people, what is the probability that each member of N is connected to another member via $k_1, k_2, k_3 ... k_n$ links?

Milgram's test examined the average path length for social networks of people in the United States by using parcel forwarding between source and target individuals via the US postal service. Milgram's experiments revealed that our society is a small world network with shorter-than-expected path lengths, so the probability of two random people knowing each other is more than the expected value. This concept has become more commonly known as the "six degrees of separation" problem. There have been many protagonists and critics to Milgram's ideas since the 1960s, for example Malcolm Gladwell's popular book "The Tipping Point" [35] proposes that connectivity in human social networks is largely dependent on a few extraordinary people, whom he calls "connectors". (Gladwell's research was based on findings from a range of articles originally published in The New Yorker.)

Of late, Social web applications such as Wikis and social networking ap-

plications like MySpace and Facebook [64] are becoming hugely popular, making experiments such as Milgram's not only more feasible to perform, but also more relevant, since social connectedness can be harnessed and used to enhance the quality of a user's experience in the online world. One of today's foremost thinkers on the topic of network analysis is Alberto Barabási. In his recent book "Linked" [6], Barabási describes structural similarities between human social networks, Web topology, hierarchies of species in nature, and a range of other scale free networks. Barabási hails the analysis of such networks as the "true science of the future" [11]. This thesis exploits the structures within another scale free network, the Social Web, with a view to harnessing trust and reputation relationships that can be inferred.

The issue of trust on the Social Web has attracted a large amount of research attention over the past number of years. Traditionally, online information was authored by site owners or webmasters. Rather than thinking of the web in its original form, as a collection of interlinked web sites containing static content, we now consider it as a framework wherein communities of users provide content which is constantly evolving. Traditional web sites are being replaced by Social Web "applications" which can be thought of as platforms for user-generated content (UGC) as opposed to the traditional webmaster-authored sites. The shift in information flow on the web has brought with it pressing demand for better ways to model trust in the online world. In 2004, O'Reilly media coined the (somewhat controversial) phrase Web 2.0, in a sense redefining the Web as second generation of Web-based services such as social networking sites, Wikis and communication tools. The core difference being that these applications place emphasis not on Web pages, but on collaboration and sharing among *users* of the pages. This has developed the Web from its original use as an *information source* into its current use as a *participation platform*.

Reputation has always played a key role on the Web, but in the past it has been the reputation of pages that has been important, and all of a sudden the reputation of participating users is playing an important role. Traditionally, the importance or relevance of a web page was computed by a now famous link analysis algorithm, *PageRank*, developed by Larry Page and Sergey Brin at Stanford university in 1995. [93]. This research led to

the development of Google inc. in 1998. A PageRank results from a "ballot" among all the other pages on the World Wide Web about how important a particular page is. A hyperlink to a page counts as a vote of support. The PageRank of a page is defined recursively and depends on the number and PageRank metric of all pages that link to it ("incoming links"). A page that is linked to by many pages with high PageRank receives a high rank itself. If there are no links to a web page there is no support for that page and accordingly, its rank is nil.

In the Social Web, users are considered the important structure: the reliability of the user providing the information is as important as the information they provide. The notion of "trust" for users of the Social Web can be viewed analogously to a PageRank in Web search. In a similar manner to each incoming link being considered as a "vote" in the PageRank algorithm, a trust-based algorithm must somehow compute an overall trust score for a user based on some combination of the individual "units" of trust that the application has as input. Using eBay as an example, a trust algorithm might use the explicit votes given by both parties in a transaction as the units of trust. The current eBay implementation simply takes the average of these values as the overall trust score. Figure 1.1 shows how eBay also makes use of reputation information by exposing past reviews of both buyers and sellers to potential future buyers and sellers.

In this thesis we examine ways to gather individual units of trust in the Social Web and examine techniques to combine these to arrive at overall trust values for users of the Social Web. This work also addresses the issues involved in presentation of trust information to users on the Social Web, from the premise that trust information must be delivered at the right time, with minimal interference for the user.

Before continuing with the discussion of the role of trust in the Social Web, we must examine some of the varying notions of trust and reputation, as well as looking at the Social Web itself. The following sections examine both trust and the Social Web individually.

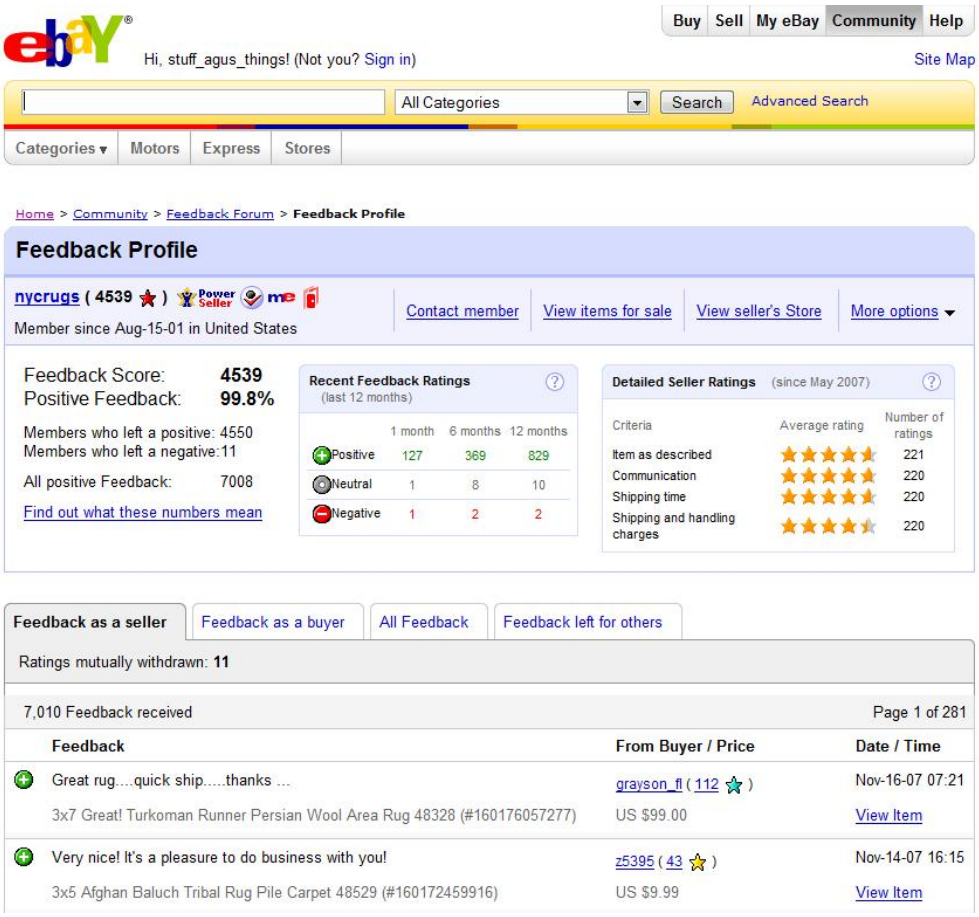

Figure 1.1: Screenshot of the Textual Reviews and Feedback Scores on eBay.

1.1 The Social Web

In everyday society, people are linked in many different ways, through friends, family, co-workers and anyone else that they interact with. The well known trivia game "Six Degrees of Kevin Bacon" is based on a variation of the concept of the small world phenomenon [96] [77] [6] which states that any actor can be linked, through their film roles, to Kevin Bacon. The Social Web brings these social relations from the real to the online world. Just as in the real world, reputation and trust are important factors in the Social Web.

The term "Social Web" was first used in its current context by Hoschka

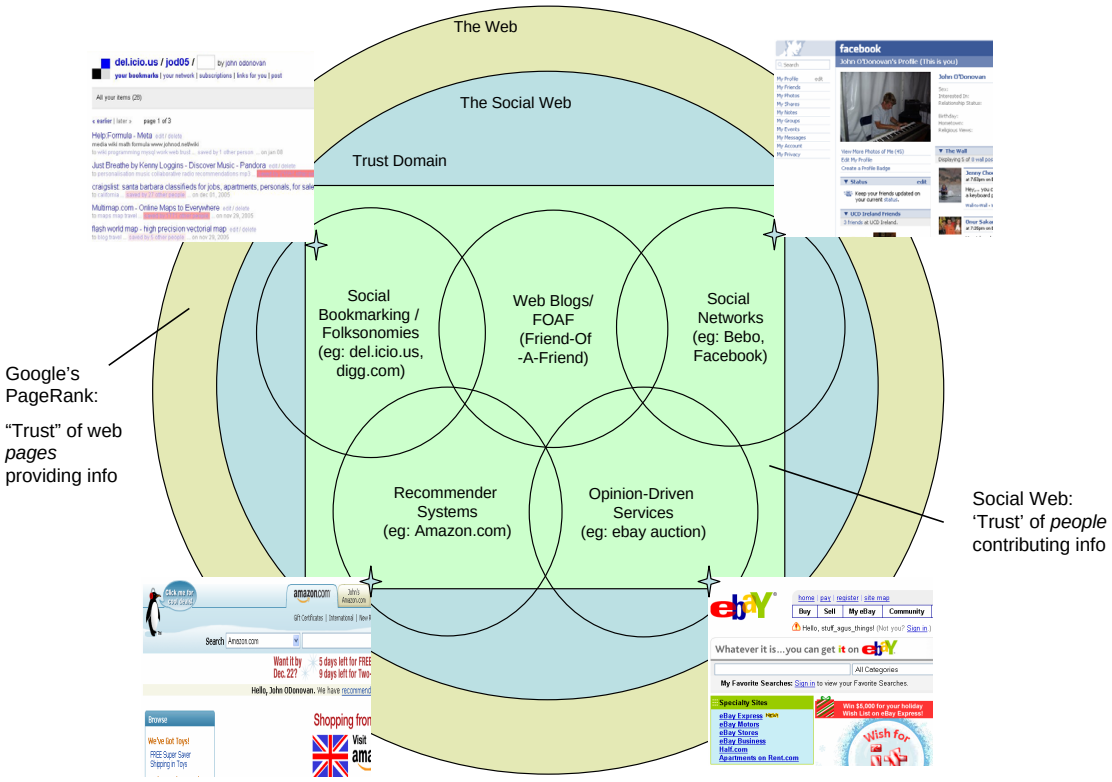

Figure 1.2: Trust Domain Within the Social Web.

in [48] to describe the change from using computers simply as cooperative business and entertainment tools to using them as a fully interactive social channels. Nowadays the Social Web can be considered as an open, global, distributed, data sharing network similar to today's World Wide Web, except instead of linking documents, the Social Web links people, groups, companies and even concepts.

The Social Web as we know it was first conceived in 2004 by Drummond Reed et al. in [101], Reed's paper is based somewhat on the work in [53] carried out in 2003 by Jordan et al on augmented social networks. Reed proposes how a new protocol for distributed mediated data sharing and synchronisation, XDI, could enable a new layer of trusted data interchange applications. The key building blocks for this layer are I-names and I-numbers

(based on the OASIS XRI specifications), Dataweb pages, and link contracts. These specifications define a generalised, extensible service for sharing, linking, and synchronising data over the Internet and other data networks using XML documents and XRIs (Extensible Resource Identifiers). These XRI's are a potential mechanism for the cross-domain transport of trust information between users of Social Web applications, along with other semantic web standards such as FOAF, and online [24], they could even be viewed as our modern day equivalent to the postcards sent in Stanley Milgram's small world experiments [77]. The analogy used in Wikipidia for the Social Web is the worldwide banking and credit card system. This infrastructure has evolved over centuries to facilitate the global exchange of a very sensitive form of data (money) by establishing a common means of exchange among trusted third party service providers (banks). The Social Web takes the same approach for exchange of private, sensitive information by establishing a common means of exchange among trusted third party service providers (i-brokers).

The bottom circles in Figure 1.2 depicts the application area of this work within the context of the larger World Wide Web. Figure 1.2 also shows some of the application areas which rely heavily on the opinions and views of communities of users and some of the overlaps and interaction areas between these applications. In essence, we are saying that any application on the Social Web which relies on the opinions, contributions or actions of communities of users stands to benefit from analysis of the underlying trust relationships that exist in that community.

This thesis focuses on the issue of trust within the Social Web, specifically on how we can evaluate and harness trust for the good of its users. For example, we will explain how trust-aware recommender systems can generate better suggestions for users by eliminating the contributions of untrustworthy users. Trust is a difficult concept to define, and there are many, sometimes conflicting definitions from a range of disciplines. A detailed discussion of these cross-discipline concepts of trust is provided as an appendix to this work. For the work presented in this thesis, we are mainly interested in trust in a *computational* sense. In particular, trust is a function of reputation, which can be computed over historical data.

1.2 Motivating Examples

Figure 1.2 shows a broad overview of the Social Web and some of its representative applications. Recommender systems are shown as a key application within the Social Web. These systems rely on the opinions of a user's peers to generate a recommendation for that user. In everyday life we can easily see the advantages of seeking recommendations from friends whose opinions we have learned to trust over time rather than from friends who have not gained our trust for making recommendations. This concept holds true for trust on the Social Web, although the task of addressing the problem is vastly more difficult due to the reduced information available upon which we can make trust decisions. For example, in the real world information can be gleaned from body language, facial expression and other idiosyncracies as people communicate- this type of information is currently not accessible in most Social Web applications. Furthermore, human trust is a function of a complex array of inputs , which online systems cannot yet model [69]. Traditionally, recommender systems have leveraged the opinions of "similar" users when making a recommendation, where similarity is usually some simple function computed over limited data, for instance, a set of past purchases in an online bookstore such as Amazon.com. In the real world when people are seeking recommendations, they tend to rely more on *trust* than on similarity. This work proposes that the traditional approach to computing a recommendation can be improved by utilising trust information, instead of simply computing similarity as a proxy for the range of complex metrics used in real world recommendation processes.

A more sinister example of the importance of trust in the Social Web lies in the area of Social Networking applications such as Facebook* [64] which is hugely popular in the United States and Europe. On Facebook, personal information such as phone numbers, photographs and addresses for example, are available only to trusted users. Currently, Facebook seeks an explicit trust statement from the user before personal information is displayed to strangers, in the form of a "friend request", but trust is still propagated/inferred because users can see some information from friends of friends, for example,

*www.facebook.com

pictures and names. Even with explicit "friend requests", it is not guaranteed that everything is above board: a tragic event* occurred in November 2007 in the United States, a teenage girl was found to have committed suicide after she had struck up a relationship with a teenage boy whom she met on the social networking site MySpace. It developed that the boy never really existed and the profile had been maliciously constructed by a number of the girls classmates (who are currently facing legal charges). This tragedy may have been averted if a sufficient framework for modelling trust had been in place on the social networking site at the time of the event. Building a reliable model of trust for Social Web applications such as MySpace and Facebook is an issue of growing importance in western culture which is wrought with problems such as bullying, spamming, identity theft, phishing and unwanted admirers. It should be noted however, that even with a trust system in place, determined malicious users can still attempt to and succeed in subverting the system in some way. This is simple fact that also pervades disciplines such as cryptology and security.

Areas of the Social Web which stand to gain probably the most from trust information include recommendation systems, social networking sites and online auctions such as eBay. To give a sense of scale, eBay has over 4.8 billion feedback comments in the system and a revenue of 8,490 million dollars [1]. Within eBay, the largest online marketplace in the world, users are forced to make their business transactions based on a very limited amount of information about the person they are transacting with. eBay is one of the few online applications which already operates a trust-based system. Later in this thesis we examine some of the flaws in the current system and propose, implement and test a new trust system based on the wealth of information hidden in eBay's feedback comments. In an application of this scale, any increase in the trust a user can place in the system can only result in better business for all concerned.

*http://www.cnn.com/2007/US/11/17/internet.suicide.ap/

1.3 Objectives

The core objectives of this thesis are to consider the importance of trust in the context of the Social Web, to develop novel ways to harness trust information, compute new models of trust, and to present trust information in a manner that can benefit users of the Social Web. The following list provides a more detailed look at the initial objectives which were defined at the outset of this work:

1. Assessment of the importance of trust within Social Web applications.
2. Identification of domains on the Social Web in which trust could be modelled from historical/new data.
3. Identification of specific domains on the Social Web which could benefit from use of such trust models.
4. Development of new ways to visualise trust and explain its concept to the end-user.

To meet the above objectives, two Social Web application areas were chosen and a range of algorithms were developed to model, integrate and visualise trust information in various ways. The list below outlines the general contributions of this thesis to research in trust on the Social Web.

- technique and application for building and using trust models in recommender systems
- technique and application for building and visualising trust relationships based on online auction feedback comments.
- algorithms for computing recommendation ranges based on error history in recommender systems.
- identification of reinforcement problem in trust based systems and analysis of attack strategies for trust based recommender systems.
- technique and application for visualisation of and interaction with trust and similarity values during the recommendation process.

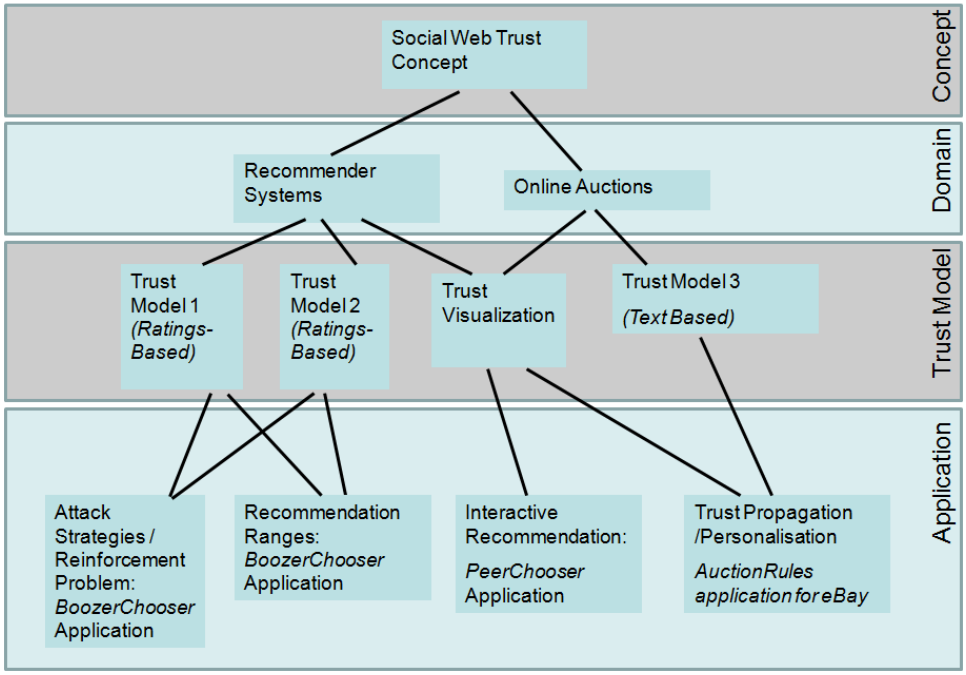

Figure 1.3: Graphical Overview of this Research.

- development of a visual *explanation interface* for recommender systems which are based on collaborative filtering.

1.4 Overview of Thesis

To gain a better understanding of the objectives defined above, we first performed a detailed analysis of current research involving the various facets of trust on the Social Web, using both recommender systems and online auctions as our application domains. In the analysis of related work, trust, recommender systems and online auctions are first examined individually and then from an integration perspective. Figure 1.3 provides a high-level graphical overview of the components in this research. Chapter 1 of this thesis introduces trust in the context of the Social Web, providing some motivating examples and outlining the core objectives of this thesis. Background research is presented in Chapter 2 on recommender systems, focussing on Automated Collaborative Filtering (ACF) and alternative approaches to the

filtering task. Chapter 3 analysis the state of the art of research into trust and reputation metrics on the Social Web, and its application to recommender systems, presenting case studies ranging from the earliest recommender systems to present day implementations which can involve trust metrics. Chapter 3 introduces and evaluates the first contribution of this work, an algorithm for computing trust in recommender systems.

Chapter 4 continues with trust-based recommendation, presenting a novel system for evaluating trust based on a user's history of errors in the recommendation process. A prototype application is presented in Chapter 4 to evaluate the benefits of trust based recommendation. Chapter 5 examines the concept of robustness of trust based recommender systems in attack situations, documenting the implementation and testing of several attack strategies and providing analysis of experimental results. Chapter 6 introduces our trust visualisation application *PeerChooser* and details its theory, design, implementation and evaluation as both a control mechanism and as an explanation interface for collaborative filtering. Chapter 7 deals with issues of trust in online auctions and introduces *AuctionRules*: an algorithm for computing trust in online auctions based on lightweight natural language processing of freetext feedback comments. The chapter details design, implementation and experimental results using data crawled from eBay.

The closing chapter outlines the main contributions of this work, beginning with a brief overview, then a detailed discussion of the merits, benefits and drawbacks of the core ideas. Detail is presented on ideas for further research in this area at both specific and general levels. Chapter 8 ends with a brief conclusion.

This thesis has three appendices. The first contains detail on the user surveys performed in the visualisation/interaction experiments from Chapter 6, and the second presents a detailed discussion of the varying concepts of trust across different disciplines, and the third presents code snippets illustrating some of the core actions available to users in the interface of the *PeerChooser* recommender system.

1.5 Summary

This chapter has presented a broad introduction into the concept of trust in the Social Web. The chapter also presented a discussion of the evolution of the Social Web as a "user participation" platform and analysed several motivating examples of why defining reliable models of trust is so important on this platform, using concrete examples from popular web sites. The following chapter provides a background research into the theory and applications relevant to this work.

Chapter 2

Recommender Systems

2.1 Introduction

In the 1990s, as more information became available on the internet, users began finding it difficult to get the information they needed in a timely manner. This problem was particularly noticeable in news groups such as USENET [102]. The term "information overload", (although originally coined in 1970 by American sociologist Alvin Toffler in his book "Future Shock"*) [116], became synonymous with searching through page after page of unwanted internet articles looking for relevant or interesting ones. As a solution to the information overload problem, recommender systems were developed to help users get the right information at the right time by responding to learned user preferences more proactively to generate recommendations and suggestions that are likely to suit user needs.

Recommender systems operate using a range of different techniques, they can be implemented as basic content-matching systems [75] [43], which simply match textual descriptions of recommendable items to a description in a user's preference profile. Content-based recommendation techniques are generally considered as a rudimentary class of recommender and historically they suffer from problems such as narrowness, since they can only recommend items similar to what is already in a users profile, they also fail on

*Toffler's book "Future Shock" was actually a continuation of a popular article which he wrote for the February 1970 edition of Playboy Magazine

non-machine analysable data such as humour for example. Content based techniques are discussed later in this chapter in Section 2.3.1. The most widely used technique for generating recommendations is Automated Collaborative Filtering ACF [15][58][92][102][107]. ACF attempts to model the normal social process of asking a friend for a recommendation. In brief, ACF algorithms compute a neighbourhood of users based on some correlation function (usually Cosine or Pearson's Correlation [92] over vectors of rating data) and use that neighbourhood to predict items that a user has not yet seen and that have been "liked" by his/her closest neighbours, see Section 2.2. ACF has two typical modes of operation, recommendations can be *pro-active*, that is, they suggest an ordered list of the top n recommendable items to be presented to a particular user. ACF can also function in a *reactive* mode, in that a user can specify a particular item he/she is interested in and request a predicted rating on that item from the system. These are generally termed "*specific* recommendations".

Recommender systems can operate using other techniques besides ACF. For example items can be recommended based on demographic data, i.e: people in age-group p generally like q, or based on domain knowledge, i.e: item p can satisfy user requirement q because it has feature r. They can also operate based on similarity between items, association rules and utility functions. This chapter provides detailed discussion and comparison of each type of recommender system, using example systems, pseudocode algorithms and case studies where appropriate.

The ultimate goal of any information filtering system is to sort through large volumes of information and present to the user those which are likely to satisfy his or her information requirement. In order to sharpen this definition, a distinction should be drawn between information retrieval and information filtering. In some domains, for example Google Groups*, the retrieval effort is minimal because the information is directed to the user. In other domains (e.g. the Social Web) the retrieval effort can be considerable because no mechanism exists to draw new information to the attention of a filtering system.

Recommender systems are a tool for tackling the information overload

*http://groups.google.com/

problem by attempting to tailor each user's information space to meet their individual information needs. They have been deployed across many different domains, from movie (eg Movielens*) and book recommenders like Amazon† to people and music recommenders, for example Pandora and the Music Genome Project‡.

To date, most recommendation systems are based on one or more of a set of standard recommendation algorithms. Standard recommendation algorithms include the class of collaborative recommendation algorithms and are discussed in the following section. The more basic class of content-based approaches to recommendation, are presented in Section 2.3.1.

Table 2.1, from Burke [15] gives a good overview of the background, input and processes involved in the various different recommendation techniques. Table 2.1 has been extended to include recently developed techniques such as critique [72] and trust-based recommendation [5]. This will be dealt with in a later in Section 3.2.4. Nomenclature for Table 2.1 is as follows: **U** is the set of users whose preferences are known, **I** is the set of items over which recommendations might be made, **u** is the active user, ie the user to whom we are making recommendations, and **i** is some item for which we would like to predict **u**'s preference.

2.2 Automated Collaborative Filtering

Collaborative filtering models the social process of asking trusted friends for their opinions or 'recommendations' on items; for example, asking a friend for his opinion about a movie he has seen. The underlying assumption of the ACF approach is that those who agreed in the past tend to agree again in the future. For example, a collaborative filtering system for music could make predictions about which music a user should like given a partial list of that user's tastes (likes or dislikes). Note that these predictions are specific to the user, but use information gleaned from many users who are considered to be related/similar to the target user. This differs from the simpler approach of

*http://movielens.umn.edu/

†www.amazon.com

‡www.pandora.com

Technique	Background	Input	Process
Collaborative (ACF)	Ratings from **U** of items in **I**	Ratings from **u** of items in **i**	Identify users in **U** similar to **u**, and extrapolate from their ratings of **i**
Content-based	Features of items in **I** with textual descriptions	**u**'s ratings of items in **I**	Generate a classifier that fits **u**'s rating behaviour and use it on **i**.
Demographic	Demographic information about **U** and their ratings of items in **I**	Demographic information about **u**.	Identify users that are demographically similar to **u** and extrapolate from their ratings of **i**.
Utility-based	Features of items in **I**	A utility function over items in **I** that describes **u**'s preferences	Apply the function to the items and determine **i**'s rank.
Knowledge-based	Features of items in **I**. Knowledge of how these meet a user's needs	A description of **u**'s needs or interests	Infer a match between **i** and **u**'s need
Critique-Based	Ratings from **U** of items in **I**, Features of $i \in$**I**	Feature Critiques and Ratings from **U** of items in **I** and some trust model.	Generate initial recommendation set from **I**. Users critique each feature in the set. Recommend a new set using the refined Feature preference.
Trust-Based	Ratings from **U** of items in **I** Trust feedback on users in **U**	Ratings from **U** of items in **I** and some trust model.	Identify users in **U** similar to **u** who are trustworthy, and extrapolate from their ratings of **i**

Table 2.1: Different Recommendation Techniques (from [15]).

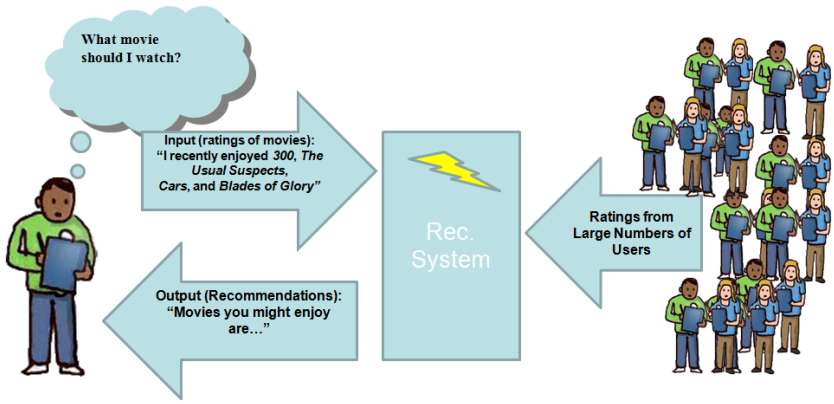

Figure 2.1: The Basic Interaction Paradigm of Automated Collaborative Filtering

giving an average (non-specific) score for each item of interest, for example based on its number of votes. The following sections discuss a range of ways to perform collaborative filtering. Figure 2.1 shows the basic paradigm of a collaborative recommender system.

Automated Collaborative Filtering (ACF) was first introduced in 1989 by John Hey of LikeMinds (United States Patent 6487541), and furthered in 1994 by the GroupLens [102] research group. The task of an ACF system is to predict items that a particular user will find useful. This task is generally achieved in two ways, *Prediction* and *Recommendation*. During prediction, the ACF system generates a rating for a *specific* item chosen by the user seeking the recommendation. During recommendation, the ACF system generates a ranked list of the top-n items which it thinks a particular user will like the most. For the remainder of this document we will refer to the user receiving recommendations as the *active user*. Predictions are based on a database of other users' votes on the set of recommendable items. ACF systems can be categorised into two main streams: Memory-Based ACF, which operates over the entire database to make a prediction in a totally online process, and Model-Based ACF, which uses the database to compile a similarity model in an offline process.

As stated by Herlocker et al. [46], *"collaborative filtering provides three key additional advantages to information filtering that are not provided by*

content-based filtering."

- Collaborative filtering allows for filtering on data that is hard to analyse by automated processes, such as movies and feelings, as it is the human user who rates the data and from this the relevance of the data to other users is computed.

- Collaborative filtering has the ability to filter items based on quality and taste, as humans are capable of analysing on such dimensions while it is hard for computers to do so.

- Collaborative filtering systems can give serendipitous recommendations. Herlocker et al found that *"serendipitous recommendations occur frequently in the movie domain, with the collaborative filtering system accurately recommending movies that a user would never have considered otherwise".*[46]

Another common categorisation of ACF systems is the manner in which they ascertain user ratings. There are two main rating types: *implicit* and *explicit*. Implicit ratings are a passive, non-invasive method of taking information from users, which has the advantage of not bothering the user with tedious questions about their tastes which are generally in the form: "rate the following items...".

Explicit rating schemes have advantages in that the user tells the system exactly what tastes they have, and they may consequently feel a sense of interaction with the system, and therefore an increased sense of trust. However, user tastes are changeable, and users may not be inclined to update their preferences. Also, there is a danger that the time required to input ratings may outweigh the user's attention span. It has been shown in [98] that the accuracy of inferred ratings from implicit ratings is in line with those from explicit ratings. In chapter 7 of this thesis we introduce our novel technique for rapid elicitation of user preference knowledge in the *PeerChooser* application. Table 2.2, from Nichols [42], outlines some of the methods used to infer user ratings non-invasively.

Action	Notes
purchase (price)	buys item
assess	evaluates or recommends
repeated use (number)	e.g. multiple check-out stamps
save/print	saves document to a personal storage
delete	deletes an item in a shopping basket
refer	cites or otherwise refers to an item
reply (time)	replies to an item
mark	add to a 'marked' or 'interesting' list
examine/read (time)	looks at a whole item
consider (time)	looks in abstract
glimpse	sees title/surrogate in list
associate	returns in search
query	association of terms from queries

Table 2.2: Methods for Non-Invasive Inference of User-Ratings.

2.2.1 Memory v/s Model-Based ACF

A distinction can be drawn between two broad approaches to the design of ACF algorithms: *memory* and *model-based*. Memory-based algorithms, (which include the commonly implemented classes of User-based and Item-based methods) utilise the entire database of user ratings to generate a prediction. These systems employ statistical techniques to find a set of users, known as neighbours, that have a history of agreeing with the target user (i.e. they either rate different items similarly or tend to "buy" similar sets of items). Once a neighbourhood of users is formed, memory-based ACF systems use different algorithms to combine the preferences of neighbours to produce a prediction or top-N recommendation for the active user. The techniques, also known as k-nearest-neighbour techniques are among the most popular form of ACF, and are widely used in practice. See [98] [86] [52] and [102] for example. There are some inherent benefits and drawbacks with Memory-Based approaches to ACF. For example, the lack of an offline modelling process means that the system runs fully in realtime and new users and new ratings effect the system immediately. This is a clear advantage for

Memory-based over Model-based techniques which must compute similarity in scheduled, offline processes. On the downside however, these techniques are very memory intensive and as a result they have difficulty scaling to large numbers of users.

At a general level, model-based collaborative filtering algorithms provide item recommendations by first developing some model of similarity between users in the system, generally in an off-line computation process. Algorithms in this category often take a probabilistic approach and envision the collaborative filtering process as computing the expected value of a user prediction, given his/her ratings on other items. The model building process is performed by various machine learning algorithms such as Bayesian models [99], clustering [106], and rule-based approaches [92] or simple profile correlation. The Bayesian network model formulates a probabilistic model for the collaborative filtering problem. The clustering model treats collaborative filtering as a classification problem and works by clustering similar users into the same class and estimating the probability that a particular user is in a particular class C, and from there computes the conditional probability of ratings. Rule-based approaches can be viewed as model-based and apply rule discovery (for example association rules, detailed in Section 2.2.4) algorithms to find relationships between co-rated items. A rule-based technique would then generate item recommendations based on the strength of the discovered association between items, as in [92], [58] and [108].

2.2.2 User-Based Collaborative Filtering

User-based Collaborative Filtering algorithms are a subset of the *memory-based* class of algorithms. User-Based ACF is performed by calculating the similarity between pairs of users in the system. Users are represented as vectors of ratings in an n-dimensional rating space, and similarity between users is usually defined as some function of the angle between the two vectors. Many user-based ACF systems employ Pearson's correlation coefficient [12] to compute the distance between the vector representations of users. This distance, or correlation, is the similarity value upon which groups of users (ACF neighbourhoods) are formed.

Prediction is achieved in user-based ACF by selecting items which a user's peers' have rated highly, but have not yet been rated by the user. There are a range of methods for calculating similarity coefficients, such as Cosine Similarity, Jaccard's coefficient [59], etc. We use Pearson's correlation as it is the most widely used in ACF and allows us to make a more direct comparison with other existing systems. When the peer-group of users is chosen based on the similarity to the active user (the user receiving recommendations), a weighted combination of their ratings is used to generate predictions, as seen in [75]. We next examine the user-based ACF algorithm in terms of how it builds its model and calculates predictions. We will repeat this examination for all of our recommendation algorithms.

The following is a step-by-step outline of a basic ACF algorithm:

1. Calculate the similarity between the active user and every other user based on some similarity function, for example Pearson's correleation, shown in Equation 2.1 below.

2. Form the peer group by selection of the top-n most similar users.

3. Select items rated highly in the peers' profiles which are not in the active users profile.

4. Return a ranked list of these items as the recommendation set.

$$corr_{x,y} = \frac{\sum_{u \in U}(R_{k,x} - \overline{R}_x)(R_{k,y} - \overline{R}_y)}{\sqrt{\sum_{u \in U}(R_{k,x} - \overline{R}_x)^2 \cdot \sum_{u \in U}(R_{k,y} - \overline{R}_y)^2}} \tag{2.1}$$

Equation 2.1 gives the Pearson Correlation in the first step of the algorithm. In this equation $corr_{x,y}$ is the Pearson correlation coefficient between user x and user y, $R_{k,x}$ is the rating of item k by user x and $\bar{R}_x$ is the average item rating by user x.

Resnick's Prediction Formula

Prediction calculation in standard ACF can be done using several techniques. By far the most common approach is that of Resnick [102] Resnick's formula

uses the weighted average of deviations from the mean of the neighbours:. In chapters 3, 4 and 5 of this thesis Resnick's formula is adapted in a range of ways to aid in generation of trust values for ACF systems and modified further to allow re-integration of the generated trust values into the system. This formula was chosen as it is widely used in the literature [102] and allows for easy comparison with existing techniques and systems.

$$p_{x,i} = \overline{r}_x + \frac{\sum_{u \in U} (r_{u,i} - \overline{r}_u) \times P_{x,u}}{\sum_{u \in U} (P_{x,u})}$$

where $p_{x,i}$ is the predicted rating of item i by user x, and $p_{x,u}$ is the similarity between users x, and u.

2.2.3 Item-Based Collaborative Filtering

Item-based collaborative filtering algorithms are a general subset of the *model-based* class of ACF algorithms. The first real implementation of an item-based ACF algorithm was shown by Donald Fisk in [28] and more popularly by Sarwar and Karypis in [107]. Item based ACF algorithms use a pre-computed model based on the items in the system rather than the users (as with user-based ACF systems). Historical information on items in the system is analysed to ascertain relations between items. An example of this might be "the purchase of one item often leads to the purchase of another item." This type of filtering can be extremely beneficial to a system with a relatively static number of items [108]. This technique is dealt with in detail in [32] and [33]. It can be used in tandem with other filters to overcome some of their inherent shortcomings. Recent research in [107] shows that this approach can be as accurate as the common user-based approach (in [102] for example) while the speed of the item-based approach is much greater.

The main difference between Item-Based ACF (IBCF) [107] and the User-Based algorithm is that IBCF makes its predictions based on a model of item similarity rather than user similarity. The IBCF algorithm looks at a set of items the active user has rated and computes their similarity to a target item i, in order to evaluate it for recommendation. The IBCF algorithm selects the

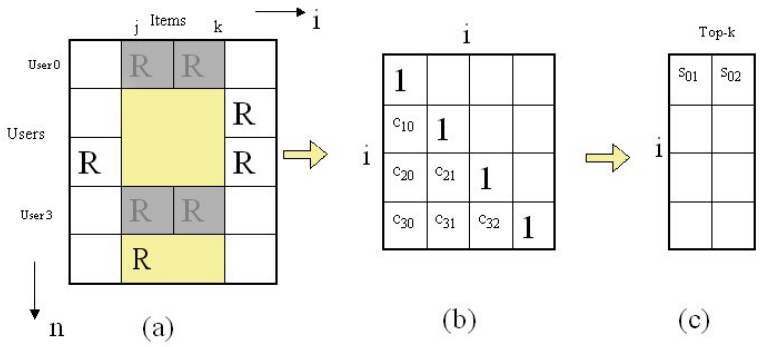

Figure 2.2: Offline Computation of the Item-Based Model.

k-most similar items $\{i_1, i_2, ..., i_k\}$ and their corresponding similarity values $\{s_1, s_2, ..., s_k\}$. A prediction is computed by taking the active user's ratings on these similar items, and weighting these by multiplying them by their similarity values. Again, the algorithm boils down to two main phases: model building and prediction calculation.

Model Building and Similarity Capture

The Model Building phase of an item-based ACF algorithm involves computing similarity over the set of items instead of the set of users.

The algorithm operates by selecting cases where items i and j have been co-rated by the same users. In Figure 2.2, using Pearson's equation from Section 2.2.2, the set U consists of only those users who have rated both item i and item j in the diagram, in this case, users 0 and 3. This calculation results in an $n \times n$ matrix of item similarities, where n is the number of items in the dataset. Generally only the top-n most similar items to each other item are used, each item profile is sorted in non-increasing order, and the top-n from this list are stored as the model. An important advantage of this method is that the entire model-building phase is an off-line computation, resulting in rapid recommendations for online users. This method is also scalable to large numbers of users, as their recommendations are generated from a static item-item model.

Prediction Calculation

The prediction phase of an Item-based ACF algorithm is generally done as follows:

1. User rates a set U of items.
2. A Candidate set C of items is formed by taking the union of the k most similar items for each item $j \in U$. (This is done by simply by taking the top-k from the ordered lists computed in the model-building phase).
3. Any duplicate items are removed, ie any $j \in (C \cap U)$.
4. Find the similarity between each item $c \in C$ and the set U. This is done by consulting the Item-Item correlation matrix from phase 1, and summing the correlation values between each $c \in C$ and every item in U.
5. C is then sorted in non-increasing order w.r.t these similarities.
6. The top-n items in C are the Recommendation Set.

2.2.4 Collaborative Filtering using Mined Association Rules

Before looking at the standard benefits and drawbacks of collaborative filtering systems, we will examine one more approach to the ACF task. The possibility of mining frequent itemsets over "market basket data" was introduced by Agrawal et al. in [2]. This technique is a case-based approach to collaborative filtering, as discussed by O'Sullivan in [92]. Case-based reasoning (CBR) is a well researched field of AI which relies heavily on good content descriptions of its 'cases'. [92] examines ways to leverage the CBR technique to perform collaborative filtering. This technique treats profiles in a collaborative filtering system as cases. There are two main factors involved here: similarity computation and recommendation. Work presented in [85] describes an adaptive recommender system which contains a Case-Based Recommendation component using data mining [10] to arrive at item

similarities and a weighted-sum method in the prediction phase. It has been shown in [10] that the Apriori rule mining algorithm serves as a successful technique for performing similarity computation.

Equation 2.2 from [92] defines sparsity in a set of user ratings. This simple formula always produces a sparsity value between 0 and 1. In cases where data is sparse, ACF systems generally perform badly due to lack of overlap between the ratings of a given pair of users.

$$1 - \frac{number\ of\ nonzero\ entries}{number\ of\ total\ entries} \tag{2.2}$$

O'Sullivan's work in [92] addresses this sparsity problem in ACF by harnessing the learned rules from the Apriori algorithm to fill in similarity gaps by computing similarities between users who had rated no items in common. Association rule-based ACF is also seen in [113]. It is, in fact, very similar to the item-based ACF algorithm described in the previous section. The Apriori algorithm [2][10] discovers hidden relationships between items by generating rules of the form $A \Rightarrow B$, where A and B are itemsets. We can then use these rules to generate an item-item similarity matrix similar to that in Figure 2.2. Association rules of this form have two important metrics: Rule Confidence and Rule Support. The confidence of a rule $A \Rightarrow B$ is the percentage of profiles containing A which also contain B. The support of a rule of this kind is the fraction of the total profiles in a database which uphold the rule:

$$support(A \Rightarrow B) = P((A \cup B) \subseteq T) \tag{2.3}$$

$$confidence(A \Rightarrow B) = \frac{support(A \cup B)}{support(A)} \tag{2.4}$$

This association rule based algorithm is similar to IBCF in that it contains a model-building phase and a prediction phase.

Model-Building and Similarity Capture

Taking each set of items in a profile as an itemset, the Apriori algorithm can derive item-item association rules and confidence levels. As can be seen

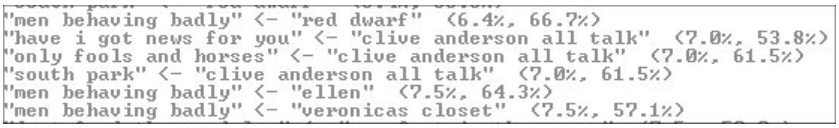

Figure 2.3: Sample Apriori Output for a Television Program Dataset

in Figure 2.3 in the following section, these rules only incorporate binary-chaining, that is, they do not consider rules of the form $A \rightarrow B \rightarrow C$. This type of rule would identify further relations between the items in a dataset, and so can be forwarded as a method to combat the sparsity problem in ACF. There is however, a trade-off with this approach: The more tenuous links become between items, the greater the risk of dissimilarity. Recommending a bad item is generally worse than failing to recommend a good item [92].

Prediction Calculation

The prediction phase of this algorithm is the same as in the IBCF algorithm. Thus the only difference between the two algorithms lies in the model-building. Item-Item similarity is derived directly from the association rules in order to compute the model. This is achieved by taking the confidence levels for rules generated by the Apriori algorithm, and using them as probabilities in the Item-Item similarity matrix. There are many ways to combine the probabilities with the support values to attain the Item-Item model. These are discussed in detail in [92]. One approach, used in [85] uses the apriori support for a rule, multiplied by the probability of the rule to arrive at a similarity value between two items. The example in Figure 2.3 is taken from experiments in [85]. The first rule is: "men behaving badly" $\rightarrow$ "red dwarf" is <6.4%, 66.7%>. From this we get a similarity for these two items of 4.26% (or 0.426 for our calculations).

$$sim(A, B) = confidence(A \Rightarrow B) * support(A \Rightarrow B) \tag{2.5}$$

The Apriori Algorithm

The Apriori algorithm is widely known in data-mining communities and used in collaborative filtering by O'Sullivan in [92] for example. This section provides detail on the modus operandi of this algorithm.

Association rule induction is a powerful method for so-called market basket analysis, which aims to find regularities in the shopping behaviour of customers of supermarkets, mail-order companies, etc. With the induction of association rules one tries to find sets of products that are frequently bought together, so that from the presence of certain products in a shopping cart one can infer (with a high probability) that certain other products are present. Such information, expressed in the form of rules, can often be used to increase the number of items sold, for instance, by appropriately arranging the products in the shelves of a supermarket (they may, for example, be placed adjacent to each other in order to invite even more customers to buy them together) or by directly suggesting items to a customer which may be of interest to him/her.

An example of an association rule is "If a customer buys wine and bread, he often buys cheese too." It expresses an association between (sets of) items, which may be products of a supermarket or a mail-order company, special equipment options of a car, optional services offered by telecommunication companies, etc. An association rule states that if we pick a customer at random and find out that he selected certain items (bought certain products, chose certain options etc.), we can be confident, quantified by a percentage, that he also selected certain other items (bought certain other products, chose certain other options, etc.). A more formal definition of the Apriori problem itself is follows.

Problem Statement: Let $I = \{i_1, ..., i_m\}$ be a set of literals, called items. Let D be a set of transactions, where each transaction T is a set of items. We say that a transaction T contains itemset X, if $X \subseteq T$. Itemset X has support "s" in the transaction set D if no less than s transactions in D contain X. Given a set of transactions D, the problem of mining frequent itemsets is to generate all itemsets that have support greater than the user-specified minimum support.(see Section 2.2.4) The general procedure for this

algorithm is as follows:

- Assume $a_{k,j}$ is the k^{th} value of the j^{th} attribute
- Assume $\bar{s_r}$ is the minimum coverage for a combination of r attributes
- Identify all pairs $(a_{k,j}, a_{l,m})$ where $j \neq m$ (ie different attributes) that have coverage greater than $\bar{s_2}$
- Identify all triplets $(a_{k,j}, a_{l,m}, a_{i,n})$ where $j \neq m \neq n$ that have coverage greater than $\bar{s}_3$
- Once all desired combinations are identified, compute the accuracy for each combination
- Report only those combinations above a user-defined minimum threshold.

O'Sullivan et al. [92] discuss rule chaining as a method of improving the performance of the Apriori algorithm in case-based recommender systems. This is where one item can indirectly imply another item by incorporating rules of the form $X \Rightarrow Y \Rightarrow Z$ to compare user profiles.

2.2.5 Computing Profile Similarity

ACF Recommendation algorithms must implement some form of similarity computation. The computation can be performed over a range of different types of data depending on the algorithm, for example, vectors of ratings in a users profile or ratings on particular items. There are a large number of ways to compute similarity over ratings data. This section provides a brief overview of three of the most common methods used to compute similarity in recommender systems.

Correlation-Based Similarity

The most commonly implemented similarity functions in ACF are Correlation-based over sets of user ratings. In correlation-based similarity functions, similarity between two vectors I and J is measured by computing the distance between their individual elements $i \in I$ and $j \in J$. Pearson$-r$ correlation is the most commonly used correlation function in ACF systems. To compute Pearson correlation between two ratings profiles for example, we must first isolate the co-rated cases (i.e. cases where the users rated both i and j) as shown in Figure 2.2. Let the set of users who rated both i and j be denoted by U, then the correlation is given by Equation 2.1 in Section 2.2.2.

Cosine-Based Similarity

In cosine-based similarity, two potentially similar items are modelled as vectors in m-dimensional user-space. The similarity between two items is measured by computing the cosine of the angle between the two vectors. The similarity between items i and j, denoted by $sim(i, j)$ is given by

$$sim(i, j) = \cos(\vec{i}, \vec{j}) = \frac{\vec{i} \cdot \vec{j}}{\|\vec{i}\|^2 \times \|\vec{j}\|^2} \tag{2.6}$$

where "." denotes the dot-product of the two vectors.

Jaccard Similarity

The Jaccard similarity coefficient (also known as the Tanimoto coefficient) is used to measure the similarity between profiles containing *binary* elements. This technique examines the OR case as well as the AND case. For example, in two profiles A and B, this technique will look at the items which are present in both sets A and B, items which are present in A but are absent in B, and items which are absent in A but are present in B. In other words, it measures the ratio of $intersection$ of profiles to the $union$ of profiles.

Adjusted-Cosine Similarity

One fundamental difference between the similarity computation in user-based ACF and item-based ACF is that in the case of user-based ACF the simi-

larity is computed along the rows of the user-item matrix but in the case of item-based ACF, similarity is computed along the columns, i.e. each pair in the co-rated set corresponds to a different user (Figure 2.2). Computing similarity using the basic cosine measure in the item-based case has one important drawback: the difference in rating scales between different users are not taken into account. The adjusted cosine similarity offsets this drawback by subtracting the corresponding user average from each co-rated pair. Formally, the similarity between items i and j using this scheme is given by

$$sim(i,j) = \frac{\sum_{u \in U}(R_{u,i} - \bar{R}_u)(R_{u,j} - \bar{R}_u)}{\sqrt{\sum_{u \in U}(R_{u,i} - \bar{R}_u)^2}\sqrt{\sum_{u \in U}(R_{u,j} - \bar{R}_u)^2}}.$$

Here $\bar{R}_u$ is the average of the u^{th} user's ratings.

2.2.6 Common Problems with ACF Systems

There exist a host of standard problems which all collaborative recommenders must address if they are to produce high-speed and high-quality recommendations. Despite the large amount of research effort into ACF systems, many of the original problems still exist, and good solutions have yet to be developed. This section details the main shortcomings and sources of failure for most ACF algorithms.

Data Sparsity

ACF systems require a sufficient number of users to interact with the system and to provide ratings before the system can provide quality recommendations with good accuracy and coverage. Even with a large membership, each user must provide enough rating information to achieve a reasonable level of profile overlap so that similarity with a peer-group can be ascertained by the system.

It has been shown by Resnick [102] and later by Konstan [108] that at low rating density levels, many users receive recommendations which are below the standard of non-personalised recommenders. Much research has been

carried out into potential solutions for this problem. O'Sullivan et al [92] use case-based recommendation to decrease sparsity in the PTV dataset. Melville and Mooney [75] employ content-based methods to decrease sparsity in the EachMovie dataset, and Goldberg et al. [43] use the Eigentaste algorithm to decrease the problem on the Jester dataset.

The Latency Problem

In addition to poor performance caused by sparse ratings data, *latency* of those ratings is another influencing factor on the performance of an ACF system. Latency is distinct from sparsity because it considers the specific items which are being rated, as opposed to simply counting ratings as a percentage of the total ratable data (as in Equation 2.2 above). The latency problem is defined as lack of sufficient user-ratings on items in an ACF system to provide enough *ratings overlap* for good recommendations. There are a number of influencing factors on the number of user ratings of items. These include the "gray sheep" problem [19], where a user's tastes do not fit well into one particular cluster, and therefore there is insufficient overlap to provide recommendations. The *early rater* problem affects new, unique, or esoteric items. It essentially means that an item does not have enough ratings to get recommended by the system. In some ACF research, this problem has also been termed the *ramp-up* [118] problem, or the *cold-start* problem [92]. Smyth et al. [113] propose use of the Apriori [10] algorithm as a technique for combatting the latency problem in ACF systems.

Scalability Issues in ACF

Throughout the last 10 years or so, much ACF research has focussed on finding solutions to problems of latency, sparsity and cold-start as well as ways to explain recommendations to end-users [46]. As a result of this, research into scalability for ACF has been left somewhat on the back-burner. This may be as a result of hardware advances such as cheaper memory and faster processors. It may also be a factor that early ACF systems (for example the Tapestry system [42]) were relatively small and did not have massive databases to contend with. Scalability in ACF systems has now become a

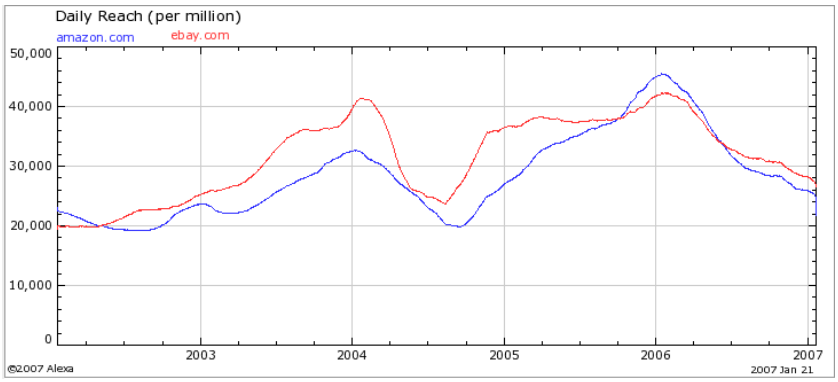

Figure 2.4: Comparison of Website Traffic between Amazon.com and eBay from 2002 to 2007. (source: http://www.alexaholic.com)

vital issue: the Amazon.com database scales to millions of users and items. Newer recommender systems such as StumbleUpon are gaining increasing popularity, and growing in size. As of September 2007, StumbleUpon has a user base of 3,498,737.* Figure 2.9 gives an idea of the volume of traffic visiting both Amazon.com and eBay over the last five years. The scale on the y-axis of Figure 2.9 represents the daily "Reach". Reach measures number of visits to a site and is typically expressed as the percentage of *all* Internet users who visit that site.† So, for example, if a site like Amazon.com has a reach of 28%, this means that if you took random samples of one million Internet users, you would on average find that 280,000 of them visit Amazon.com. It is clear from the many millions of users that scalability of collaborative filtering systems is a vital factor for the performance and therefore success of the system.

Goldberg's Eigentaste algorithm [43] claims to produce recommendations in linear time to the number of items in the system. However, this involves a lot of offline computation and model-building. Item-Based techniques such as those from Fisk [28], Karypis [58] and Sarwar [108] [107] tend to scale upwards with a lot more ease than common user-based ACF implementations, the fastest of which operate in quadratic time complexity, with a less-than-

*http://www.stumbleupon.com/

†http://www.mediasmith.com/white/msn/msn062002.html

linear gain in throughput for an incremental use of hardware resources [102]. Solutions such as dimensionality reduction via factor analysis and singular value decomposition have been used to combat the latency problem by Goldberg in [43], and by Billsus and Pazzani in [8], where the singular value decomposition (SVD) of the original user ratings matrix is used to project user ratings and rated items into a lower-dimensional space. By doing this they eliminate the need for users to have co-rated items in order to be predictors for each other.

Privacy Issues in ACF

Most personalisation techniques present a trade-off between the level of privacy and the quality of personalisation. The more an ACF system knows about a user, the more equipped it is to provide that user with high quality recommendations.

The development of more general Web standards has solved some ACF privacy issues. One solution is the Open Profiling Standard (OPS). which was introduced by the World-Wide-Web Consortium (W3C)* as a way of giving control of data back to the user.

The W3C approach to the privacy versus personalisation tradeoff is known as the Platform for Privacy Preferences Project (P3P) which aims to give Internet users more control over their personal information. P3P technology is designed to allow consumers to express their privacy preferences through their browser, which communicates those preferences to web sites in a machine-readable format. It offers a technological alternative to having consumers read the privacy policy at each site. A users browser would automatically read a sites privacy policy to see whether it meets the users preferences. If a site shares data in ways that go beyond the users preferences, the user can terminate the connection. This new technology, discussed in [52] is expected to change the face of recommendation techniques as we know them today. Intuitively if users are provided with a more secure environment, they will be more likely to disclose more information upon which better quality recommendations can be based. There is also an issue of portability, or gen-

*http://www.w3c.org

eralisability to be considered: Performance of ACF systems would increase greatly if ratings and profile information could be harnessed across multiple platforms, systems and locations. For example, if the Amazon.com recommender system could know about the plethora of books that I bought when I walked into the Barnes and Noble store, it would learn more about my tastes, and of course not recommend books which I already have read. This is a somewhat utopian view however, in practice, users generally do not like systems which profile too much data as it is viewed as an infringement on their privacy. [107]. Technologies such as social bookmarking and FOAF [24] are enabling control over information flow between applications and taking the first steps towards a framework for high-performance, cross-platform and even ubiquitous ACF systems.

2.3 Alternative Recommendation Strategies

Up to now the discussion has focused mainly on ACF, which is the main approach to the collaborative filtering adopted in later chapters. The following sections provide an overview of the various other strategies for filtering information for a target user. Different strategies have both strong and weak features throughout the recommendation process. Figure 2.5 broadly depicts the advantages and disadvantages of each technique. The variance in strengths and weaknesses over different techniques shown in Figure 2.5 illustrates the importance of combinations and hybrids of the basic techniques, which are discussed in Section 2.3.6

2.3.1 Content-Based Recommendation

Content-Based recommenders automatically gather a user profile based on the features of the items that user has rated. Content-based methods are therefore generally restricted to domains where items have rich textual descriptions, for example Web pages. The type of profile developed depends on the learning method employed. Burke [15] describes the use of Decision Trees, Neural Networks and Vector-Based representations of user profiles in content-based recommenders. Content-Based recommenders are emergent

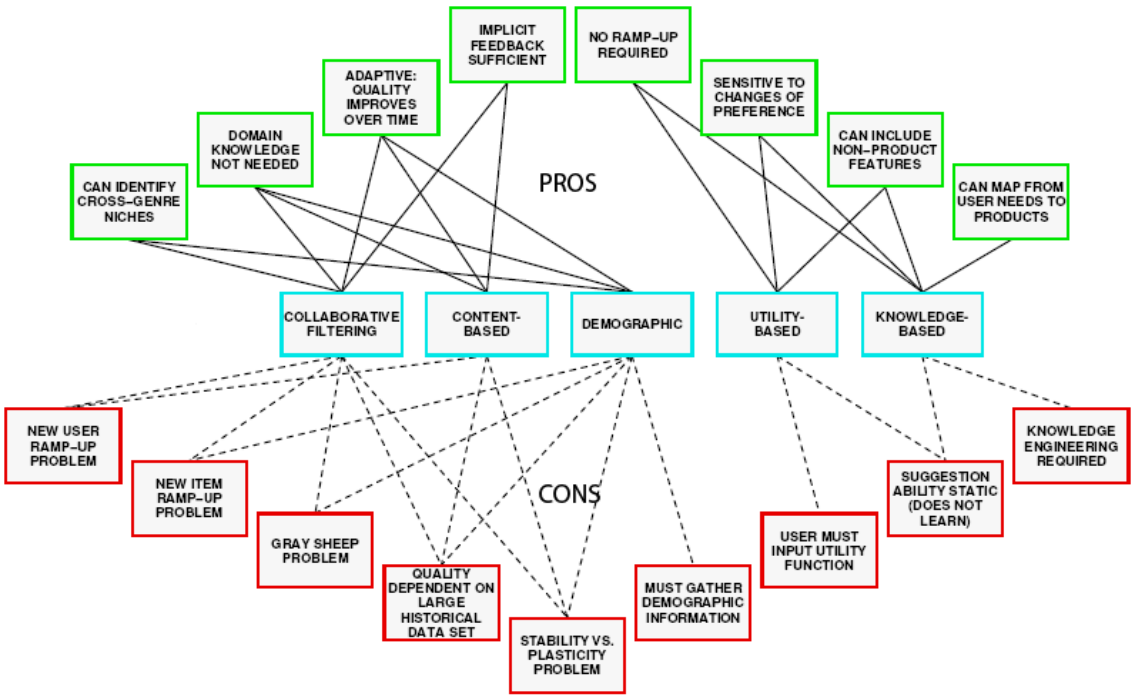

Figure 2.5: Overview of Strengths and Weaknesses of each Recommendation Technique. (from Burke [14])

systems (as are ACF systems), in that user-models are built up, extended and refined as the system is used. In January 2006, Google launched is personalised news recommendation service on Google News*. The feature offers story recommendations "based on what you've searched for and clicked on in the past", and operates by harnessing a user's Google search queries and applying a content-based approach to recommendation of new articles by matching text based search queries against the content of recommendable news articles. As a user clicks on more news articles the personalised profile becomes more refined and the content based recommendations improve over time. This feature of Google news only works if a user is logged into their personalised Google account while using the news site. Figure 2.6 shows recommended articles on Google news.

Advantages of content-based methods include the following:

*http://news.google.com/

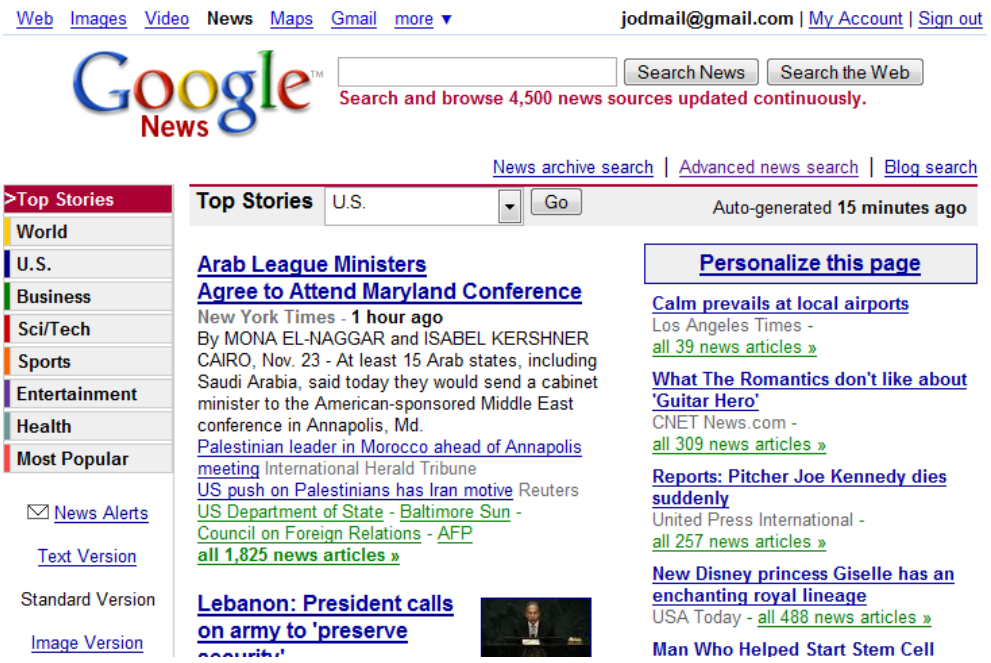

Figure 2.6: Screenshot of Personalised News Articles from the Google News Content-Based Recommender System.

- Knowledge of domain is not required. (Barring descriptions of individual items);
- Adaptive quality improves over time with feedback from users;
- Implicit user-feedback is sufficient to improve recommendation quality.

Some of the disadvantages of these methods are:

- Recommendation quality is dependant on a rich historical dataset;
- They suffer from the "ramp-up" problem, i.e: new users have to rate a sufficient number of items before the system can reward them with quality recommendations;
- Narrowness: Content-based methods cannot provide serendipitous recommendations. i.e: users are generally recommended items which are

similar to those already seen. This lack of diversity rules out recommendation of items which may be of interest to a user by overlooking any item that does not have a similar textual description. For example, analysis of the MovieLens dataset shows that more than 80% of users who rated the movie "Shawshank Redemption" highly (4 or 5 out of 5) and had also rated "Pulp Fiction"', gave "Pulp Fiction"' a rating of 4 or 5. The content descriptions of these two movies are completely different and accordingly, a content based recommender system would not tend to recommend "Pulp Fiction" to a user who has rated "Shawshank Redemption" highly, even though from analysis of the data it is obvious that the the movies are positively correlated. O'Sullivan et al. [92] describe this problem in more detail.

2.3.2 Utility-Based Filtering

Both utility-based and knowledge-based filters differ from content-based and ACF methods in that they don't attempt to compile long-term profiles of users. Instead, recommendations are made based on an assessment of the correlation between a user's need and the set of available options. Utility-based techniques make recommendations based on the computation of a utility function created for each individual user. In their 2004 work, O'Mahony et al propose a Utility-based neighbourhood formation for efficient and robust collaborative filtering. [91]. Work in 2006 by Park et al. [105] outlines the implementation of a utility based system for recommending music. In this system, users receive recommendations based not only on their basic profile information, but also based on the current *context*. Park defines an explicit set of possible "moods" which a user can be associated with. Based on a range of sensory inputs and information crawled from the web, the system computes the probability that a user is in a certain mood. By computing probabilistic values representing the user's current mood and by combining this with information already stored in a user's profile the system can compute utility-based music recommendations. Park et al. perform a live user survey assessing user satisfaction with recommendations from their system in parallel with other techniques. In these experiments, the utility based

approach was found to improve user satisfaction compared with a standard ACF approach.

Linden et al. [66] describe a candidate critique agent, which develops and refines a utility function based on dialogue with the user. The basic operation of this critique agent is as follows:

- Suggest optimal and near-optimal solutions based on the system's current model of the user's preferences;
- Elicit and refine the user model;
- Possibly by using bad candidates, encourage the user to refine criteria;
- Indicate the range of available solutions in the dataset.

These agents provide additional information about a user's preferences, which ultimately leads to better recommendations. Utility-Based methods have the following advantages:

- There is no ramp-up problem: users do not have to wait for the system to build up a detailed profile before good recommendations can be made;
- This technique is sensitive to changes in user preference. (Does not rely on old models etc.)
- Can include non-product features.

Some of the disadvantages include:

- A user must enter a utility function, which can be time-consuming;
- The system's suggestive ability is static, ie the system does not learn.

More recent work by McCarthy et al. [72] introduces a dynamic critiquing recommender system which operates on a similar principle to Linden's system. Work in [72] applies dynamic critiquing to a digital camera recommender system, allowing users to refine the recommendation by critiquing each feature of the recommended item individually. This technique is discussed in Section 2.3.4.

2.3.3 Knowledge-Based Filtering

Knowledge-based filters attempt to draw inferences about user requirements and tastes based on specific information about the items and users it is recommending to. The core difference between the inferences made by a knowledge-based filter and the general inferences of the other recommendation techniques is that this filter uses functional knowledge, i.e. they use knowledge about how a particular item can satisfy a specific user requirement. This knowledge affords the system the ability to reason about the relationship between a user need and a candidate item. User profiles can be built using any knowledge which supports this inference. Google's *PageRank* [93] uses information about links between Web-pages to define hubs and authorities. Older knowledge-based recommender systems include Lucent's *Notebook Expert* [52] and Burke's *FindMe* [61]. A newer knowledge-based recommender which is part of the Social Web is the StumbleUpon system*. This system harnesses information from the bookmarks in a users Web Browser to attain some knowledge to enhance the explicitly provided user profile. This knowledge is used on top of user's like/dislike ratings of web pages they visit. The StumbleUpon system is discussed in a case-study later in this chapter. Other popular knowledge based recommendation systems include *recommender.com* and Burke's *Entree* system [14] which also recommends restaurants. In these systems, specific knowledge is used about each restaurant and users tweak their requests by specifying commands such as "more like this one" or "cheaper please". Figure 2.7 shows the recommendation screen to the *Entree* system. The circular buttons at the bottom of this image show the range of requests a user can make to tune their recommendations in the *Entree* system.

The advantages of knowledge-based methods include:

- They do not suffer from the ramp-up problem;
- There is sensitivity to changes in user-preferences and tastes;
- Non-product features can be included;

*www.stumbleupon.com

Figure 2.7: Screenshot of Burke's *Entree* Knowledge-Based Restaurant Recommender System.

- The technique can map from user needs to products.

Some of the disadvantages of knowledge-based filtering techniques are:

- There is a *lot* of knowledge-engineering required, which is a big drawback;
- Essentially the system does not learn;
- Because knowledge-based techniques require lots of information about the particular product domain they operate in, these systems have a poor performance record in large or dynamic product spaces.

2.3.4 Critique-Based Filtering

Dynamic critiquing of recommendations is a recent approach to ACF developed by McCarthy et al. [72]. In their critiquing approach, McCarthy et al

define a novel type of "recommendation session" which consists of a cyclic process whereby recommendations are presented to the user in the usual ACF manner, and then users are afforded the opportunity to examine individual features of each recommended item and *critique* them by making simple preference statements such as "less expensive", "higher quality" "longer battery life" etc. McCarthy et al. perform a study of the effects of increased diversity on the displayed, "critiquable" features of items in their system [73]. In this study it was found that applying diversity generally makes the critiques more useful, leading to shorter critiquing cycles (i.e.: on average the user becomes satisfied with the recommendation by critiquing less features). Figure 2.8 shows McCarthy's camera recommender system *Quikshop* which was used as a test-bed for their experiments in dynamic critiquing. In this screenshot the currently recommended item description is displayed on the left, with an image. In the centre of the page, users can specify preferences about the item on a per-feature level by clicking on the up and down arrows to tweak or "critique" the features to the desired level. There are several advantages to the critique-based approach, firstly, the user is more involved in the recommendation process and this tends to increase overall satisfaction with the final recommendation, as discussed by Herlocker in [46]. Secondly, the user exerts more control than in standard ACF systems and guides the algorithm towards the target recommendation. This leads to large increases in recommendation accuracy, albeit at the expense of asking the user for extra input. It has been shown in [72] that users generally prefer to use dynamic critiques (which are slightly more invasive than standard ACF techniques) and receive better quality recommendations.

2.3.5 Demographic Methods

Demographic recommendation techniques use a form of stereotyping to arrive at recommendations. Users are categorised into demographic groups based on personal attributes, and recommendations are made based on the tastes of a demographic group.

Recent work by Vosalis and Margaritis [119] describes a technique for using demographic data in an ACF system based on singular value decom-

Figure 2.8: Screenshot of Dynamic Critiquing on the Quickshop.com Camera Recommender System.

position (SVD). In their system, which was tested over the MovieLens dataset [78], demographic data is shown to have a positive effect on the accuracy of recommendations, in addition to overcoming problems of general data sparsity and specific items with few ratings. Vosalis' algorithm is comprised of four steps:

1. Construct demographic vectors for the m users and n items that participate in the recommendation process. The information required for those vectors can be usually found in the utilised Collaborative Filtering data sets, like MovieLens and EachMovie. Once constructed, the demographic vectors are collected in array D.

2. Select one of two possible scenarios: Case 1: Leave the array of demographic vectors, D, intact, or,

   Case 2: Perform SVD on the array of demographic vectors.

3. Proceed with Data Representation which concludes with the construc-

tion of the initial useritem matrix, R, of size $m \times n$.

4. Resume with the Neighbourhood Formation. One of the following possible ways is selected:

   Case 1: Neighbourhood Formation without SVD. The Similarity metric of choice is applied on the ratings of the original useritem matrix, R, and ratings-based correlations for pairs of users or items are generated.

   Case 2: Neighbourhood Formation with Singular Value Decomposition (SVD). SVD is performed on the useritem matrix, R, before proceeding with the Neighbourhood Formation.

5. The Demographic Correlation between the active user or item, and each of the members of its neighbourhood, is calculated by computing their corresponding vector similarities.

One of the most popular demographic recommender systems is the Lifestyle Finder [61], in which a technique of *elicitation* and *exploitation* is used to acquire user information. The Lifestyle Finder elicits information in a "friendly" manner, which does not require the user to answer any identifying information. Once a user has answered some subtle lifestyle-related questions, the system places the user into a pre-defined demographic cluster. Web pages liked by the cluster are then recommended to the user. The system uses 62 clusters based on a commercial consumer marketing system. If a user falls into more than one cluster, then all matching clusters form the candidate set for recommendation. A user can refine this set by answering more questions. The data used to form demographic clusters is taken from user's home pages. WINNOW [44] is used to learn a classifier for users based on home-page data. In the evaluation, 40% of users liked the recommended pages, compared with 30% for random recommendations.

Advantages of this method of recommendation include:

- The system's adaptive quality improves over time by eliciting feedback from users in particular demographies;
- No specific domain knowledge is needed. Knowledge-based systems use detailed information about a particular domain. Demographic systems

operate based on broad knowledge of the "stereotype" that a user is categorised to;

- This technique can identify cross-genre niches.

Disadvantages of demographic filtering include:

- The system will suffer from the new item ramp-up problem;
- Users suffer from the "gray sheep" problem [19], where a user's esoteric tastes mean they do not fit well into any particular cluster and therefore lack overlap with other users' profiles;
- The quality of recommendations depends on a large historical dataset;
- Gathering of demographic data can be a costly task.

2.3.6 Hybrid Information Filtering Techniques

Most recommender systems use collaborative or content-based Filtering to predict new items of interest for a user. Both of these methods fail to provide good recommendations in many situations [15]. For example, a content-based recommender will only generate recommendations of items that have similar content to ones seen before, thereby ignoring any diversity in our tastes[115]. Hybrid recommenders have been forwarded as a solution to this, among other problems, [15]. For example, content-based filtering can make up for the initial poor performance of the collaborative technique, i.e: when there are not enough ratings in the dataset to provide decent overlap between user profiles, a content-based approach is applied to recommend items based on their textual descriptions.

A good example of a hybrid recommender is PTV [113] [21]. The introduction page to PTV contains an effective profiling system that records user-specific TV programme interests. The ACF component of the filtering system initially gives poor ratings since the collaborative matrix is sparse and overlap is low. During this "teething" phase Content-Based filtering is used to give recommendations. Once-off programmes cannot be picked up by the ACF system since there is not sufficient time for users to rate them. The

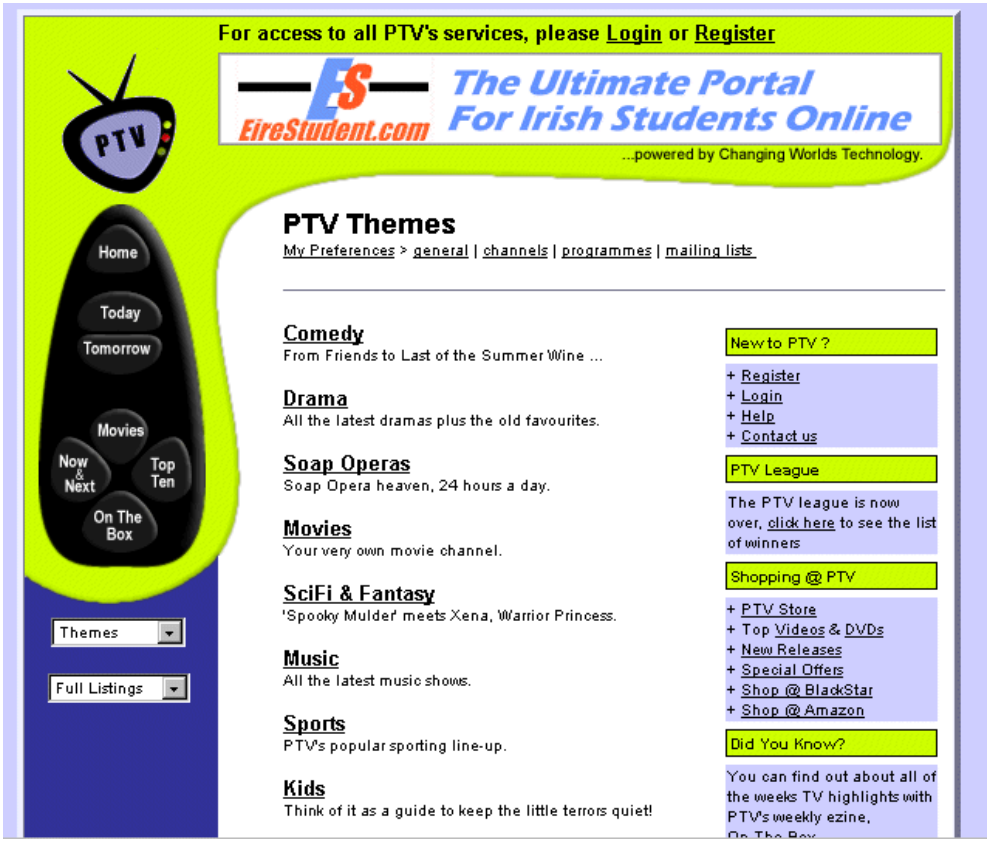

Figure 2.9: Screenshot of PTV, A Personalised Television Listings Service which uses Hybrid Recommendation Techniques.

content-based component may recommend these shows. Once the collaborative matrix starts to fill up, ACF overrides the content-based approach. A new approach to the integration of the above techniques is outlined in PTVplus [113]. In this approach, data mining techniques are used to extract association rules between TV programmes; these are then used to counter the overlap problem with traditional ACF. Metadata about programmes is "extended" by new techniques (eg "likes a" implies "likes b") and then used to enhance the content-based part of the system. Another hybrid system, INVAID [59], is an integration of content-based and collaborative recommendation. Experimental evidence that a content-collaborative hybrid performs better in a recommender system is demonstrated in Melville and Mooney's

Content-Boosted Collaborative Filtering [75], where the hybrid technique is evaluated in a movie recommender. Content-Based predictions are used to "pad out" the user-rating matrix in order that the ACF technique can provide better recommendations. They tested this hybrid idea against pure Content-Based, pure ACF and a naïve hybrid technique, using Receiver Operating Characteristic (ROC) and Mean Absolute Error (MAE) measurements. The naïve hybrid actually performed generally worse than the pure techniques, whereas the Content-Boosted hybrid outperformed every other technique. This research shows that the manner in which the Content-Collaborative hybrid is implemented is of paramount importance in a system. A "bad" hybrid is shown here to be worse than a pure technique. Work in [85] gives an account of AdRec, a recommender system which employs an adaptive approach to the recommendation task. Based on a lightweight classification of the dataset such as sparsity sampling, ratio of users to items and sampling for latency analysis the system selects the most appropriate from a set of three recommendation algorithms, which have been shown using regression analysis over test runs to be the best predictor for datasets with similar properties in the past.

Approaches to Hybridisation

There are a number of different approaches to hybrid recommender systems. Burke gives a good overview of these in [15], and Jameson et al. provide a very thorough review in [52]. The following are the main approaches to hybridisation that have been defined in the literature.

- Weighted: In this approach, the scores (votes) from several recommendation techniques are combined together to produce a single recommendation. For example, Burke's *Entree* System [15] uses a weighted combination of five recommendation methods.

- Switching: In the switching method, the system switches between recommendation techniques based on the current situation. For example, for items which have a rich textual description a content-based approach would be used, whereas conversely, a collaborative approach might be employed for non-machine-analysable items, [52].

- Mixed: A mixed approach is when recommendations from several different recommenders are presented at the same time. An example of this approach is Amazon's* Web pages.

- Feature Combination: In this approach, features from various recommendation data sources are thrown together into a single recommendation algorithm. For example Case-Based Recommendation [92] uses both item features and responses of other users to generate a recommendation for a user.

- Cascade: In a cascading hybrid system, one recommender refines the output given by another recommender. For example, Burke's *Entree* recommender [15] employs a content based approach, but uses a collaborative solution in a tie-break situation in the same way. c.f Weighted Hybrids. Melville and Mooney [75] augment their ACF algorithm with a content-based component.

- Feature Augmentation: In feature augmentation hybrid systems, the output from one recommendation technique is used as input for another. An example of the use of this technique is shown by Good et al. in [45], where the ACF sparsity problem is tackled by treating agents as users, and employing their item ratings in the recommendation process. One issue with this approach is that the agents are not providing real data, so the actual live user ratings get "diluted" by the agent ratings. It might be a feasible solution to employ a weighting mechanism to this technique whereby real user ratings get higher weights than agent-generated ones.

- Meta-Level: In this approach, the model learned by one recommender system is used as the input to another. An example of this is the "collaboration by content" in [15], where terms in a content-based profile are assigned weights, and these term-weights are passed to a collaborative recommender and used in place of item ratings.

*http://www.amazon.com

2.4 Case Studies

Before the discussion on trust based recommenders, short case-studies on four "traditional" recommendation systems are presented. To provide a good analysis of the progression and development of recommender systems over the years. the case studies presented here range in age from among the earliest (Tapestry, 1992)[42] to the most recent (StumbleUpon, 2006) discussed in Section2.11.

2.4.1 Tapestry

After John Hey's "likeminds" (the first company to file a patent on an ACF technique*) the first use of the term "collaborative filtering" was in the Tapestry system presented by Goldberg et al. in [42]. This was an email filtering system in which an ACF process was defined as a process in which "people collaborate to help one another perform filtering by recording their reactions to documents they read." Users of the system can annotate documents with keywords and phrases in order to share information on their preferences. Recommendations are made when the user issues a complex query on these words and phrases, and the system chooses items based on the annotations and on the identity of the annotator. One general query of this type is "show me documents that user z annotated as interesting" The system also allows for implicit user-feedback on queries. For example knowing that user a only sends email responses to documents he finds interesting, selection could be made on this basis.

On closer analysis, Tapestry was just an extended querying system, where users can benefit from the annotations of other users. There were a number of limitations to this system, however. The main drawback to the system was that users required some knowledge of or prior relationship with the advisor, or else his recommendation could not be trusted. Another problem with the system was that users were free to annotate items with any keywords they chose, thereby making the probability of two users using the same set of keywords very unlikely. Tapestry [42] was a fairly cumbersome ACF system,

*http://www2.sims.berkeley.edu/resources/collab/collab-archive/0036.html

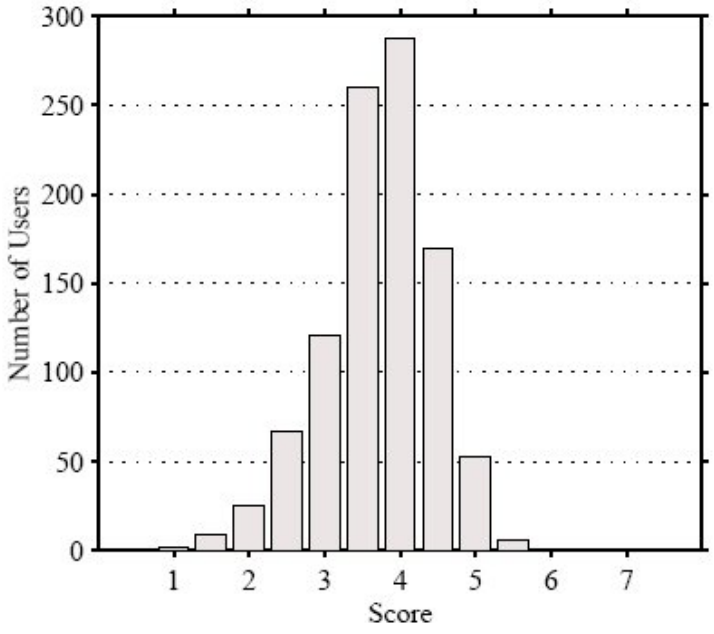

Figure 2.10: Ratings Distributions in RINGO.

but it laid the groundwork for the more advanced systems we have today. The other example systems discussed in this section do not rely on a prior relationship between users in order to recommend items.

2.4.2 Ringo

Shardanand and Mae's online music recommender RINGO [111] was a statistical ACF system. Users rated albums which they had listened to and the system provided them a list of albums it believed they would like. One interesting feature of this system was that users could append new items (albums) to the system's itemset. This resulted in a >500% increase in the itemset size in the first six weeks of operation.

Predictions were computed in this system by a computation of weighted averages of user-ratings. Weight intensity depends on similarity between the active user and the user whose ratings are being weighted. This is computed using one of three common similarity metrics: Mean Squared Differ-

ence (MSD), which is simply the average deviation in either direction; Pearson correlation, described in detail in Section 2.2.2; and constrained Pearson correlation. The latter metric attempts to use both positive and negative correlations between users. Ringo has a seven-point rating scale varying from "much liked" to "average" to "awful". Ratings above 4 indicate liked items, while ratings below 4 indicate disliked items. Due to the apparent Gaussian distribution of ratings [20], illustrated in Figure 2.10, this metric will only increase a similarity coefficient between two users if they have both rated an item either positively or negatively. This type of rating distribution is common across many collaborative filtering datasets. However, on our eBay dataset which we examine in Chapter 7 of this thesis, there is a very different distribution, with the vast majority of user ratings in the top 10% of the rating scale. Ringo has another prediction algorithm known as *artist-artist* correlation, which uses the correlations between items, calculating a weighted average of the active user's scores on similar items. This technique is not too dissimilar to the Item-Based ACF algorithm described in Section 2.2.3, by Fisk in [28], and Sarwar et al. in [107]. The results shown in [111] show poorest performance from the artist-artist technique, which is in stark contrast to the results in [28], [108] and [107].

2.4.3 StumbleUpon

StumbleUpon is one of the largest and most successful of Web page recommenders. It has become very successful due to the quality of recommendations it provides as well as its feature rich browser extension. StumbleUpon enables users to discover and share new web sites. During the registration process users specify their interests and a profile is built up. Pressing the "stumble" button on the firefox menu bar brings the user to a recommended site. StumbleUpon uses collaborative recommendation to deliver new sites to users. Users can easily rate the new site by clicking on a thumbs up or thumbs down icon on the menu bar. Currently, StumbleUpon has a user base of 3,498,737, with almost a 400% increase in one year. Trust information can be explicitly specified in the application by explicitly setting "friend" profiles from which you can receive recommendations. Figure 2.11 shows a screen-

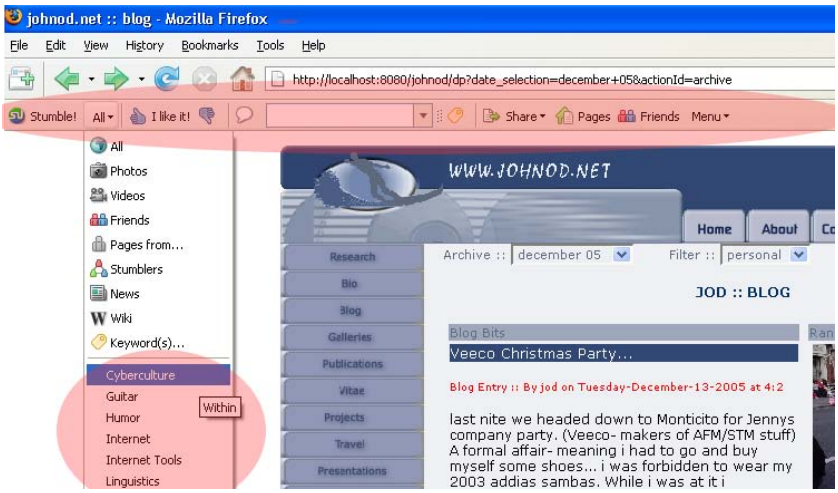

Figure 2.11: Screenshot of the StumbleUpon Firefox Add-on, Recommender components are shown in shaded area

shot of this application with the recommender functionality highlighted in red. The "stumble" button is shown in the top left corner. StumbleUpon recommends photos and videos as well as basic Web pages.

Collaborative feedback from community peers prevents the spread of poor content, yet quickly distributes compelling content due the "small-world" nature of social networks. This social moderation and distribution model is resistant to spam and offers users interactive web surfing within a community context. The combination of such social networking principles with interactive content discovery allows StumbleUpon to simplify web navigation and augment existing search systems. Figure 2.12 depicts a design diagram of the StumbleUpon system. The key feature of note in this diagram is the users social network. In a traditional recommender system a user's peer group would normally be selected using some similarity metric computed over all profiles in the database. Here, however, the user's peergroup is a combination of the standard collaborative filtering peers and an explicitly specified set of peers. Standard peers are computed using profile similarity metrics and explicit peers are specified directly by the user when they initialise their profile, and anytime afterwards. This enables the StumbleUpon system to overcome many of the problems inherent in traditional collaborative filter-

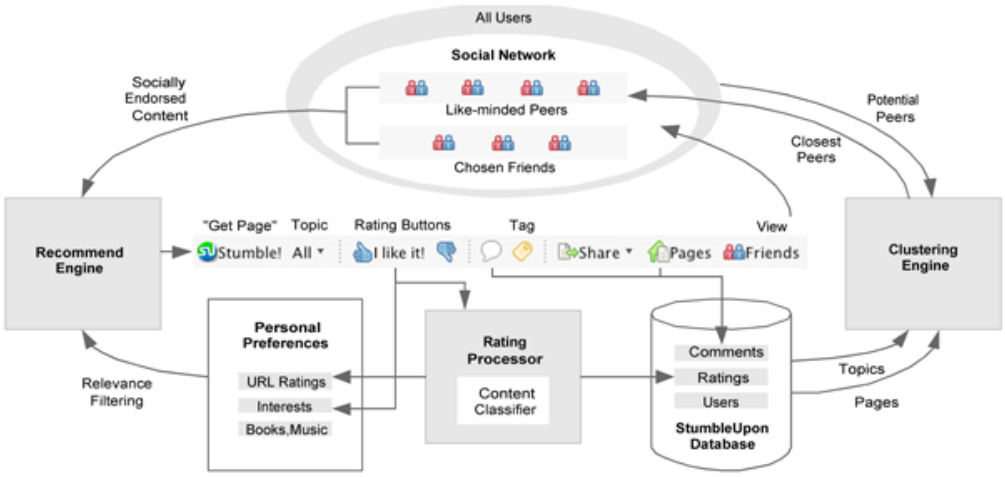

Figure 2.12: Operation Diagram of the StumbleUpon Recommender System

ing systems. A similar approach is adopted later in this thesis. In chapter 7, it is shown how our *PeerChooser* application enables explicit selection of peers in addition to the standard set of "similar" peers. Furthermore, our application allows for dynamic modification of the standard peergroup during recommendation- a feature which is not currently implemented in StumbleUpon.

2.4.4 MovieLens

The GroupLens Research Project has been carrying out research into ACF and recommender systems since 1992. The GroupLens system is one of the first automated collaborative filtering (ACF) systems. Work on this system has been extensively reported; see, for example, [107], [108], [45], [109] and [108].

The GroupLens ACF system was first introduced as a solution to the information overload problem in Usenet News, which was losing popularity due to the mass of irrelevant information it contained for any one user.

This system identifies peer groups based on the Pearson correlation (see Section 2.2.2 and [12]). Two users may not rate items identically, but if they have a high number of co-liked and co-disliked items then they will have a

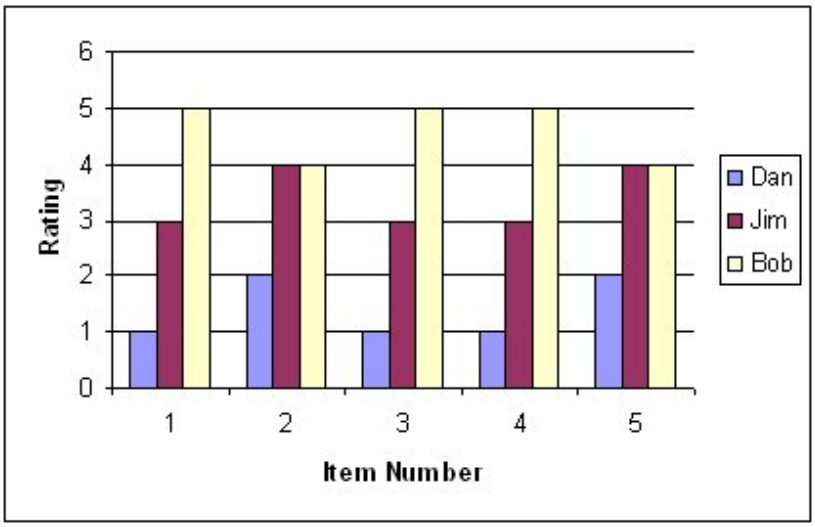

Figure 2.13: Rating Histories Example

	Dan	Jim	Bob
Dan	1	1	-1
Jim	-1	-1	-1
Bob	-1	-1	1

Table 2.3: Correlations generated from Ratings History in Figure 2.13

positive correlation value. Predictions are then generated using the deviation from the active user's mean rating.

Figure 2.13 depicts an example ratings history for three users and Table 2.3 shows the resulting correlations (Example taken from [18]).

The majority of experiments carried out in later chapters use the MovieLens* database. MovieLens, shown in Figure 2.14 was developed by John Riedl, Joseph Konstan and others at the university of Minnesota. MovieLens uses ACF to recommend movies from the Internet Movie Database(IMDB)†. The system has been running for over five years and has recently been redeveloped with the addition of many new features. In the old system, users

*www.movielens.umn.edu

†www.imdb.com

provided ratings on a scale of 1 (strong dislike) to 5 (strong like) and the system provided either specific recommendations or a top-n list. The improved system is feature-rich and allows collaborative tagging of movies, rating of individual tags, categorisation of users, recommendation to groups, and much more. The tagging and rating system forms a "folksonomy" (also known as "social classification" or "social indexing"), thus allowing collection of rich descriptions of content by which users can be profiled and matched. The GroupLens team recently published an award winning article on this feature of the MovieLens system [110]. This additional data will undoubtedly be used in future experiments by the GroupLens team. Currently the system gets about 800 people logging on per week. The highlighted area in Figure 2.15 shows an interesting new features in MovieLens. The *utility* of a particular set of ratings to a specific group within the system [78] is computed and displayed to the person making the ratings. In this example, the ratings are said to be most useful to the "comedy group". The following chapter of this thesis discusses a similar method to compute "utility" of sets of ratings on the MovieLens dataset, although we compute this "utility" not on a group level, but for individual users and show that the resulting utility (or "trust") models are highly useful for increasing robustness, accuracy and user-satisfaction in ACF systems.

2.5 Summary

Recommender systems help to cope with the problem of information overload. Automated Collaborative Filtering (ACF) is the most popular technique used in these systems. ACF generates recommendations by harnessing the opinions of users who are "similar" (based on some similarity function).

This background research chapter has focussed on ACF and the various other techniques and approaches used to generate recommendations. There are many different strategies used in recommendation, which vary based on domain characteristics such as product details, types of ratings and number of users, for example. The chapter has examined a broad range of approaches taken from both literature and operational systems, with a strong focus on ACF which is the core technique used in this work. The following list outlines

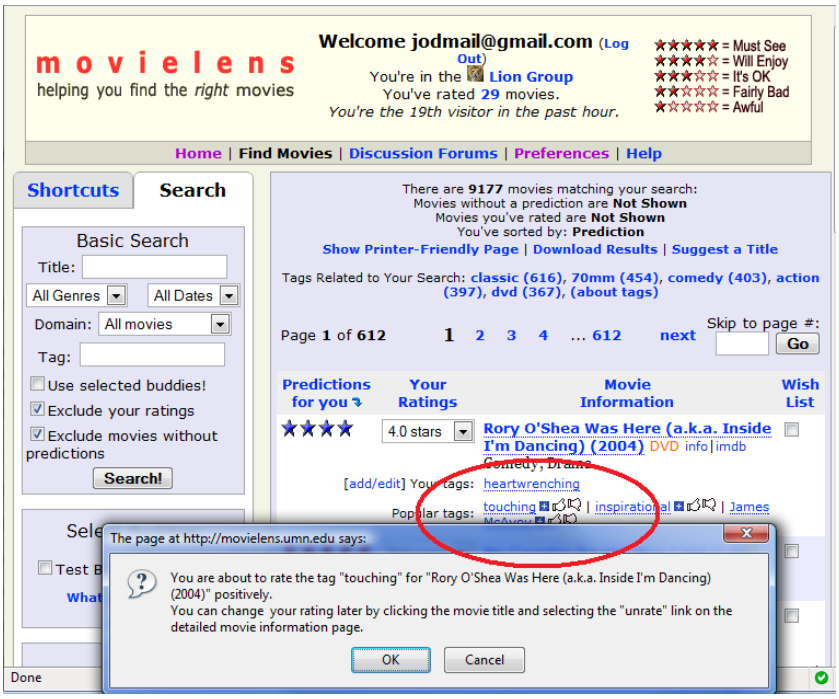

Figure 2.14: Screenshot of the Tagging and Rating System in the MovieLens Recommender. (The Red Circle Highlights the Tag-rating Controls.)

the core advantages of ACF over other techniques.

1. ACF techniques have been proven to be reasonably accurate.
2. ACF does not depend on (possibly erroneous) machine analysis of content.
3. Due to the above, ACF can successfully recommend non-machine analysable content such as movies and art.
4. Because it models the real-world social process of recommendation, ACF does not suffer from narrowness and can make serendipitous recommendations.
5. ACF can be integrated with other recommendation techniques (e.g: content-based techniques) to achieve better performance when rating data is sparse.

Figure 2.15: Explanation and Profiling Methods in the MovieLens Recommender System

Chapter 3 presents a discussion of the role of trust on the Social Web with a focus on several case-study recommender systems, followed by the first major contribution of this work: a novel approach for computing a model of trust within an ACF system.

Chapter 3

Trust in Recommender Systems

3.1 Introduction

Recommender systems have proven to be an important response to the information overload problem, by providing users with more proactive and personalised information services. And collaborative filtering techniques have proven to be an vital component of many such recommender systems as they facilitate the generation of high-quality recommendations by leveraging the preferences of communities of similar users. In this chapter we suggest that the traditional emphasis on user similarity during recommendation may be overstated. We argue that additional factors have an important role to play in guiding recommendation. Specifically we propose that the trustworthiness of users must be an important consideration. We present two computational models of trust and show how they can be readily incorporated into standard collaborative filtering frameworks in a variety of ways. We also show how these trust models can lead to improved predictive accuracy during recommendation.

We propose to modify the way that recommendation partners are generally selected or weighted during the recommendation process. Specifically, that in addition to profile-profile similarity—the standard basis for partner selection—we argue that the *trustworthiness* of a partner should also be considered. A recommendation partner may have similar ratings to a target user but they may not be a reliable predictor for a given item or set of items. For

example, when looking for movie recommendations we will often turn to our friends, on the basis that we have similar movie preferences overall. However, a particular friend may not be reliable when it comes to recommending a particular *genre* of movie. For example, my friend Bob is great at recommending comedy movies, but he is awful at recommending sci-fi movies. The point is, that partner similarity alone is not ideal. Thus far, the discussion of collaborative filtering in this thesis has been restricted to systems which rely only on traditional models of partner similarity for recommendation. Of course recommendation partners should have similar tastes and preferences, but they should be also *trustworthy* in the sense that they have a history of making reliable recommendations, since trust is considered more important than similarity when it comes to seeking recommendations in the real world.

A number of computational models of trust are proposed in this chapter which are based on the past rating behaviour of individual profiles. These models operate at the profile-level (average trust for the profile overall) and at the profile-item-level (average trust for a particular profile when it comes to making recommendations for a specific item). The methods for incorporating this trust information into the recommendation process are discussed and it is demonstrated that the method has a positive impact on recommendation quality, in terms of both openness of the system and accuracy.

3.2 Background

Before presenting the detail on our computational model of trust for recommender systems, a general overview of research into trust and reputation in the Social Web is presented here. This background compares and contrasts three recommender systems as case studies, all of which attempt to model trust in some way. The discussion focuses on modelling trust within Social Web applications, and on the potential uses of trust models.

As more and more users provide more of the content on the Web, new ways to assess their trust and reputation are constantly being developed, along with mechanisms to integrate this information into Web applications to provide better all-round performance. For the sake of discussion let us firstly categorise Trust in the Social Web into two general areas: Trust in

the Web application as whole v/s trust in the users of the application. For example, I might trust eBay to have a broad range of second-hand electronics available, and I might trust the processes on their site to function well, but I might not necessarily trust user *FlyByNight1969* to sell me a second-hand i-pod. Conversely, I would not trust the web site FriendFinder.com with my personal data because it will likely be used for spamming purposes, however, many of the users on this site have expressed explicit trust for one and other by getting married. A trust-based system for the Social Web must be capable of distinguishing between the various types of trust that exist therein.

Gaines [32] highlights a very relevant point about the migration of social interaction to the online world by asserting that society may be placing too much faith in the engineers who design and develop Social Web applications:

"Our adaption to the automated world that we have created requires engineering disciplines to formalise the social dimension of their activities as much as they have formalised the technological dimension. We can "trust" technology only to the extent that we can trust the engineering professions to accept the responsibility for this formalisation." [32]

This is an interesting point which has knock-on effects for this research into the trust of individual users within the application. As discussed throughout this thesis, human trust is a function of reputation, context and many other complex factors. If a user on the Social Web cannot trust the system as a whole, then surely this will adversely influence the trust a user has for the other users in the system. If Gaines' assertions are indeed correct, the problem of system trust will have to be addressed before we progress to analyse trust of the individual users in the system. The counter argument to Gaines' theory is that system trust is a "bottom-up" rather than "top-down" phenomenon and that system trust can be viewed as a function of the individual users trust within that system. Most likely the truth lies somewhere in between these extremes, and that system trust can have an effect on user trust. This argument warrants future research in its own right.

The second categorisation, trust in Social Web *users* commands a lot of research attention, [5] [4] [38] [39] [103] for example, and is the main focus of this thesis. Within this categorisation, a further trust division could be made by distinguishing between trust with respect to existing user content,

and trust with respect to future actions of users, for example a seller shipping a product on eBay. However, for this discussion they remain under the same heading. The need for trust values for Social Web users has already been discussed in Chapter 1. The manner in which they are achieved is under discussion here.

Across all the applications on the Social Web, there are countless ways in which user trust can be inferred, extracted, propagated or otherwise computed. There are generally two core steps for a trust-based system on the Social Web. Firstly, building a trust model and secondly, using that model to enhance system performance in some way. Before proceeding with the details of our trust modelling process for ACF systems, we need to gain a clearer picture of what a trust model for the Social Web should represent, what meaning is contained in it, and what it can be used to achieve. The following section outlines a controlled experiment which examines the different ways trust can vary with changes in context. This helps to provide a general understanding of the concept which we attempt to model later in this chapter.

3.2.1 Trust in Context: The Nintendo Test

Trust is a highly subjective variable which depends intricately on a range of inputs. This concept is realised in some manner or other in all of the major research works that deal with trust. For example Ramchurn et al [100] describe trust for an individual as the product of a reasoning process combining facets from socio-cognitive models, reputation models, and evolutionary models. This means that trust is highly dependent on its *context*. Marsh [69], Josang [55], Luhmann [68] and most other popular works on the subject conform to the idea that trust is highly contextual.

As a precursor to the trust based systems researched and developed later in this thesis, a relatively simple experiment was designed and implemented to illustrate contextual variance in trust. The idea behind the experiment is to illustrate the fact that users will *lower* their trust thresholds in the face of adverse conditions. To put it another way: people will trust where they usually would not in adverse conditions. To test this concept and calibrate

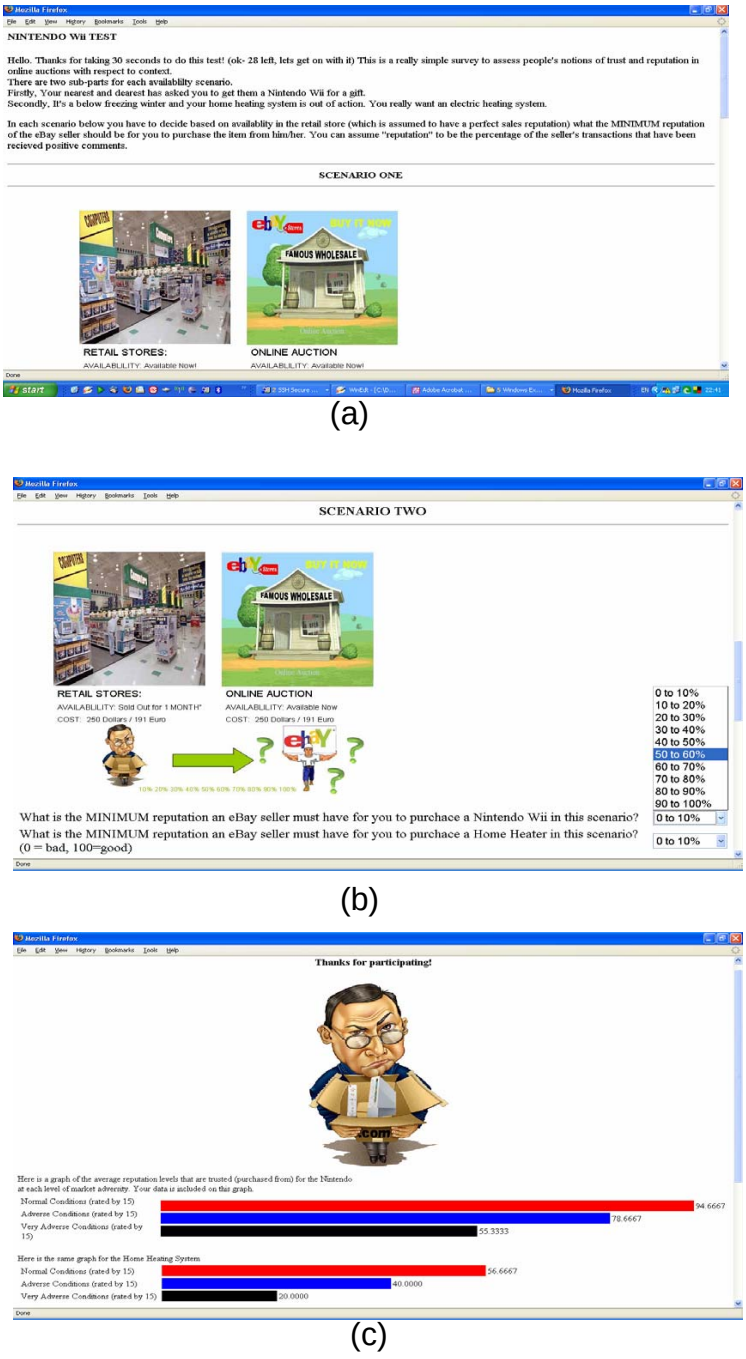

Figure 3.1: Screenshots of the Nintendo Test (a) Introductory Text. (b) Scenario One (c) Dynamic Result Charts and Conclusion Screen. (Result charts in (c) show the variance of trust with respect to reputation on a scale of increasingly adverse conditions.)

Scenario	Retail Stores		Online Auction	
	Cost	**Availability**	Cost	**Availability**
1	250	**Available Now**	250	**Available Now**
2	250	**Sold Out for 1 Month**	250	**Available Now**
3	none	**Sold Out Forever**	250	**Available Now**

Table 3.1: Range of Availability Scenarios Presented to Users in the Nintendo Test

the rate of change in trust with respect to reputation (performance history) of the trustee to some extent, the following test was developed.

One hundred users were shown sample auction scenarios on the web site shown in Figures 3.2.1(a)(b) and (c). They are told in the introduction to assume that they really would like to purchase a luxury item: a "Nintendo Wii". Users must also assume that they need to purchase a crucial item: a "home heater", during a freezing winter. Both items are shown for sale in retail stores and online auctions for the same overall price. Users were then presented with three different availability scenarios and asked to decide for each what the *minimum* reputation of a seller in an online auction should be for the user to make the purchase from the online auction. Users must provide the minimum reputation the seller must have for the purchase to take place of both the crucial and non-crucial items, using the scenarios outlined in Table 3.2.1.

An image of a store is presented showing the required items available in a trusted retail store, along with the item for the same price on eBay. The second scenario is similar except the items in the store are sold out for one month. In the third scenario it is sold out forever, as listed in Table 3.2.1.

Participants selected the *minimum reputation score* that the eBay seller must have for them to purchase each item individually on a percentage scale from a drop-down menu, seen in Figure 3.1(b).

This experiment simulates increasingly adverse market conditions and requests users to directly specify the minimum reputation that they would trust at each level of adversity. Figure 3.2 shows the results of this survey of one hundred users. The ratings given by each user show a smooth, correlated

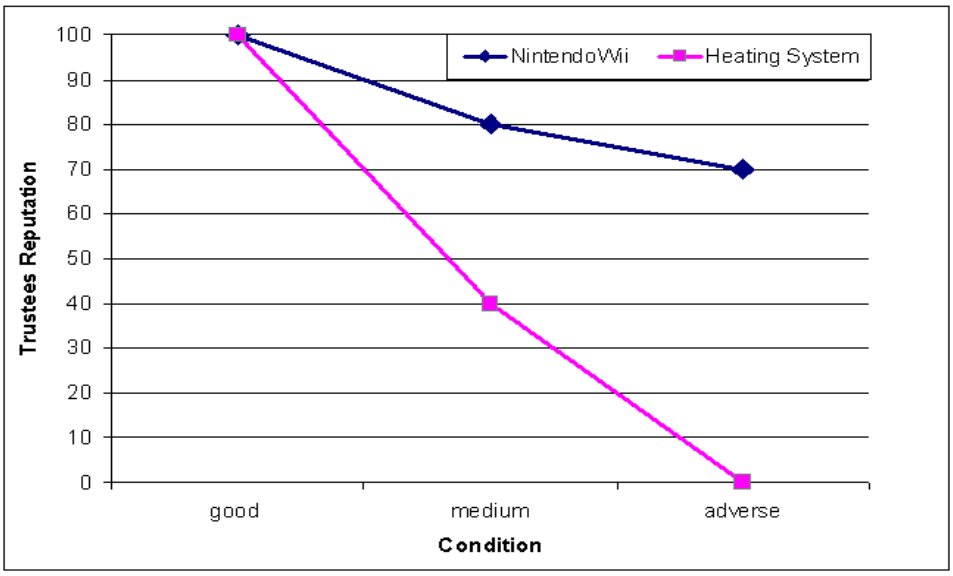

Figure 3.2: Results of the "Nintendo Wii" experiment showing direct correlation between minimum trusted reputation and adversity in the online market.

reduction in the "action threshold" or "trust" with respect to the trustees reputation, (defined to the users as the percentage of successful past transactions). This correlated reduction occurs as the severity/adversity of the situation is increased.

The correlation occurs for both the crucial and non-crucial items, however for the crucial item the correlation between trust in bad reputations and adversity is notably stronger (Figure 3.2). For this experiment, crucial and non-crucial items were chosen in the adversity simulation following from a detailed discussion by Gaines in [32] of psychologist Abraham Maslow's 1971 hierarchy of human needs, and its application to the online world. Detail on this study is presented as an appendix to this thesis. Being an entertainment device Nintendo Wii is categorised higher in Maslow's pyramid than the heating system, which can be categorised as a basic human need in cold temperature. The results in Figure 3.2 show that on average, the adverse condition a buyer will "trust" a seller with a reputation 70% less for the heating system- a basic need, than for the Nintendo (ie entertainment) which is categorised in the third rung of Maslow's pyramid. The seller of the Nintendo

is trusted with a reputation 30% less than normal under the adverse condition. Normally, i.e: under "good" conditions, the minimum seller reputation that is trusted by the buyer is at its highest and is the *same trust* for both the crucial item and the non-crucial one.

The "Nintendo experiment" clearly shows that trust is a very subjective phenomenon, and is highly affected by the context in which it must be applied. Later in this work, techniques to model and use trust are presented in recommender systems and in an online auction. This experiment has shown that the context *must* be considered in these models. Chapters 4 to 6 of this thesis explain how context is modelled and represented in both of the trust-based solutions that are developed.

3.2.2 Trust and Reputation in the Social Web

Of late there has been a lot of research attention towards issues of trust and reputation on the Social Web. [5] [70] [103] [123] [123] [57]. Looking at the online auction eBay as an example, Resnick [103] carried out a comprehensive analysis of eBay's trust and reputation system. This survey highlights some interesting findings regarding the nature of trust within the online auction, and the behaviour of its users. Five of Resnick's more salient findings from his work in [103] are listed below (with parenthesised addenda):

1. Despite incentives to free ride, feedback was provided more than half the time. (If people are generally willing to provide feedback most of the time, this means there will be raw data for a trust-modelling system to operate on.)

2. Well beyond reasonable expectation, the feedback was almost always positive. (see 7.2.1 in this thesis for a comparative survey. These 'false positives' in raw comment data may effect a trust value between users. Chapter 7 shows our method for compensating for unnaturally high positivity on eBay.)

3. Reputation profiles were predictive of future performance. However, the net feedback scores that eBay displays encourages pollyanna assessments of reputations, and is far from the best predictor available.

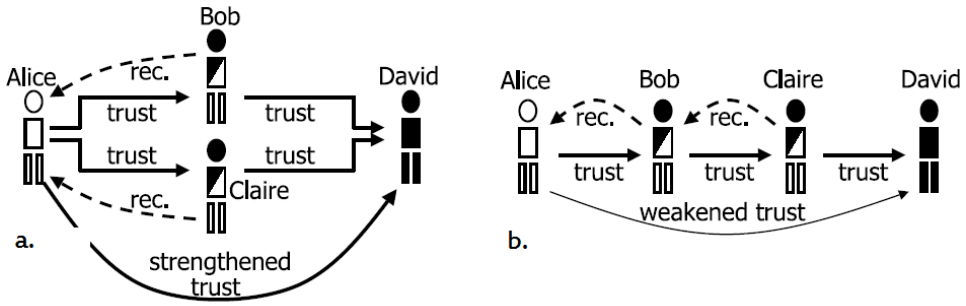

Figure 3.3: Trust transitivity diagrams from Andun Josang's work in [56]. Part (a.) shows recommendation from multiple sources and resultant parallel trust propagation. Part (b.) shows basic recommendation and trust propagation across a chain of users.

4. Although sellers with better reputations were more likely to sell their items, they enjoyed no boost in price, at least for the two sets of items that we examined.

5. There was a high correlation between buyer and seller feedback, suggesting that the players reciprocate and retaliate. (It would be interesting to analyse the variance in feedback correlation between the current mechanism and a 'blind' feedback mechanism wherein a user knows the commentee will never see the feedback)

A similar survey to Resnick's was carried out in 2006 by Josang et al. in [54]. In this work, Josang describes the eBay reputation system as a *collaborative sanctioning system* which naturally provides incentive for good behaviour of individuals thereby increasing performance of the market as a whole. Other relevant work by Josang et al. focuses on the transitive property of trust in online applications in [57] and [56]. In their approach, a diverse set of dimensions was constructed to represent trust, and a mathematical notation was defined to manipulate trust values based on this simplification. The dimensions used in [56] are *trust origin*, *trust target* and *trust purpose*. Represented according to these dimensions, Josang et al. find that trust does have some properties of transitivity, based on their study of some basic trust topologies (Figure 3.3). This is very relevant to the ideas

presented in Chapter 6 of this thesis, where a trust propagation system for eBay is proposed, and a diverse set of dimensions have been defined for the system.

Figure 3.3 shows two of the core ideas from Josangs work in [55]. Part (*a.*) depicts Josang's notion of parallel trust combination. In this figure, Alice can assess trust for a third party (David) by making decisions based on input from multiple sources (Bob and Claire). In this example, if Bob and Claire's recommended trust for David was highly correlated, it would stand to reinforce the derived trust that Alice has for David. Conversely, if Bob and Claire's recommendations about David were highly uncorrelated, it should lower the confidence level placed in the derived trust that Alice has for David. Josang deals in detail with various approaches to combine uncorrelated trust expressions of this form. An interesting perspective that was not discussed in [56] is that when Bob and Claire have uncorrelated trust statements for Alice about David, this tells us valuable information about Bob and Claire as recommenders. For example, if it manifests that David is highly trustworthy, and Bob has cautioned Alice that he is not trustworthy, then this misinformation should be remembered the next time Alice receives a recommendation from Bob. Essentially, the mistake should decrease Bobs overall trustworthiness as a recommender.

Figure 3.3 part (*b.*) shows the most basic form of trust transitivity defined in [56]. In this example, "recommendations" are propagated from Claire, (who directly trusts the target David) through Bob to the source, Alice. As in a real world social or business situation, this results in a weaker trust (less confidence) than if Alice interacted with David directly. (Note: In this example we use the term "recommendations" to represent statements of the form "I recommend that you should trust this person". This is distinct from the general usage of the term recommendations elsewhere in this thesis, where it generally applies to products or services.)

Research in [121] and [123] by Ziegler details the implementation of a propagation model for trust among users in the Social Web. Ziegler focused largely on use of semantic knowledge to achieve trust metrics. A classification scheme for trust metrics is developed in [121], and the benefits and drawbacks of this system, are analysed in the context of several Semantic

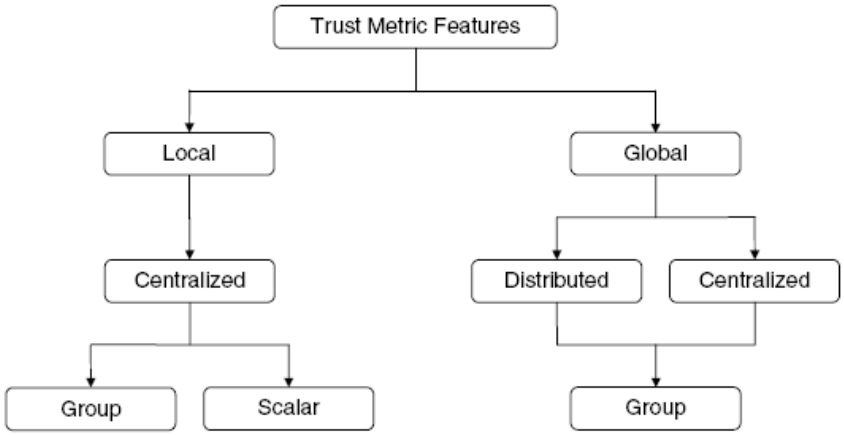

Figure 3.4: Local and global trust metric classification defined by Ziegler in [121]

Web scenarios. Ziegler introduces the *Appleseed* system in [121] which is enabled to compute trust for local groups by borrowing ideas from spreading activation models in neuropsychology. Figure 3.4 shows the classifcation scheme for trust metrics adopted by Ziegler. According to Ziegler, trust metrics can be broadly categorised into *local* and *global*, where global metrics consider all peers/users and all the links between them. Global trust ranks are computed on the full trust graph, in a manner analogous to the Google PageRank algorithm. Local trust metrics on the other hand, only use partial trust graphs, and constitute *personalised* trust, based on the rationale that the set of persons trusted highly by agent a may be different from those trusted by agent b. Ziegler's second categorisation for trust metrics (shown in Figure 3.4 as centeralised/distributed) refers to the place where trust relationships are evaluated. Local trust metrics are generally centralised because the trust relation between one Social Web user and others can be computed on one machine. Ziegler asserts that trust metrics which are distributed (i.e: computed on multiple machines over a network) are inherently global since a node would need to store trust information about *any* other node. The final axis shown in Figure 3.4 is that of link evaluation. Leivens 2003 work [65], posits that scalar metrics analyse trust assertions independently, while

group trust metrics evaluate groups of assertions in tandem. This essentially prohibits metrics in Ziegler's global classification from being computed by scalar metrics. An example of global trust computation is PageRank, since the computed rank of pages is dependant on the rank of associated pages. In general, scalar metrics compute trust between two given individuals a and b taken from set V of all agents, whereas group metrics generally compute trust based on *sets* of users in V.

Another research on evaluating and using trust for users in the Social/Semantic Web is that of Massa [5], in which a skiing application is developed which operates based on propagated trust values. This application is discussed in detail later in this chapter. Golbeck introduces a similar approach to computation and propagation of trust values in [38], this time in a movie recommendation system. Details of Massa's and Golbeck's systems are discussed later in this chapter as case-studies.

The issue of trust has been gaining an increasing amount of attention in a number of research communities and of course there are many different views of how to measure and use trust. The algorithms and applications presented later in this thesis maintain a computational view of trust which is constructed from varying types of user rating information. This concludes our analysis of current trust research on the Social Web for now, case-studies are presented under the heading of recommender systems in Section 2.4

3.2.3 Robust Recommendations

One area where trust is likely to play a critical role is in Recommender Systems, specifically with respect to their robustness. Recommender systems are being deployed in increasing numbers of commercial applications. This has motivated recent research into the robustness and security of recommender systems. Work in [63] by Lam and Riedl examines four open questions concerning attacks on recommenders: what is the recommendation algorithm being used; how detectable is the attack and what are the properties of the attacked item. The study also looks at questions such as the attack intent, the targets, required knowledge and cost of attack on different collaborative filtering algorithms, for instance the user based algorithm [102] and the

item-based algorithm [107]. They define a *shilling attack* as malicious user or set of users attempting to change the behaviour of the system to suit their own needs. Lam and Riedl conclude in [63] that current techniques for detecting and evaluating shilling attacks in recommender systems is in need of improvement, that shilling attacks are effective and easy to carry out, and that an attacker only needs a small amount of information about the system in order to perform a very successful attack. Lam and Riedl also point out in [63] that malicious attacks are non-trivial to detect with typical measures of recommender system performance.

Burke's Study on Attacking CF

Another comprehensive study on attacks in collaborative filtering was performed by Burke and Mobasher in [15] and [79]. This work is especially relevant to the consumer selection techniques introduced later in this thesis. [79] outlines six attack types. For instance, a *sampling attack* in which the attack profiles are sampled from real user profiles in the database. A *bandwagon attack* which inserts extremely popular items into the attacking profiles in the hope of maximising overlap between the attacking and the genuine profiles. If the attacking profiles have rated the same items as other users then they will be used more frequently in the recommendation process, and potentially have their input weighted more heavily if the system uses a technique such as the standard Resnick prediction formula [102]. Another attack strategy outlined in [15] is termed the *favourite item attack*. This attack does not target the system as a whole, it is aimed only at one specific user. In this attack, it is assumed that the attacker has some prior knowledge of a user's tastes, and only puts one specific user's preferred items into the attacking profiles. In [79] and [15], Burke and Mobasher evaluate the effectiveness of each of the attack strategies on an item-based collaborative filtering algorithm according to two metrics. The difference in rating for an attacked item before and after an attack (called *prediction shift*) and the percentage of times a targeted item gets into the top-n recommendations for a target user. This is termed *hit ratio*. For the more common user-based collaborative filtering algorithm however, analysed in a similar manner by

Burke in [15], there is only an evaluation of prediction shift for the targeted item. In our evaluation section we look at both prediction shift and accuracy for an attacked item as we feel that although an attacker may shift the predicted rating for an item, that shift may actually be in a direction which is favoured by the user receiving the recommendation.

O'Mahony's Study on Attacking CF

There is a close relation between the concept of trust and that of robustness within the context of recommender systems. Robustness can be defined as the degree to which the recommender system can perform normally under adverse (attack) conditions. Trust is a tool which can be used to increase the robustness of recommenders by helping to identify potential sources of attack. Ongoing research by O'Mahony et al. [89][90] examines the robustness of collaborative filtering algorithms under attack. O'Mahony et al. [90] formalises two aspects to robustness in collaborative filters, recommendation *accuracy* which determines whether or not the products recommended after an attack are actually liked, and recommendation *stability* which examines whether the system recommends different items after an attack. Work in [90] also defines several types of attack. For instance, a *push* attack, where one particular item is targeted for promotion by the attacker and a *nuke* attack, where one item is targeted for demotion. In their evaluation O'Mahony et al examine accuracy with respect to attack size using an absolute error metric. A similar metric is used in the evaluation of accuracy later in this work.

This concludes the background research on robustness of collaborative filtering algorithms. In Chapter 6, a trust-based strategy for combatting the effects of attacks on collaborative filtering systems is proposed, implemented and tested. The following sections provide a discussion on the potential ways in which recommender systems can begin to use trust to enhance the user experience through explanation, transparency and better accuracy.

3.2.4 Towards Trust-Based Recommendation in the Social Web

So far in our discussion of Recommender Systems we have examined many of the different approaches to the recommendation task and looked at case studies of some existing systems that use traditional forms of collaborative filtering. The focus of this thesis is on trust and reputation within Social Web applications such as recommenders and auctions. This section examines current state of research on recommender systems which attempt to harness and use trust and reputation information in some way.

A Trust-Based Semantic-Web System

Research by Golbeck in [37] describes an algorithm for generating locally calculated reputation ratings from a semantic web social network. This work describes TrustMail, an email rating application. TrustMail harnesses trust information from its users by explicit request, and propagates this information around the network. The goal of the application is to filter spam emails by use of trust and reputation information. The concept is simple: users rate everyone they know on a scale from one to ten, and the application will calculate a reputation score for each and every incoming mail message (using a local trust metric algorithm and some FOAF information). Users can then define the reputation threshold which filters mail as "spam". The advantage of this mail filtering technique is that it does not rely on text analysis, since most spam email attempts to trick the spam filter using some form of text.

Trust scores in this system are calculated through inference and propagation, of the form $(A \Rightarrow B \Rightarrow C) \Rightarrow (A \Rightarrow C)$, where A, B and C are users with interpersonal trust scores. The TrustMail application [37] looks up an email sender in the reputation / trust network, and provides an inline rating for each mail. These trust values can tell a user if a mail is important or unimportant. Trust values in this system can be defined with respect to a certain topic, or on a general level, as discussed in sections 3.6 and 3.7 of this chapter. A big limitation of the work in [37] and [71] is that they require some explicit trust ratings in order to infer further trust rating, although it may be possible to change this by using an existing source of trusted senders

such as a "clean" inbox or an address book. Experimental evidence is presented in [37] which shows that untrustworthy nodes in the trust propagation network cause rating accuracy to drop drastically.

Existing Trust-Based Recommenders

More directly related to the work in this chapter are a number of recent research efforts that focus on the use of trust and reputation models during the recommendation process. In our research we are interested in automatically inferring trust relationships from ratings based data, and using these relationships to influence the recommendation process. Recently a number of researchers have tackled a related issue by using more directly available trust relationships.

For example, Massa and Avesani in [71] build a trust model directly from trust data provided by users as part of the popular *Epinions.com* service. *Epinions.com* is a web site that allows users to review various items (cars, books, music, etc.). In addition they can assign a trust rating to reviewers based on the degree to which they have found them to be helpful and reliable in the past. They argue that this trust data can be extracted and used as part of the recommendation process, especially as a means to relieve the sparsity problem that has hampered traditional collaborative filtering techniques. The sparsity problem refers to the fact that on average two users are unlikely to have rated many of the same items, which means that it will be difficult to calculate their degree of similarity and so limits the range of recommendation partners that can participate in a typical recommendation session. Massa and Avesani argue that it is possible to compare users according to their degree of connectedness in the trust-graph encoded by *Epinions.com*. The basic idea is to measure the distance between two users in terms of the number of arcs connecting the users in the trust-graph encoded by the *Epinions.com* trust data. They show that it is possible to compare far more users according to this method than by conventional forms of ratings similarity and argue that because of this trust-based comparisons facilitate the identification of more comprehensive communities of recommendation partners. However, it must be pointed out that while the research data presented

does demonstrate that the trust data makes it possible to compare far more users to each other it has not been shown that this method of comparison maintains recommendation accuracy.

Similar work on the *Epinions.com* data in [70] introduces a trust-aware recommendation architecture which again relies on a web of trust for defining a value for how much a user can trust every other user in the system. This system is successful in lowering the mean error on predictive accuracy for cold start users, i.e: user who have not rated sufficiently many items for the standard CF techniques to generate accurate predictions. Trust data is used to increase the overlap between user profiles in the system, and therefore the number of comparable users. Work in [70] also defines a trade-off situation between recommendation coverage and accuracy in the system. This work however lacks an empirical comparison between a standard CF technique, such as Resnick's user based algorithm, and the trust-based technique. Future work in [70] mentions utilising both *local* and *global* trust metrics, which we discuss later in this chapter in the form of Item and Profile level trust. The same research group are involved in *Moleskiing*, [70], a trust-aware decentralised ski recommender, which uses trust propagation in a similar manner, and is detailed as a case study in a following section.

Montaner et al. [82] contemplates the availability of large numbers of virtual recommendation agents as part of a distributed agent-based recommender paradigm. Their main innovation is to consider other agents as personal entities which are more or less reliable or trustworthy and, crucially, that trust values can be computed by pairs of agents on the basis of a conversational exchange in which one agent solicits the opinions of an other with respect to a set of items. Each agent can then infer a trust value based on the similarity between its own opinions and the opinions of the other. Thus this more emphasises a degree of proactiveness in that agents actively seek out others in order to build their trust model which is then used in the *opinion-based* recommendation model. This approach is advantageous from a hybrid recommender perspective, in that agents can represent individual techniques and they can be combined using opinions based on trust.

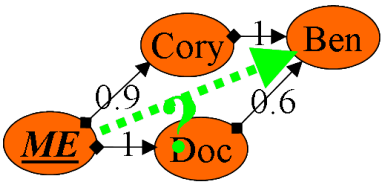

Figure 3.5: Propagation of Trust Inside the Moleskiing Recommender System.

Case Study: Moleskiing

Moleskiing is a trust-based online ski recommender designed by Massa et al. [5] [4] [70] [71]. Figure 3.6 shows a screenshot of this web application. Moleskiing was conceived to solve a real problem involving ski mountaineers, that being to make skiing safer by propagating reliable information regarding ski terrain. Massa argues in [5] that use of a "local" trust metric can be effective in improving the performance of the recommender engine.

The moleskiing.it application is a blog-based architecture which collects user experiences on ski mountaineering and user opinions on other users. Massa highlights the difference between "authority" and "trust", where trustworthiness is a "user cantered notion that requires the computation of personalised metrics." This is a similar notion to the analogy with the PageRank algorithm discussed in the opening chapter of this thesis. Trust, by its nature is subjective and personalised, that is, it is a value tailored to a particular individual. Reputation, in a computational sense, is objective in the Social Web because it is a function of every transaction that user has been involved in.

In the real world, trust decisions are generally based in some way on the objective concept of reputation, although there are usually other complex human influences. The "trust" value currently presented on moleskiing, (and eBay) however, is simply measure of a users *reputation*, since it is merely an average of binary ratings over each transaction that the user has been involved in. Therefore the value presented is *not subjective* since it is the same value for every user in the system. A more close approximation to real-world human trust might be to introduce *personalised* trust scores to

the application, which are somehow based on the individual profile that has to do the trusting. In this manner, an element of subjectivity could be introduced.

Information within the moleskiing application is maintained in an open architecture wherein all community information is openly published in Semantic Web formats. Massa's motivation in [5] for publishing this data is to enable different models to be tested on the same data, i.e to take a step towards standardisation of trust research methods. Later in this thesis we argue that trust models are *portable* across different applications. Moleskiing uses the FOAF (Friend-Of-A-Friend) format to publish its trust statements, meaning that the framework exists to incorporate trust statements from anywhere on the Web. It would be an interesting empirical experiment to test if trust information could be taken from the moleskiing application and used to provide some benefit in a different application (or vice-versa), possibly even in various semantic web formats, for example the XML Resource Description Framework.

Moleskiing delivers personalised trust information by using propagation techniques and assuming some transitive properties of the trust values. For example, Figure 3.5 (from [4]) shows the existing trust statements on a simple graph with nodes representing people and edge values representing trust between the nodes. "Local" trust values are computed by limiting the number of hops in this graph to 4. Massa argues that this technique is not only personalised, but much more attack-resistant than a global PageRank-like computation which shows what the community as a whole thinks about one particular user.

Although [5] and [4] deal exclusively with the Moleskiing application, they do not contain an empirical evaluation to test the effects of using trust in the recommender system, making it difficult to compare the system with those presented in this work.

Case Study: FilmTrust

FilmTrust is another recently conceived trust based recommender system which uses Semantic Web-based social networks augmented with trust in-

Figure 3.6: Screenshot of the Moleskiing Trust-Based Ski Recommender

formation to generate its recommendations. Users log on to the system and provide profile information explicitly by manually rating the American top 50 movies list. This technique for ratings elicitation may prove slightly misleading since this list contains only movies which people generally like. Chances are that this will lead to a "top-heavy" rating distribution, and not the expected normal distribution show in Figure 2.10 for the Ringo recommender system.

Once profile information is set up in FilmTrust, users can add other users as trusted friends and directly specify a trust value for that friend (shown in Figure 3.7). In this manner the FilmTrust application builds a rich database of social trust information albeit slightly burdening for new users. Once the profile is set up, users receive recommendations on movies, which appear to be of good quality. The application has an interesting feature where a user can visualise their location in the social network graph generated by the

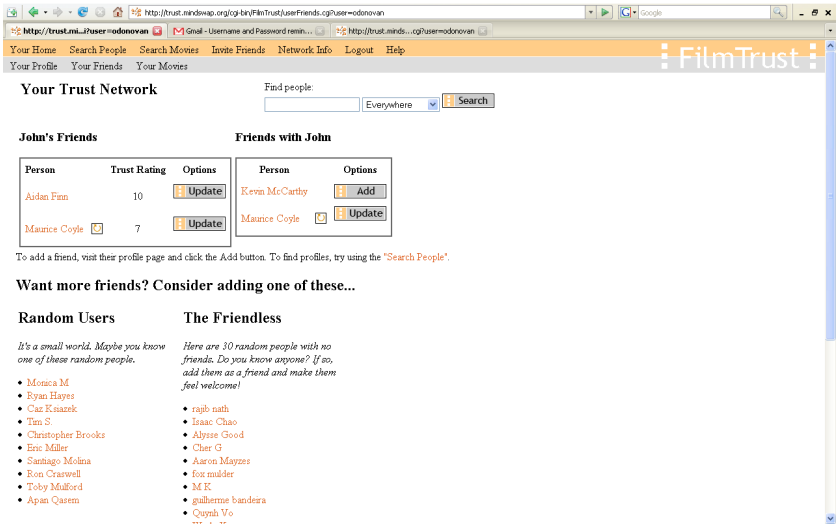

Figure 3.7: Screenshot of FilmTrust Friend Information

system. Figure 3.8 shows an example of this social network graph.

Golbeck and Hendler show in [38] that trust-based recommendations are more accurate than standard collaborative filtering recommendations in cases that were tested. The trust inference algorithm that drives the FilmTrust application is called TidalTrust. This algorithm assumes a basic property of transitivity of trust values. A user study is presented in [38] in which all users proved to be more satisfied with recommendations that were ordered by trust information than with those recommendations not ordered by trust.

More recent work on the FilmTrust application shows that trust based recommendations are much more accurate when the user receiving a recommendation has an opinion which is divergent from the average opinion.

As with the Moleskiing application mentioned previously [5], trust information in the FilmTrust application [36] is maintained in FOAF format, meaning that the information is highly portable and can be used easily in other applications. One such application which harnesses the social trust information from FilmTrust is a very new Web-browsing aide called Social-Browsing. [40] The following section gives a short overview of this new system.

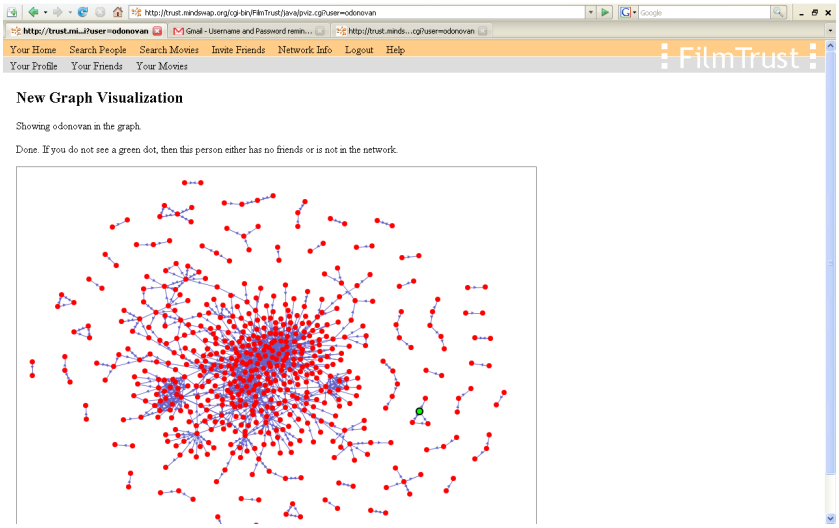

Figure 3.8: Visualisation of the Trust Graph of 1100 FilmTrust users.

SocialBrowsing

SocialBrowsing is a novel trust-based application, presented by Golbeck in her 2007 work [40]. SocialBrowsing has a Firefox extension that adds social context to the Web browsing experience. What makes this application interesting and relevant to this thesis is the manner in which the social, or "trust" information is ascertained by the application. SocialBrowsing harnesses trust information from other applications by its ability to read information from FOAF descriptor files. Applications such as FilmTrust store their trust information in this format, meaning that all trust information is easily sharable between the two applications. SocialBrowsing also reads trust information in FOAF format from other applications. Figure 3.9 shows a screenshot of this application. The tool-tip box in the centre displays information about the Web content that is socially relevant to the active user. This work is introduced in [40] and although interesting, does not contain any empirical evaluation or user surveys to test this application. However, the idea that trust information can be shared across applications is very relevant to this work. A discussion is presented in Chapter 7 on the mechanisms for building a trust model on data from one online auction and using to generate successful recommendations on another.

Figure 3.9: Screenshot of the SocialBrowsing Extension for Firefox.

This concludes our background into the state of the art in trust-based recommendation. Now we introduce our first computational model for trust in recommender systems.

.

3.3 A Computational Model of Trust for Collaborative Recommender Systems

To date collaborative filtering systems have relied heavily on what might be termed the *similarity assumption*: that similar profiles (similar in terms of their ratings histories) make good recommendation partners. However, in the real world, a person seeking a recommendation would look further than just similarity with a potential recommender. For example, my friend Bob is a mechanic and is quite similar to me in many respects, but I happen to know from past experience that he has a limited knowledge of computer science, and because of this I would never seek a recommendation from him on anything related to computer science. However, I would trust him highly to recommend anything related to cars, as he has proven to be highly competent in this area in the past. We believe that similarity alone is not enough to base recommendations on in the online world. Trust is an important factor in deciding who to receive recommendations from in Social Web systems as

well as in real world situations. Following is a description of the design, implementation and evaluation of a novel technique for modelling trust in ACF recommender systems.

A distinction is drawn between two types of profiles in the context of a given recommendation session or rating prediction. The *consumer* refers to the profile receiving the item rating, whereas the *producer* refers to the profile that has been selected as a recommendation partner for the consumer and that is participating in the recommendation session. So, to generate a predicted rating for item i for some consumer c, we will typically draw on the services of a number of producer profiles, combining their individual recommendations according to some suitable function, such as Resnick's formula as discussed in Chapter 2, for example. (see Equation 3.1 in this chapter)

Our benchmark algorithm uses Resnick's standard prediction formula which operates by computing a weighted average of deviations from each neighbour's mean rating, and which is reproduced below as Equation 3.1; see also [102]. In this formula $c(i)$ is the rating to be predicted for item i in consumer profile c and $p(i)$ is the rating for item i by a producer profile p who has rated i, $P(i)$. In addition, $\overline{c}$ and $\overline{p}$ refers to the mean ratings for c and p respectively. The weighting factor $sim(c,p)$ is a measure of the *similarity* between profiles c and p, which is traditionally calculated as Pearson's correlation coefficient, which has been discussed previously in Section 2.1.

$$c(i) = \overline{c} + \frac{\sum_{p \epsilon P(i)} (p(i) - \overline{p}) sim(c,p)}{\sum_{p \epsilon P_i} |sim(c,p)|} \tag{3.1}$$

Item predictions are generated using benchmark correlation and prediction methods as it allows for ease of comparison with existing systems. As we have seen above Resnick's prediction formula discounts the contribution of a partner's prediction according to its degree of similarity with the target user so that more similar partners have a larger impact on the final ratings prediction.

The key to our technique is an assessment of the reliability of such partners profiles to deliver accurate recommendations in the future. This is

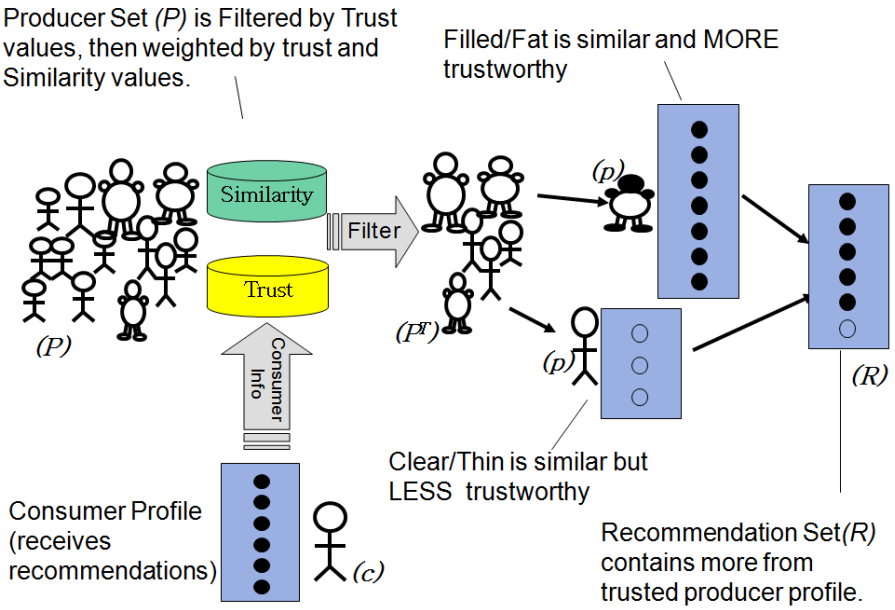

Figure 3.10: Integrating Trust Models Into Standard Collaborative Filtering.

achieved by assuming continuity and assessing the quality of each partner's contribution to recommendations in the past. Intuitively, if a profile has made lots of accurate predictions in the past they can be viewed as more trustworthy that another profile that has made many poor predictions. In this section we define two models of trust and show how they can be readily incorporated into the mechanics of a standard collaborative filtering recommender system.

3.3.1 Combining Trust in ACF

Figure 3.10 shows an overview of the trust-based recommendation process. In this figure, trust is integrated into a standard ACF algorithm by using trust information to filter recommendation partners and weight recommendations. Initially, trust and similarity are calculated for every pair of users in the database in an offline process. There are a range of ways in which trust is computed, resulting in different models. These are discussed in detail in Section 3.3.2, for now the focus is on integration of trust into ACF. During the recommendation process, peer groups are firstly constructed based on a

similarity function and then they are filtered based on trust. In Figure 3.10, C is the active user (or "consumer") who is seeking recommendations from the system. P is the set of potential recommendation partners (also known as the "neighbourhood", "peergroup" or "recommendation producers") who are most highly correlated and therefore "similar" to the active user. A personalised trust score is retrieved from a database for the consumer and each potential recommendation partner in turn. If this value is below a certain threshold, that potential producer is dropped from the candidate set. In this manner, only highly similar and highly trustworthy users from the final set of recommendation producers P_t.

Now that the set of producer profiles has been selected, the problem of combining their individual contributions must be addressed. To give a quick example, if I had two highly trusted friends who both were computer scientists, and I needed a recommendation on the quality of a new programming language, I would listen to both opinions, but most likely I would give more weight to the recommendation from person who had the better reputation in the field of programming languages. Accordingly, in Figure 3.10 the contribution of each member of the trusted producer set P_t is weighted according to the associated trust value. In this manner, more trusted users (indicated as the fatter users in Figure 3.10) have more influence over the final recommendation set R that is presented to the consumer. Section 3.3.3 provides a detailed description of the specific algorithms which have been developed to combine trust with similarity in the recommendation process.

3.3.2 Capturing Profile-Level & Item-Level Trust

Figure 3.11 shows a high-level overview of the computation described in this section. A ratings prediction for an item, i, by a producer p for a consumer c, is deemed *correct* if the predicted rating, $p(i)$, is within ϵ of c's actual rating $c(i)$; see Equation 3.2. Of course normally when a producer is involved in the recommendation process they are participating with a number of other recommendation partners and it may not be possible to judge wether the final recommendation is correct as a result of p's contribution. Accordingly, when calculating the correctness of p's recommendation we separately perform the

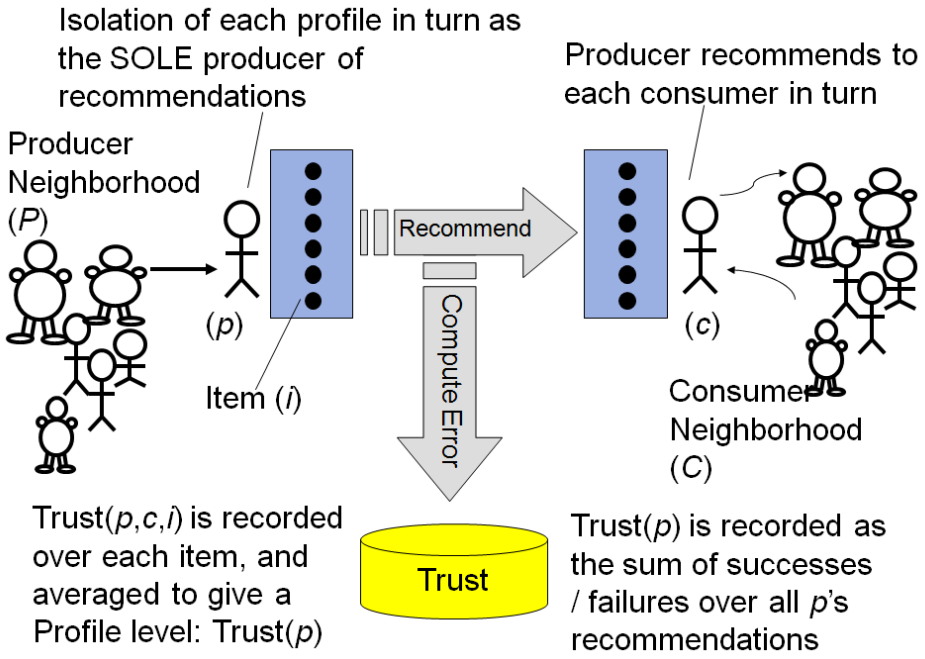

Figure 3.11: Overview of the Trust Building Process

recommendation process by using p as c's sole recommendation partner. For example, in Figure 3.12, a trust score for item *i1* is generated for producer b by using the information in profile b *only* to generate predictions for each consumer profile. Equation 3.3 shows how each box in Figure 3.12 translates to a binary success/fail score depending on whether or not the generated rating is within a distance of ϵ from the actual rating a particular consumer has for that item. In a real-time recommender system, trust values for producers could be easily created on the fly, by a comparison between our predicted rating (based only on one producer profile) and the actual rating which a user enters.

$$Correct(i, p, c) \Leftrightarrow |p(i) - c(i)| < \epsilon \tag{3.2}$$

$$T_p(i, c) = Correct(i, p, c) \tag{3.3}$$

From this we can define two basic trust metrics based on the relative number of correct recommendations that a given producer has made. The

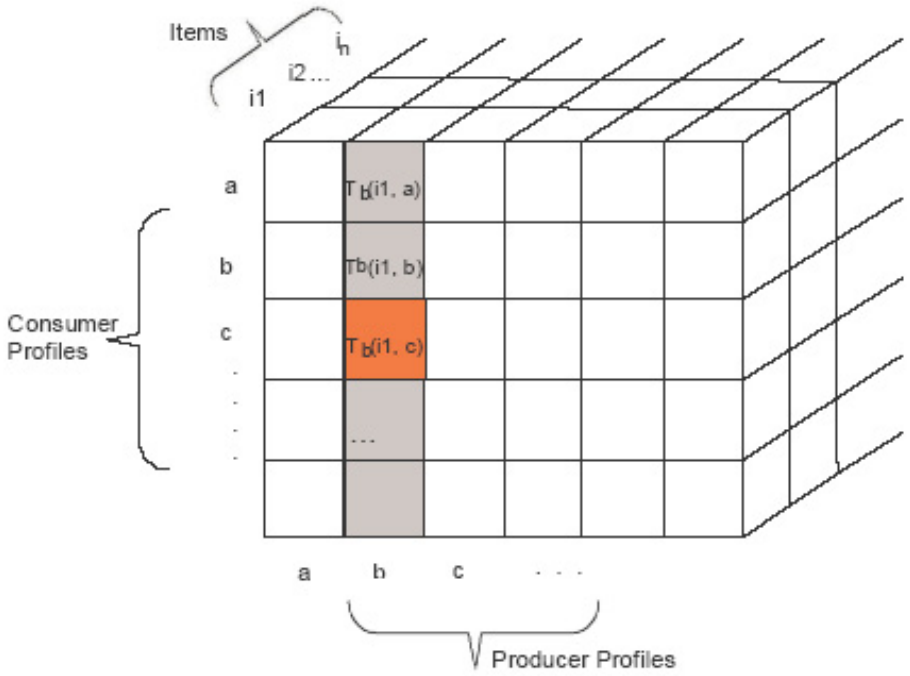

Figure 3.12: Calculation of Trust Scores from Rating Data

full set of recommendations that a given producer has been involved in, $RecSet(p)$, is given by Equation 3.4. And the subset of these that are correct, $CorrSet(p)$ is given by Equation 3.5.

$$RecSet(p) = \{(c_1, i_1), ..., (c_n, i_n)\} \tag{3.4}$$

$$CorrSet(p) = \{(c_k, i_k) \in RecSet(p) : Correct(i_k, p, c_k)\} \tag{3.5}$$

The *profile-level trust*, $Trust^P$ for a producer is the percentage of correct recommendations that this producer has contributed; see Equation 5. Obviously, profile-level trust is very coarse grained measure of trust as it applies to the profile as a whole.

$$Trust^P(p) = \frac{|CorrSet(p)|}{|RecSet(p)|} \tag{3.6}$$

Suppose a producer has been involved in 100 recommendations, that is they have served as a recommendation partner 100 times, and for 40 of these

recommendations the producer was capable of predicting a correct rating, the profile level trust score for this user is 0.4. In reality, we might expect that a given producer profile may be more trustworthy when it comes to predicting ratings for certain items than for others. Accordingly we can define a more fine-grained item-level trust metric, $Trust^I$, as shown in Equation 3.7, which measures the percentage of recommendations for an item i that were correct.

$$Trust^I(p,i) = \frac{|\{(c_k,i_k) \in CorrSet(p) : i_k = i\}|}{|\{(c_k,i_k) \in RecSet(p) : i_k = i\}|} \tag{3.7}$$

3.3.3 Trust-Based Recommendation

Section 3.3.1 presented a high level graphical overview of the process by which trust is integrated into an ACF algorithm. Now we present details of three specific techniques for harnessing trust to achieve better recommendations. We will consider 2 distinct adaptations of a standard ACF technique: *trust-based weighting* and *trust-based filtering*, both of which can be used with either profile-level or item-level trust metrics. The third technique presented is a hybrid of the first two adaptations.

Trust-Based Weighting

Perhaps the simplest way to incorporate trust in to the recommendation process is to combine trust and similarity to produce a compound weighting that can be used by Resnick's formula; see Equation 3.8.

$$c(i) = \overline{c} + \frac{\sum_{p \epsilon P(i)} (p(i) - \overline{p}) w(c,p,i)}{\sum_{p \epsilon P(i)} |w(c,p,i)|} \tag{3.8}$$

$$w(c,p,i) = \frac{2(sim(c,p))(trust^I(p,i))}{sim(c,p) + trust^I(p,i)} \tag{3.9}$$

For example, when predicting the rating for item i for consumer c we could compute the arithmetic mean of the trust value (profile-level or item-level) and the similarity value for each producer profile. A modification on this has been made by using the harmonic mean of trust and similarity; see

Equation 3.9 which combines profile similarity with item-level trust in this case. The advantage of using the harmonic mean is that it is robust to large differences between the inputs so that a high weighting will only be produced if both trust and similarity scores are high. Algorithm 1 is a pseudocode representation of the prediction process using trust-based weighting.

Input: Set of producer profiles P, trust model T, consumer profile c
Output: item recommendation for consumer profile
foreach *Producer Profile p* **do**
  foreach *CandidateItem i* **do**
    w(c,p,i) = harmonicMean(sim(c,p), trust(p,i)) ;
    sum = sum + (p(i)-p)w(c,p,i) ;
    div = div + abs(sim(c,p)) ;
  end
  RecList(i) = ConsumersAverageRating + (sum/div) ;
end
return sortByValue(RecList)

Algorithm 1: Prediction using trust-based weighting

Trust-Based Filtering

As an alternative to the trust-based weighting scheme above we can use trust as a means of filtering profiles prior to recommendation so that only the most trustworthy profiles participate in the prediction process. For example, Equation 3.10 shows a modified version of Resnick's formula which only allows producer profiles to participate in the recommendation process if their trust values exceed some predefined threshold; see Equation 3.11 which uses item-level trust ($Trust^I(p,i)$) but can be easily adapted to use profile-level trust. The standard Resnick method is thus only applied to the most trustworthy profiles. Pseudocode for the prediction process using trust-based filtering is shown in 3.10.

$$c(i) = \overline{c} + \frac{\sum_{p \epsilon P^T(i)} (p(i) - \overline{p}) sim(c,p)}{\sum_{p \epsilon P^T(i)} |sim(c,p)|} \tag{3.10}$$

$$P_i^T = \{p \epsilon P(i) : Trust^I(p,i) > T\} \tag{3.11}$$

Input: Set of producer profiles P, trust model T, consumer profile c
Output: item recommendation for consumer profile
foreach *Producer Profile p* **do**
  if *Trust(i,p,c)* $\geq$ *threshold* **then**
    foreach *CandidateItem i* **do**
      sum = sum + (p(i)-pAvg)sim(c,p) ;
      div = div + abs(sim(c,p)) ;
    end
    RecList[i] = ConsumersAverageRating + (sum/div) ;
  end
end
return sortByValue(RecList) ;

Algorithm 2: Prediction using trust-based filtering

Combining Trust-Based Weighting & Filtering

Of course, it is obviously straightforward to combine both of these schemes so that profiles are first filtered according to their trust values and the trust values of these highly trustworthy profiles are combined with profile similarity during prediction. For instance, Equation 3.12 shows both approaches used in combination using item-level trust.

$$c(i) = \overline{c} + \frac{\sum_{p \epsilon P^T(i)} (p(i) - \overline{p}) w(c,p,i)}{\sum_{p \epsilon P^T(i)} |w(c,p,i)|} \tag{3.12}$$

3.4 Evaluation

So far the argument has been that profile similarity alone may not be enough to guarantee high quality predictions and recommendations in collaborative filtering systems. Trust has been highlighted as an additional factor to consider in weighting the relative contributions of profiles during ratings pre-

diction. In the discussion section all of the important practical benefits of incorporating models of trust into the recommendation process are analysed. Specifically, a set of experiments conducted to better understand how trust might improve recommendation accuracy and prediction error relative to more traditional collaborative filtering approaches is described.

3.4.1 Setup

This experiment uses the standard MovieLens dataset [102]. This set contains 943 profiles of movie ratings. Profile sizes vary from 18 to 706 with an average size of 105. We divide these profiles into two groups: 80% are used as the producer profiles and the remaining 20% are used as the consumer (test) profiles.

Before evaluating the accuracy of the new trust-based prediction techniques, trust values must firstly be build up for the producer profiles as described in the next section. It is worth noting that ordinarily these trust values would be built on-the-fly during the normal operation of the recommender system, but for the purpose of this experiment they have been constructed separately in an off-line process, but without reference to the test profiles. Having built the trust values, the effectiveness of our new techniques are evaluated by generating rating predictions for each item in each consumer profile by using the producer profiles as recommendation partners. This is done by using the following different recommendation strategies:

1. *Std* - The standard Resnick prediction method.
2. *WProfile* - Trust-based weighting using profile-level trust.
3. *WItem* - Trust-based weighting using item-level trust.
4. *FProfile* - Trust-based filtering using profile-level trust and with the mean profile-level trust across the producers used as a threshold..
5. *FItem* - Trust-based filtering using item-level trust and with the mean item-level trust value across the profiles used as a threshold.

6. *CProfile* - Combined trust-based filtering & weighting using profile-level trust.

7. *CItem* - Combined trust-based filtering & weighting using item-level trust.

3.4.2 Building Trust

Ordinarily the proposed trust-based recommendation strategies contemplate the calculation of relevant trust values on-the-fly as part of the normal recommendation process or during the training phase for new users. However, for the purpose of this study, trust values must be calculated in advance. This is done by running a standard *leave-one-out* training session over the producer profiles. In short, each producer temporarily serves as a consumer profile and rating predictions are generated for each of its items by using Resnick's prediction formula with each remaining producer as a lone recommendation partner; that is, each producer is used in isolation to make a prediction. By comparing the predicted rating to the known actual rating we can determine whether or not a given producer has made a correct recommendation — in the sense that the predicted rating is within a set threshold of the actual rating – and so build up the profile-level and item-level trust scores across the producer profiles. This correctness was expressed in Equation 3.5.

This approach is used to build both profile-level trust values and item-level trust values. To get a sense of the type of trust values generated histograms of the profile-level and item-level values for the producer profiles in Figures 3.13 and 3.14 are presented. In each case we find that the trust values are normally distributed but they differ in the degree of variation that is evident. Not surprisingly there is greater variability in the more numerous item-level trust values, which extend from as low as 0.5 to as high as 1. This variation is lost in the averaging process that is used to build the profile-level trust values from these item-level data. Most of the profile-level trust values range from about 0.3 to about 0.8. For example, in Figure 3.14 approximately 13% of profiles have trust values less that 0.4 and 25% of profiles have trust values greater than 0.7. By comparison less than 4% of the profile-level trust values are less than 0.4 and less than 6% are greater than

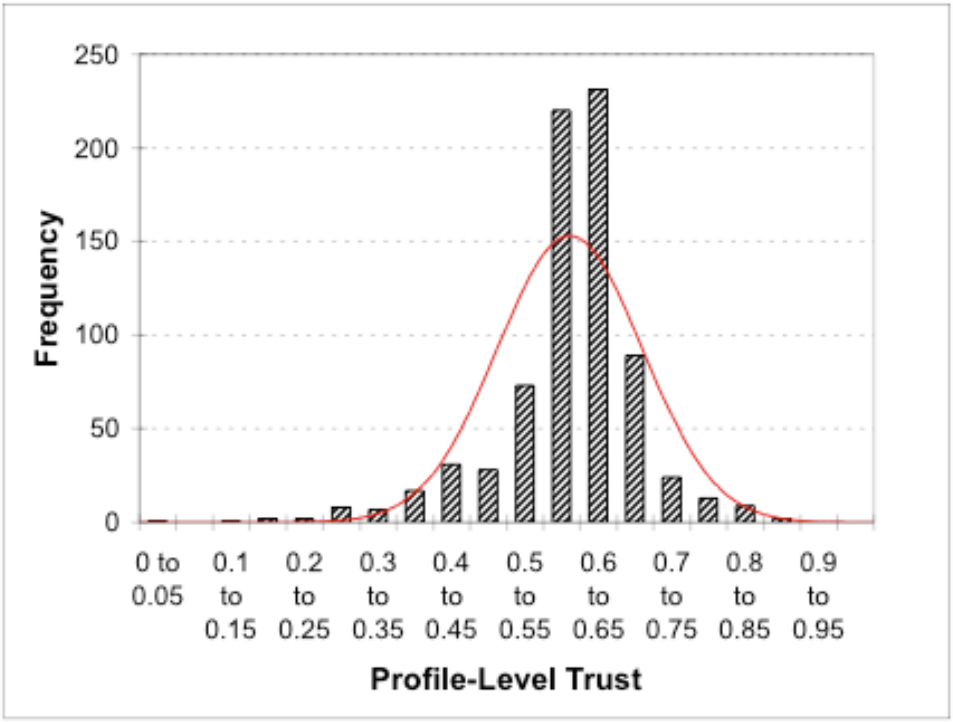

Figure 3.13: The Distribution of Profile-Level Trust Values Among the Producer Profiles.

0.7.

If there was little variation in trust then trust-based prediction strategies would not be expected to differ significantly from the standard Resnick method, but of course since there is much variation, especially in the item-level values, then significant differences between the predictions made by Resnick and the predictions made by our alternative strategies are expected. Of course whether the trust-based predictions are demonstrably better remains to be seen.

3.4.3 Recommendation Error

Ultimately we are interested in exploring how the use of trust estimates can make recommendation and ratings predictions more reliable and accurate. In this experiment we focus on the mean recommendation error generated by each of the recommendation strategies over the items contained within the

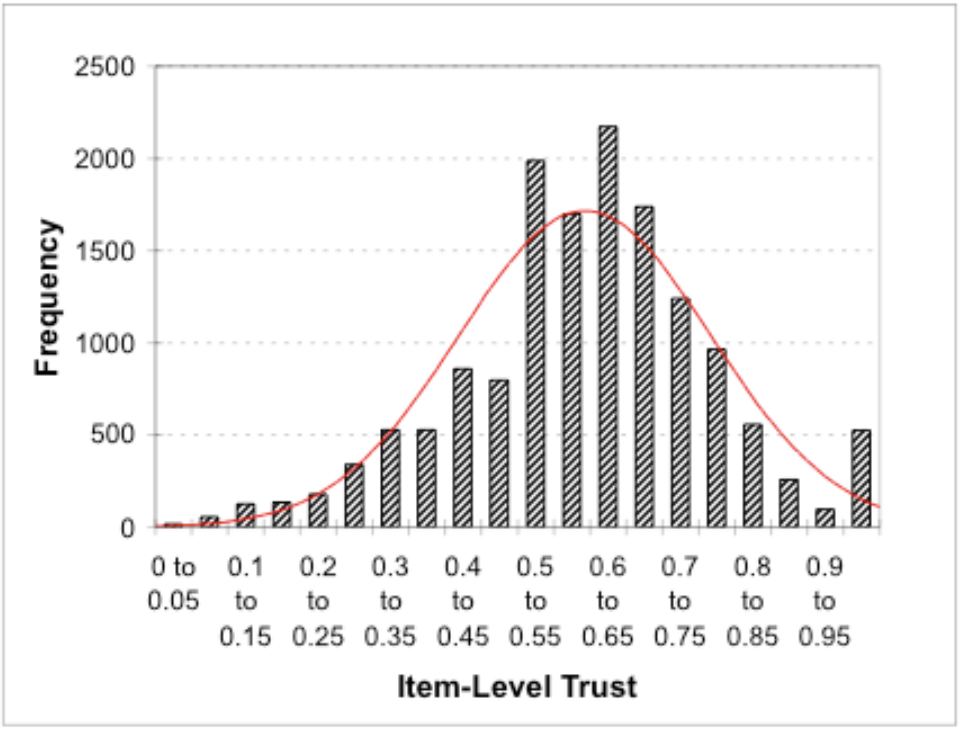

Figure 3.14: The Distribution of Item-Level Trust Values Among the Producer Profiles.

consumer profiles. That is, for each consumer profile, each of its rated items are temporarily removed and the producer profiles are used to generate a predicted rating for this target item according to one of the 7 recommendation strategies proposed above. It is worth pointing out that the rating error is calculated with reference to the item's known rating and an average error is calculated for each strategy.

The results are presented in Figure 3.15 as a bar-chart of average error values for each of the 7 strategies. In addition, the line graph represents the relative error reduction enjoyed by each strategy, compared to the Resnick benchmark. A number of patterns emerge with respect to the errors. Firstly, the trust-based methods all produce lower errors than the Resnick approach (and all of these reductions are statistically significant at the 95% confidence level) with the best performer being the combined item-level trust approach ($CItem$) with an average error of 0.68, a 22% reduction in the Resnick error.

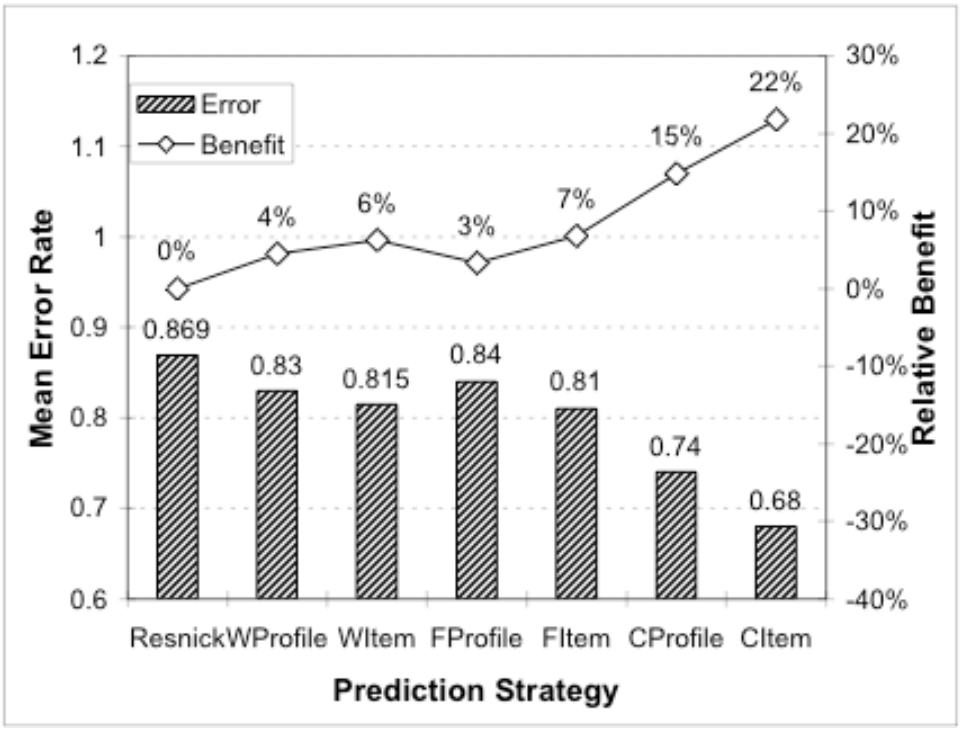

Figure 3.15: Average Prediction Error and Relative Benefit (Compared to Resnick) for each of the Trust-Based Recommendation Strategies.

It was found also that in general the item-level trust approaches perform better than the profile-level approaches. For example, $WItem$, $FItem$ and $CItem$ all out-perform their corresponding profile-level strategies ($WProfile$, $FProfile$ and $CProfile$). This is to be expected as the item-level trust values provide a far more fine-grained and accurate account of the reliability of a profile during recommendation and prediction. An individual profile may be very trustworthy when it comes to predicting the ratings of some of its items, but less so for others. This distinction is lost in the averaging process that is used to derive single profile-level trust values, which explains the difference in rating errors.

In addition, the combined strategies significantly out-perform their corresponding weighting and filtering strategies. Neither the filtering or weighting strategies on their own are sufficient to deliver the major benefits of the combination strategies. But together the combination of filtering out

untrustworthy profiles and the use of trust values during the ratings prediction results in a significant reduction in error. For example, the combined item-level strategy achieves a further 16% error reduction compared to the weighted or filter-based item-level strategies, and the combined profile-level strategy achieves a further 11% error reduction compared to the weighted or filter-based profile-level approaches.

3.4.4 Winners & Losers

So far we have demonstrated that on average, over a large number of predictions, the trust-based predictions techniques achieve a lower overall error than Resnick. It is not clear, however, whether these lower errors arise out of a general improvement by the trust-based techniques over the majority of individual predictions, when compared to Resnick, or whether they arise because of a small number of very low error predictions that serve to mask less impressive performance at the level of individual predictions. To test this, the percentage of predictions where each of the trust-based methods wins over Resnick are examined here, in the sense that they achieve lower error predictions on a prediction by prediction basis.

These results are presented in Figure 3.16 and they are revealing in a number of respects. For a start, even though the two weighting-based strategies ($WProfile$ & $WItem$) deliver an improved prediction error than Resnick, albeit a marginal improvement, they only win in 31.5% and 45.9% of the prediction trials, respectively. In other words, Resnick delivers a better prediction the majority of times. The filter-based ($FProfile$ & $FItem$) and combination strategies ($CProfile$ & $CItem$) offer much better performance. All of these strategies win on the majority of trials with $FProfile$ and $CItem$ winning in 70% and 67% of predictions, respectively.

Interestingly, the $FProfile$ strategy offers the best overall improvement in terms of its percentage wins over Resnick, even though on average it offers only a 3% mean error reduction compared to Resnick. So even though $FProfile$ delivers a lower error prediction than Resnick nearly 70% of the time, these improvements are relatively minor. In contrast, the $CItem$, which beats Resnick 67% of the time, does so on the basis of a much more impressive

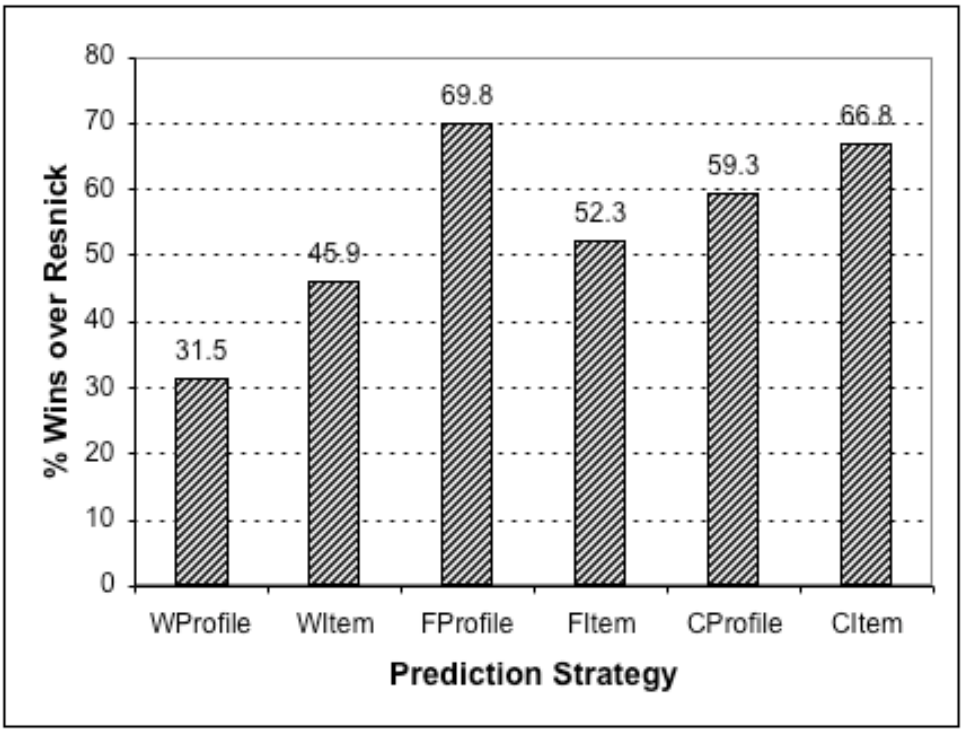

Figure 3.16: The Percentages of Predictions where each of the Trust-Based Techniques Achieves a Lower Error Prediction than the Benchmark Resnick Technique.

overall error reduction of 22%.

3.5 Summary

Traditionally collaborative filtering systems have relied heavily on similarities between the ratings profiles of users as a way to differentially rate the prediction contributions of different profiles. This chapter has presented the argument that profile similarity on its own may not be sufficient, that other factors might also have an important role to play. Specifically, the notion of trust has been applied in reference to the degree to which one might trust a specific profile when it comes to making a specific rating prediction.

Two different trust models have been developed, one that operates at the level of the profile and one at the level of the items within a profile. In both

of these models trust is estimated by monitoring the accuracy of a profile at making predictions over an extended period of time. Trust then is the percentage of correct predictions that a profile has made in general (profile-level trust) or with respect to a particular item (item-level trust). Several ways in which these different types of trust values might be incorporated into a standard collaborative filtering algorithm are presented. Each trust-based prediction technique is evaluated against a tried-and-test benchmark approach and on a standard data-set. In each case it has been found that the use of trust values has a positive impact on overall prediction error rates with the best performing strategy reducing the average prediction error by 22% compared to the benchmark Resnick CF algorithm [102].

The first two chapters of this thesis introduced research into trust in the Social Web and provided a look at current research in the area, with a focus on recommender systems and online auction technologies. This chapter has dealt with trust in recommender system applications, and with our new methods for modelling and integrating trust metrics into the mechanics of a standard ACF recommendation algorithm. Here we have discussed our contextual trust experiment, which produced empirical evidence to show that trust is very much a multivariate phenomenon, and that despite a consistent reputation, trust levels will vary based on external influences such as adversity.

The following chapter continues with an examination of trust in a collaborative filtering application, presenting a new model of trust which harnesses error-information in the rating histories of users to develop recommendation ranges for presentation to a particular consumer. Chapter 4 also introduces our first of three demonstration systems, *BoozerChooser*: a prototype online recommender system for pubs in the greater Dublin city area. This system is used to evaluate user satisfaction with the novel *range* approach to rating prediction.

Chapter 4

Computing Uncertainty for an ACF Trust Model

4.1 Introduction

This chapter continues our analysis of trust in recommender systems, specifically, in the use of *errors* in users' recommendation histories to generate trust information, in a manner that can represent the uncertainty inherent in each step of the recommendation process. We posit that useful information can be harnessed from the general trends that are present in user ratings. For example, consider a user who is a "romantic" in a movie recommender: this user may tend to rate movies based solely on their romantic content, or lack thereof. If we use this user's profile to generate predictions for another user, who happens to like comedy but not romance, chances are it is not going to serve as a good predictor. However, if the trends in recommendation error can be identified for the user producing the recommendation, then it becomes possible to adjust predictions that user generates to account for consistent errors such as a strong preference for romantic content in movies. In this chapter we define and empirically test a technique for eliciting trust values for each producer of a recommendation based specifically on the errors that this producer has made in the past.

We compute a recommendation *range* to present to a target user. This is done by leveraging under/overestimate errors in users' past contributions

in the recommendation process. Three different models to compute this range are presented and evaluated. As in the previous chapter, it is shown how this trust-based technique can be easily incorporated into a standard collaborative filtering algorithm. A fair comparison is defined in which our technique outperforms a benchmark algorithm in predictive accuracy.

The goal of this chapter is to show that presentation of a range of rating predictions is more likely to increase user trust in the recommendation system than presentation of absolute rating predictions. To evaluate the trust benefits resulting from the transparency of our recommendation range techniques, we carry out user-satisfaction trials on *BoozerChooser*, a pub recommendation system. Our user-satisfaction results show that the recommendation range techniques perform up to twice as well as the benchmark.

By implementing a system that presents range-based predictions to the end user, we also consider the effects a recommendation system interface has on the user. In [112], Sinha and Swearingen present a user-study of five music recommender systems to assess the effects that look and feel of a recommender has on user-satisfaction. They found that users are generally more confident when the system is transparent. The work of Cosley et al. in [20] is especially relevant, showing (through a technique called "re-rating") that users' opinions can actually be manipulated by a recommender system. They present evidence that users tend to rate toward the prediction that the system shows. Cosley et al. [20] also presents an experiment to examine users' reactions to different prediction displays. They find that users tend to rate fairly consistently across different scales.

4.2 An Error-Based Computational Model of Trust

This section presents technical detail of a technique for dealing with uncertainty in a trust model for an ACF recommender system. Firstly, the details of the technique are presented, followed by a worked example to highlight differences between several different approaches which were implemented. As with the trust models presented in Chapter 3, we define two separate profile

types in the recommendation process: A *consumer* profile refers to one receiving the item rating, whereas a *producer* refers to a profile that has been selected as a recommendation partner (a contributor to the recommendation) for the consumer and that is participating in the recommendation session. So, to generate a predicted rating for item i for some consumer c, we will typically draw on the services of a number of producer profiles, combining their individual recommendations according to some suitable function, such as Resnick's formula, for example. (see Equation 4.1). Once again, our benchmark algorithm uses this standard prediction formula. The nomenclature in Equation 4.1 is the same as that presented in the previous chapter.

$$c(i) = \overline{c} + \frac{\sum_{p \epsilon P(i)} (p(i) - \overline{p}) sim(c,p)}{\sum_{p \epsilon P_i} |sim(c,p)|} \tag{4.1}$$

4.2.1 Eliciting Trust From Recommendation Error

Following from models presented in the previous chapter, we again distinguish between two types of trust, *Profile-Level* and *Item-Level*. As with the previous model, these values summarise the proportion of correct recommendations that a producer has been involved in, according to a pre-defined error bound.

Here, we compute profile and item level error in a similar manner. Consider some producer profile p and some arbitrary item i which has been rated by p. We calculate the average error on p's contributions to recommendations of item i over a training set of profiles. Each ratings prediction for an item, i, by producer p for a consumer c, is within a finite error ϵ of c's actual rating $c(i)$; see Equation 4.2.

In Resnick's approach to CF, producers involved in recommendation would be operating in tandem with a number of other recommendation partners, so it may not be possible to assess whether or not producer p's contribution had a positive or negative effect on the final recommendation. To overcome this problem we isolate p's contribution to the overall recommendation by making p the sole recommendation partner for consumer c for a recommendation on item i. For example, in Figure 3.12, error scores for item *i1* are generated for producer b by using the information in profile b *only* to

generate predictions for each consumer profile.

To get a clear idea of each error computation, consider the following worked example: Table 4.2.1 shows sample item ratings for three users A, B and C. These ratings are on three items P Q and R.

	Item P	Item Q	Item R
Rater A	1	4	1
Rater B	3	3	5
Rater C	5	2	3

Table 4.1: Sample ratings for three users on a scale of 1 to 5.

Producer	Consumer	Item P	Item Q	Item R
A	A	0	0	0
A	B	-2	+1	-4
A	C	-4	+2	-2
B	A	+2	-1	+4
B	C	-2	+1	+2
C	A	+4	-2	+2
C	B	+2	-1	-2

Table 4.2: Sample prediction errors computed from data in Table 4.2.1 for different Producer-Consumer combinations.

$$T_p i, c = |p(i) - c(i)| = \epsilon \tag{4.2}$$

Equation 4.2 shows how each box in Figure 3.12 contains a separate error score based on each individual profile p being used as the sole recommendation partner for each consumer c and item i.

$$RecSet(p) = \{(c_1, i_1), ..., (c_n, i_n)\} \tag{4.3}$$

The recommendation set for a producer profile p on a specific item i is shown in the shaded area of Figure 3.12, and given by Equation 4.3. It is worth noting that in a deployed recommender system operating in real time, these values can be generated on-the-fly as new ratings are made in the system. For our experiments, we record 6 different error metrics which we show in the next section can be readily incorporated into the mechanics of a generic CF system to produce a *recommendation range* to be presented to the user. Following is an explanation of each error metric with reference back to the worked example in Table 4.2.1.

1. *PError* - The average prediction error over any time p has been involved in producing a recommendation. Using the sample data to calculate this metric for user A is as follows: $(-2+(-4))/2$ for item P, $(-1+2)/2$ for item Q and (-4+(-2))/2 for item R, giving *PError* of $(-3 + 0.5 + (-3))/3 = (-11/3)$ or -3.6

2. *IError* - The average prediction error over any time p has been involved in producing a recommendation for each item i. This is stored as a separate value for each item. Again, using our sample data, a computation of this metric for user A yields: $(-2 + (-4))/2 = -3$ for item P, $(-1 + 2)/2 = 0.5$ for item Q and (-4+(-2))/2 = -3 for item R

3. *PUnderestimate* - The average underestimate over all the times p is involved any recommendation, and there has been an underestimate in the prediction. On our sample data, this computation is: $(-2 + (-4))/2 = -3$ for item P, $-1/1$ for item Q and $(-4+(-2))/2 = -3$ for item R, which gives a *PUnderestimate* of $(-3+(-1)+(-3))/3 = -7/3$ or -2.33

4. *IUnderestimate* - The average underestimate over all the times p is involved in recommendation of each specific item, and there has been an underestimate in the prediction. For this metric a value is stored for each specific item. On the sample data we get: $(-2 + (-4))/2 = -3$ for item P, $-1/1$ for item Q and $(-4 + (-2))/2 = -3$ for item R

5. *POverestimate* - The average overestimate over all the times p is involved in recommendation of any item, and there has been an overestimate in the prediction. Again, using the sample data to compute this metric for user A: No overestimate is available on item P, (2/1) for item Q and again, no value available for item R. This gives a *POverestimate* of +2 for user A.

6. *IOverestimate* - The average overestimate over all the times p is involved in recommendation of each specific item, and there has been an overestimate in the prediction. One *IOverestimate* value is stored per item. Computing this metric on the sample data for user A: No overestimate is available on item P, $(2/1) = 2$ for item Q and again, no value available for item R.

$$IError(p,i) = \frac{\sum_{i \in R} |p(i) - c(i)|}{n} \tag{4.4}$$

$$PError(p) = \frac{\sum_{i \in P} (IError)}{n} \tag{4.5}$$

Equation 4.4 shows our calculation of the basic error for producer profile p at the item level (for item i). n represents the number of recommendations used in the calculation. $p(i)$ is the rating predicted on item i using producer p as the sole recommendation partner for consumer c. $c(i)$ is the actual rating which consumer c gave to item i. This is a fine grained approach to trust value elicitation. A more coarse metric $PError$ is given by Equation 4.5, which denotes the average $IError$ over all the items in profile p.

$$IUnderest(p,i) = \frac{\sum i \in Rp(i) - c(i)}{n} : ifp(i) - c(i) > 0 \tag{4.6}$$

$$IOverest(p,i) = \frac{\sum i \in Rp(i) - c(i)}{n} : ifp(i) - c(i) < 0 \tag{4.7}$$

The case statement in Equation 4.6 defines the computation of average underestimate at the item level. Here, n represents the number of times there was an underestimate in the predictions generated using producer p for consumer c on item i. A similar computation is denoted in Equation 4.7

to get the average overestimate in p's contributions to recommendations of individual items.

To calculate these values at the profile level, we use Equation 4.8 which is simply an average of the item level underestimates over every item in profile p.

$$PUnderest(p) = \frac{\sum i \in P(IUnderest)}{n} \tag{4.8}$$

We have shown the processes involved in building our trust models based on recommendation error histories. The next step is to demonstrate how these values are manifested in the recommendation process to arrive at a more reliable and transparent recommendation solution.

4.2.2 Using Error Ranges in Recommendation

As a result of these calculations, for every producer profile and every item in these profiles, we have the producer's normal rating for this item plus the average degree to which this user tends to underestimate ratings, the average degree to which he overestimates ratings, and the overall average error for this producer and item pair. How can we use this in recommendation? One obvious direction to take is to produce a *recommendation range* rather than a point rating.

For each time we wish to predict a rating for some new target user t and item i we get 3 ratings values: r_1, r_2, r_3 which basically gives us a rating range $[r_2, r_3]$ and an expected rating, r_1, rather than just a single rating. For example, we can now say to the target user: "My predicted rating for item i is 3.2, but no less than 2.5 and no more than 4". This is showing more information to the target user, since it reflects the historical errors in contribution to recommendation, which have been mined from the data. For example, when user a overestimates a rating, it is usually by x percent. Intuitively, this is a better recommendation strategy than simply predicting a set individual value for the user. This intuition is proven to be correct in user trials discussed later in this chapter.

We propose three such recommendation strategies:

Addition and Subtraction

This is the approach loosely described above. For each recommendation r we generate for an item i, we produce r_1, r_2, r_3 respectively in the form of Equation 4.9

$$R = \{(r - avg(IUnderest))...r...(r + avg(IOverest))\} \tag{4.9}$$

The second addition/subtraction approach is more basic and used as our benchmark in comparison with the other techniques. In this approach, instead of using the average under and overestimates, we simply produce our recommendation range by discounting the standard rating by the average error to get r_1 and increment the standard rating by the average error to arrive at our upper bound r_3. This is shown in Equation 4.10

$$R = \{(r - PError)...r...(r + PError)\} \tag{4.10}$$

In this equation, R is the recommendation range and r is the standard Resnick rating.

Modified Resnick Approach

A more interesting way to produce a recommendation range from our error values is to use them *inside* the standard Resnick prediction formula. The basic equation is given in 3.1, and our modified version in 4.11, below. Instead of discounting the underestimates after a recommendation has been produced (as with the approach above), we discount the underestimates from all of the neighbours ratings as they are calculated in the standard formula. This produces a lower-bound rating r_u. A similar approach is used to increment neighbours ratings by their pre-calculated average overestimate to produce our upper-bound value r_o.

$$r_u = \overline{c} + \frac{\sum_{p \epsilon P(i)} (p(i) - IUnderest(p,i) - \overline{p}) sim(c,p)}{\sum_{p \epsilon P(i)} |sim(c,p)|} \tag{4.11}$$

The formula for computing r_o, is that in 4.11 but with a $+IOverest$ in place of the $-IUnderest$. These results yield another recommendation range:

$$R = \{(r_u)...r...(r_o)\} \tag{4.12}$$

4.3 Evaluation

In this work we have forwarded a new technique for building a model of user trust based on their history of errors in contribution to recommendations. We have proposed several techniques for integrating these into a standard recommendation algorithm as an additional metric to be considered along with a standard correlation function. In our discussion we will consider the general benefits of producing a recommendation range using these error metrics. The following section describes a specific set of experiments designed to provide us with a better grasp on the empirical improvements our technique can make to the standard CF prediction strategy.

4.3.1 Building the Error-Based Trust Model

In the following experiment we use the standard MovieLens dataset [102] which contains 943 profiles of movie ratings. Profile sizes vary from 18 to 706 with an average size of 105. We divide these profiles into two groups: 80% (753 profiles) are used as the producer profiles and the remaining 20% (190 profiles) are used as the consumer (test) profiles.

Our implementation falls into two phases: Firstly, building the trust-error model, and then using the model in recommendation. In a deployed recommender system, error values can be computed in real time, as users enter ratings to the system. In this experiment however, we build the trust values separately. For each producer profile in our training set, we temporarily assign that profile to position of *consumer*, and using the standard Resnick prediction formula, we generate predictions for that consumer, using each of the other producer profiles in turn as the sole recommendation partner. We assess the proximity of the predicted ratings to the actual ratings by employing a standard leave-one-out test, where we hide $n\%$ of the consumers profile and try to predict the hidden ratings. We compare the predicted ratings to the known real rating and record the error. These errors fall into either underestimate or overestimate categories.

This procedure is carried out for every producer-item pair in the system, and the result is an average error for each producer-item pair over all of the times that producer was involved in the production of a recommendation

Table 4.3: Max, Min and Average Errors For Each Technique.

	Average	Maximum	Minimum
Algorithm	(0-5)	(0-5)	(0-5)
IError	0.938	3.489	0
PError	0.943	1.804	0.679
IOverest	0.865	3.833	0
IUnderest	0.745	3.489	0
POverest	0.931	2.217	0.328
PUnderest	0.868	1.865	0.162

of that item. This is our item-level model. In order to get our more coarse grained profile-level model, we average these error values again, but this time over each item in a producers profile. This provides us with an average error per producer (profile level error).

Table 4.3 gives us an overview of the type of recommendation errors we found for each of our strategies. Noting that *IError* and *PError* values are for overall average error at the item and profile levels respectively, and do not necessarily have to be the sum of the mean individual under and overestimates at that level. The largest average error is at the item level at 0.937 on the MovieLens rating scale of 1 to 5. This is as expected from our recommendation results in the next section, in which the recommendation ranges produced based on these errors outperforms the others in accuracy tests, most likely due to the broader range produced from these values. We note that the profile level errors are much less extreme, *POverest*, for example having a minimum error of 0.328 in comparison with minimum *IOverest* at 0 error. This result is exactly as expected since our profile level errors are averaged over every item in a users profile. A similar trend is the case for the other profile / item level comparisons.

4.3.2 Predictive Accuracy Experiment

This experiment examines the accuracy of our techniques by assessing the percentage of correct recommendations each technique achieves. For the 190 profiles in our test set, we generated recommendation ranges for each rated

item using the three techniques described in section 3, using both item level and profile level error values separately for our modified Resnick approach, and average error value for the addition / subtraction approach. For this set we defined accuracy as a consumers actual rating falling within the predicted recommendation range. Note that training and test data are completely independent sets. Results of this experiment are presented in Figure 4.2. It is clear that the modified Resnick approach using item level error values outperforms the other techniques, beating its closest rival $avgErr$ by 4.5%. This may be due to the greater variance in the item level errors. The worst performer in fact is the modified Resnick approach using profile level errors, at 14% worse performance than its item level counterpart. This suggests that the variance in errors does have a notable effect on predictive accuracy for collaborative filtering.

4.3.3 Comparison with a Benchmark Resnick Algorithm

Since we are predicting ranges on a recommendation scale, as opposed to scalar or "point" ratings, we must define a new method to evaluate the performance of our technique with respect to a benchmark (Resnick) algorithm. To this end, we define a scalar constant which is the average of the absolute differences between our upper bound r_o and our lower bound r_u, (which is 1.85). We propose that a fair comparison would be to assume a standard Resnick prediction to be 'correct' if it falls within half of this distance either side of the real rating. This is only a pseudo-comparison, as it is impossible to compare these techniques directly.

Figure 4.1 is a trend graph of the of the recommendation ranges for our best performer, the modified Resnick approach using item level errors. Here we can see the upper bounds r_o and lower bounds r_u of the recommendation range generated by using $IUnderest$ and $IOverest$ respectively in the standard Resnick prediction formula, see Equations 3.1 and 4.11. The real ratings provided by our test profiles tend to remain within the range, even when the standard Resnick prediction tends away from the real rating. This outcome has a positive effect not only on predictive accuracy, but also on users overall

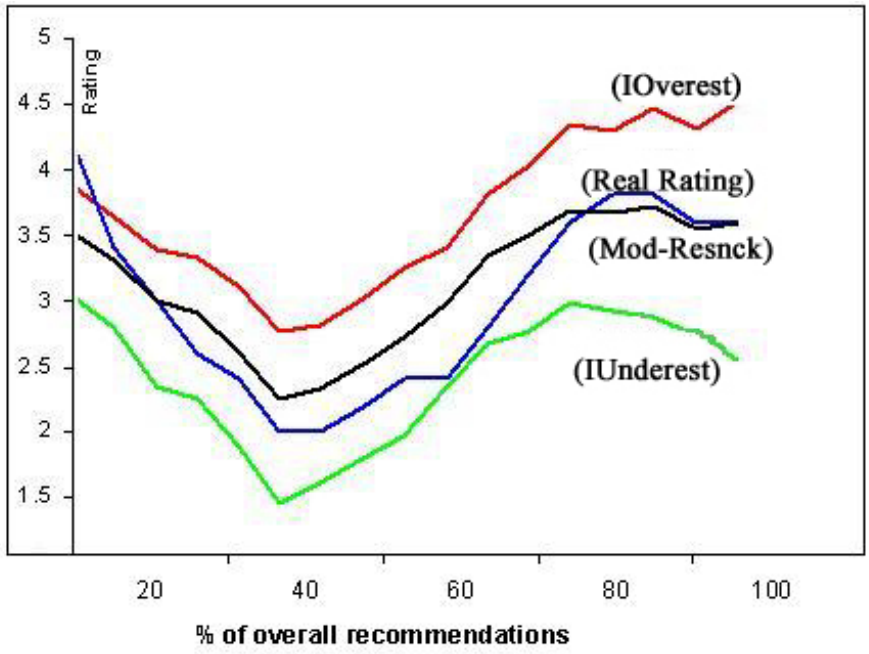

Figure 4.1: Distribution of the Recommendation Ranges and Actual Ratings for the Modified Resnick Technique.

trust in the recommender system as a whole, reducing the mal-effects of false positives in recommendations.

From Figure 4.2, it is clear that the range technique performs 15% better than the benchmark using our fair comparison. This does show that the direction in which the error models pull the r_o and r_u ratings is having a positive effect on predictive accuracy.

4.4 Live User Study

Empirical evaluations of our trust-based recommendation strategies have been presented, and it was shown that there are performance benefits on a static dataset. We further our evaluation by presenting an assessment of the performance of our algorithms in a live system. The previous section defined a "fair" comparison with a standard, single-point recommendation

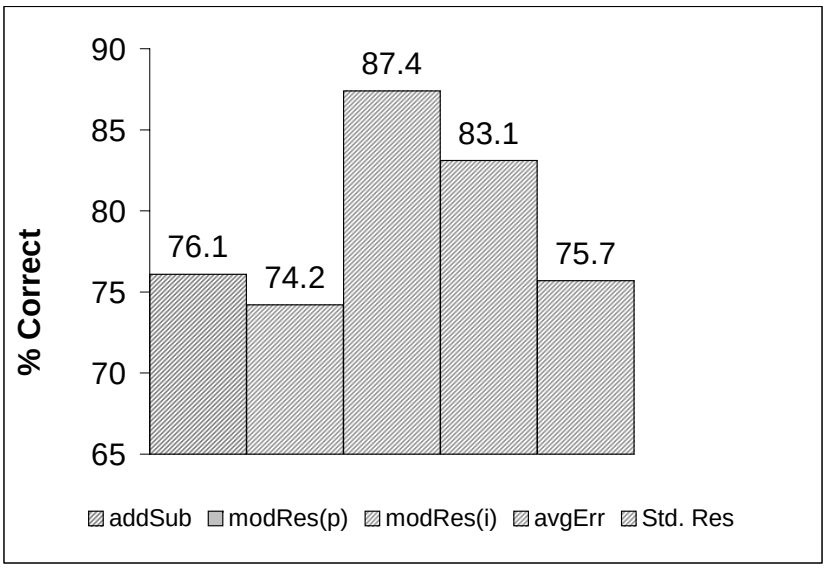

Figure 4.2: % Ratings Inside Recommendation Range.

algorithm. We note however that a defining a true fair comparison between a discrete recommendation and a continuous recommendation range is a difficult task.

In this section, we focus on assessing the impact of recommendation ranges on real world users. Intuitively, the more transparent a recommender system is, the more a user would be inclined to trust output from that system, provided of course that the system is basing its recommendations on reasonable assumptions. The effects of transparency on user satisfaction in recommendation systems has been dealt with extensively in work by O'Mahony et al [90], and Burke [16], and it has been shown by Sinha and Swearingen in [112] that "users like and feel more confident about recommendations they perceive as transparent." It is not possible to empirically evaluate the effects of providing recommendation ranges to users directly, without defining assumptions such as our comparison in the previous section. To overcome this problem we deployed each algorithm in a live recommendation system

Figure 4.3: Introductory Screenshot of the *BoozerChooser* System.

in order to directly assess the (subjective) impact that a recommendation range approach has on users, in the form of live user trials.

4.4.1 *BoozerChooser*: Prototype Recommender System

*BoozerChooser** is a live online recommender system, designed to provide personalised recommendations of Dublin city pubs to its users. Currently, the user base is small, consisting of less than 100 users from a similar demographic. To overcome issues such as cold-start and sparsity in the system, the number of items in the system has been restricted to 100, and each user has been asked to rate as many of these as possible at registration. At present, there are approximately 2000 ratings, meaning the dataset is about 80% sparse.

*www.johnodonovan.net/BoozerChoozer

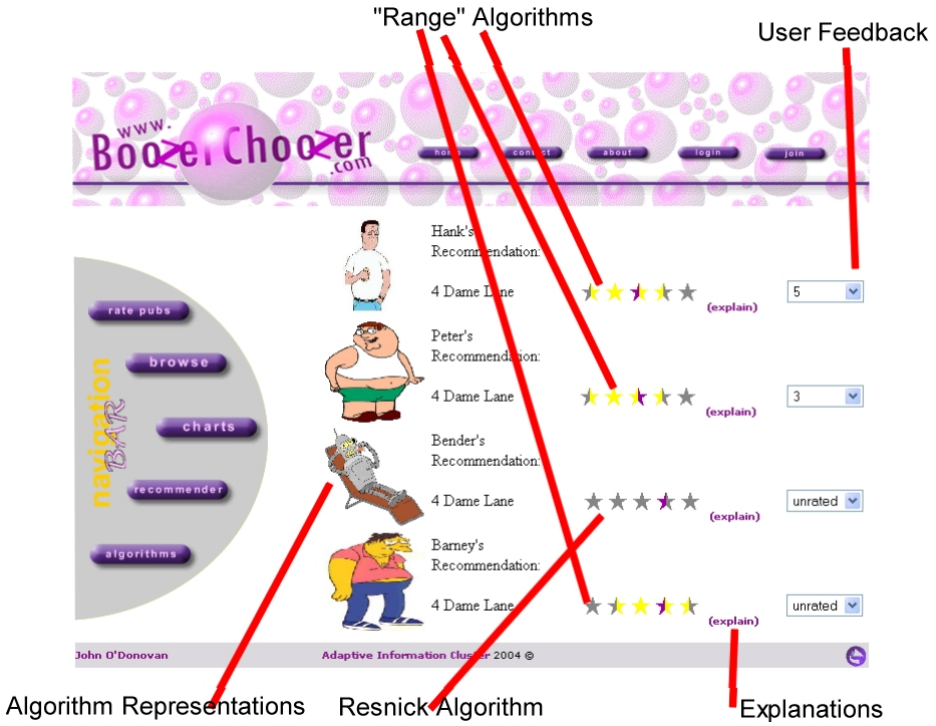

Figure 4.4: Screenshot of a Recommendation Page.

In this system, users can sign up and become members to receive personalised recommendations. Figure 4.3 shows a screenshot of the system. Apart from the recommendation pages, users can learn about the algorithms which drive the system, browse individual venue details, or view leaderboard details, a dynamically updated list of the highest and lowest rated venues (shown in Figure 4.5).

4.4.2 User Trials

User-satisfaction is evaluated by explicitly asking users to rate recommendations from each algorithm in a series of live recommendation sessions. Similar trials have been carried out to assess subjective features of recommenders in [72], where an assessment of a critiquing system is presented. In the *Boozer-Chooser* trial, a user was asked to rate more than 20 items in the system to build a profile to enable the algorithms to provide good quality recommendations. Once a reasonable profile has been built, users are asked to select

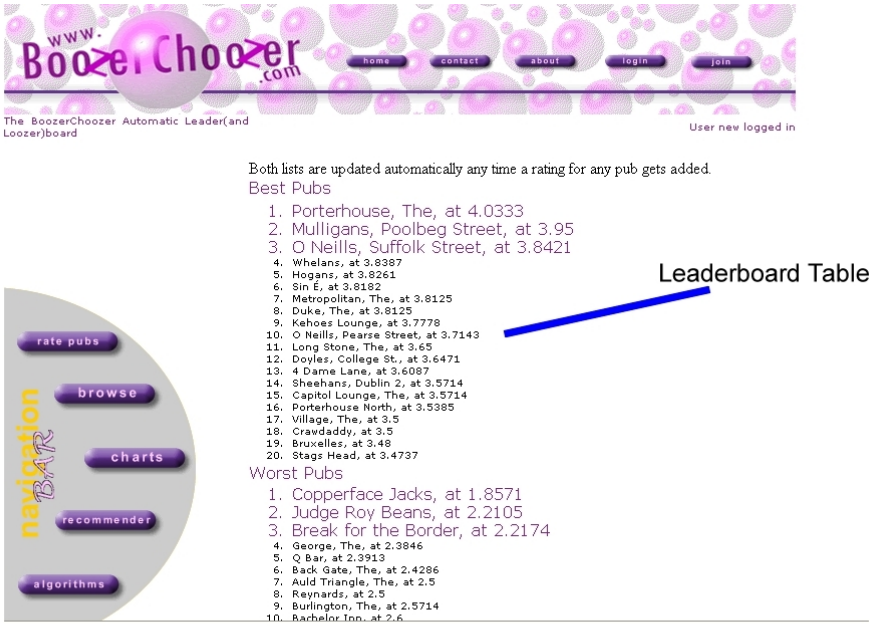

Figure 4.5: Leaderboard Charts.

as many items as possible for recommendation, provided that they already have an idea of the rating that they would give for each item. The system then provides four different recommendations for that item, one from each of our test algorithms, *addSub(), modRes(i), modRes(p), resnick()*, and participants are asked to assign a rating to each recommendation. The algorithms are disguised in the form of well known cartoon characters, so a test user has no idea which algorithm generated which recommendation. Figure 4.4 shows a screen-shot of the recommendation page for the test, displaying the four characters, the predictions from each, a link to explanations for each and a facility for the user to rate the recommendation. The benchmark Resnick algorithm is distinct in Figure 4.4, in this case the recommendation is denoted by half a gold star in the position of the Resnick prediction. For the other algorithms (highlighted in Figure 4.4, a recommendation range is represented as a series of contiguous gold stars, and the discrete prediction (r) is shown as half a red star somewhere in this series. For all of the algorithms,

the remainder of the scale outside of the recommendation range is denoted as greyed out stars. This technique was chosen as it is portable to any size recommendation scale, and users can immediately see the size of the scale, as well as the predictions.

This simple trial aims to show that the recommendation range technique is preferred by users as it displays more information and is less of a black-box than standard CF techniques. In total, 40 users participated in the trial, averaging 8 recommendations each. Figure 4.6 shows the results of the trial. It is immediately apparent that users prefer recommendation ranges over a discrete recommendation. Each of the range techniques beats the Resnick technique by a convincing margin, our best performing technique performs twice as well as the benchmark.

Obviously, a users satisfaction with presented recommendations will rely more heavily on the content of the recommendation than on the delivery mechanism. From the results presented in Figure 4.6, we can see that the benchmark algorithm performed almost as well as both the *addSub()* and the *modRes(p)* techniques. In our satisfaction trial however, *all* of the range techniques performed substantially better than the benchmark, yielding an average relative benefit of 34.8% over the standard technique. This result shows that there is a strong case for presentation of recommendation ranges over discrete recommendations.

4.5 Summary

In this chapter we have proposed two approaches to building up trust values based on error models, one that operates with respect to a specific user-item combination, e.g: "Bob's reputation for recommendation Toyota Landcruisers", and another that operates at a user level; e.g: "Bob's trustworthiness as a recommender". This chapter introduced three techniques for integrating these models into a standard CF algorithm to produce recommendation *ranges* to present to a user, and defined a fair method for comparing to the benchmark. Results from the evaluations indicate that trust-aided recommendation based on error models increases predictive accuracy over the benchmark by 15%. We have also shown that there is a clear increase in the

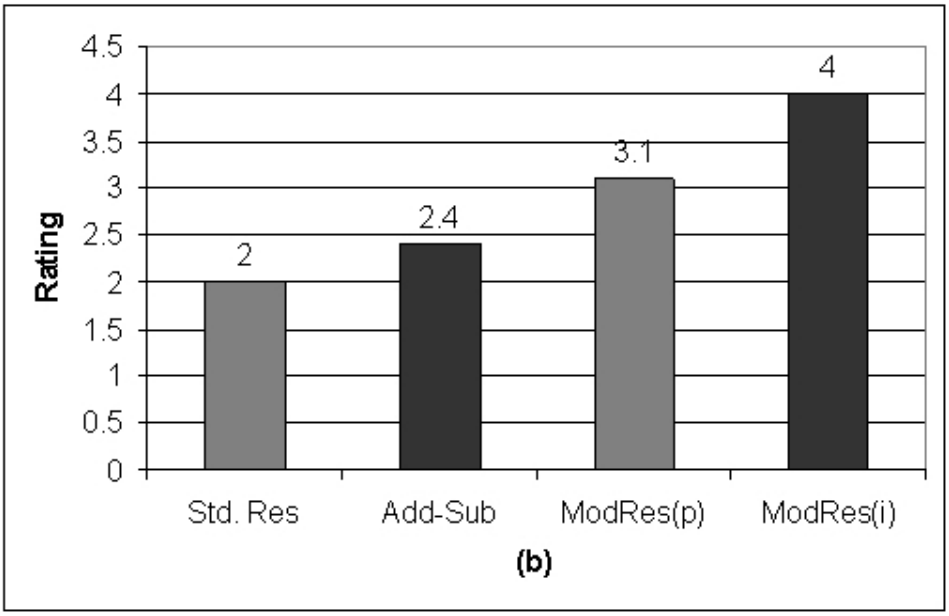

Figure 4.6: Average Rating of Recommendations From Each Algorithm in a User Survey

trust a user places in recommendations when they are presented in the form of a recommendation range derived from our error model. Looking back to the discussion on trust in earlier chapters , the recommendation range approach may provide an increase in system trust, as defined by Marsh in [69] which enhances the experience of using a recommendation system for the end user. This concludes the discussion on the methods for harnessing error-metrics to enhance recommendations.

Before moving on to the second Social Web application area, the following short chapter deals with security issues and robustness of trust based collaborative filtering. The chapter identifies some of the inherent security risks associated with trust based systems and provides some approaches to ease these risks. For each technical solution presented there is a empirical evaluation a discussion of results.

Chapter 5

Attacking Trust-Based Recommender Systems

5.1 Introduction

Systems that adapt to input from users are susceptible to attacks from those same users. Recommender systems are common targets for such attacks since there are financial, political and many other motivations for influencing the promotion or demotion of recommendable items, as outlined by O'Mahony et al. in [89] [90] and Burke et al. in [15] and [80].

Chapters 3 and 4 have shown that incorporating trust and reputation models into the recommendation process can have a positive impact on the accuracy of recommendations. In this chapter we examine the effect of using five different strategies for selecting producer profiles in a trust-based ACF algorithm on the *robustness* of the system in an attack situation. In our analysis we also consider the quality and accuracy of recommendations. Experimental results caution that including trust models in recommendation can either reduce or increase prediction shift for an attacked item depending on the model-building process used, while highlighting approaches that appear to be more robust.

This chapter deals with new ways to improve the resistance of collaborative recommenders to malicious attacks, particularly by employing trust metrics. We propose several trust-based techniques for improving the robust-

ness of collaborative recommenders in the face of malicious attack. Following on from the discussions in Chapters 3 and 4 which introduce trust models in collaborative filtering as a means to improve accuracy, we identify a security weak-spot with the trust-based approach in the face of attack. This problem, termed the *reinforcement* problem (outlined in Figure 5.1) manifests itself in terms of a greater prediction shift for an attacked item for the trust-based approach than with a benchmark collaborative filtering algorithm. The cause of this large prediction shift is that the attacking profiles tend to reinforce each other's trust values as these values are being built up during the early stages of the process, if those profiles have been present from the start. Lets say for instance that there were 10 attacking profiles in the system $p_1, ...p_{10}$. During the trust building process, the trust rating for user p_1 is calculated by allowing p_1 to generate predictions for the other users in the system. Included in this group are the remaining attacking profiles $p_2, ...p_{10}$. Assuming that the attacking profiles have similar rating trends, profiles $p_2, ...p_{10}$ will reinforce the predictions generated by p_1 yielding a higher trust value for that attacking profile. This means that in the final recommendation process, the opinions of the attacking profiles will actually carry *more* weight than the genuine profiles.

As a solution to this problem we propose to modify the manner in which we choose the profiles for which p_1 generates recommendations during trust building. By selecting the users which receive recommendations during the trust building process according to metrics such as the time a user profile has been in the system, the diversity and the reliability of the profiles for instance, we show that we can lower the prediction shift for an attacked item to less than that of the standard benchmark technique, with our best performing technique reducing the prediction shift for an attacked item by 75%.

From an end-user perspective, a change in their predicted rating due to a malicious attack does not necessarily imply that the user will be unhappy with the change. It is possible that a user may share the opinions of the attacking users. To analyse this possibility, we conduct a series of accuracy tests for the attacked item, and for the system as a whole. We find that in general, as the size of the attack approaches 15% of the profiles in the

system the prediction shift caused by the attack profiles tends to make users *less* satisfied with the predicted rating.

In this chapter it is proposed that trust models can be used to increase the robustness of collaborative filtering systems. There are two defined research areas especially relevant to this work. Firstly, research in the area of trust modelling in recommender systems, and secondly, research concerning attacks on recommender systems and methods of improving robustness of collaborative filtering algorithms in the face of such attacks. The following sections present a brief overview of the research related to attacks on collaborative recommender systems.

5.2 Background

It has been shown in a variety of research that ACF systems are vulnerable to attack by users, whether malicious or otherwise. The focus of the vulnerability is the idea that classification accuracy can be perturbed by noisy data. In his 2004 work [90], O'Mahony describes an extension of the noise-free Probability Approximately Correct (PAC) model which was originally engendered by Albert and Aha in 1991. [3]. This extension provided k-nearest neighbour classifiers with the ability to handle biased class noise. The recent focus on trust in adaptive systems has led to a number of different studies which attempt to use trust as a mechanism to patch vulnerabilities in ACF systems. One such study is that of Massa [70] in which a trust-aware ACF system is developed for the skiing domain. Within this system, the filtering process is informed by the reputation of users computed by propagating trust values. The robustness of the ACF algorithm has been shown to increase as a result of the applied trust metrics, which are explicitly provided by users. Another ACF system which uses trust to improve robustness is Golbeck's FilmTrust application, which was discussed in a case study in Chapter 2. Analysis of the trust-based ACF algorithm in FilmTrust also indicates that using trust values can have a positive impact on the overall robustness of ACF algorithms. Different ACF implementations have been shown to have varying levels of robustness, for example Lam and Riedl [63] have shown that item-based ACF algorithms are generally more robust than user-based ones

in the face of attacks.

5.3 Attack Models

Previous work on algorithm robustness has primarily focused on very basic attack models. For example, OMahony et al. [90] used an attack that draws attack profiles directly from the rating database. An attack on an ACF algorithm must consist of a set of attack profiles which contain biased rating data. Attacks will usually define a *target item* which will be contained within all of the attacking profiles. An attack can attempt to either promote (push) or demote (nuke) [89] the target item. Mobasher et al. draw two distinctions between attack types in their 2007 work, they define a *Low-Knowledge-Attack* as one that requires system-independent knowledge, for example, knowledge that is obtainable from public information sources, and a *High-Knowledge-Attack*, which requires highly detailed knowledge of the ratings distribution within the ACF system's database. Burke also defines a range of attack models. The following list briefly describes the various attack models which were defined by Burke in [80]:

1. Bandwagon Attack: This is a form of push attack (i.e.: it attempts to promote the target item). Highly popular items are assigned high ratings so that the attacking profiles will be "similar" to a large number of users. Then the target item is given a max rating.

2. Segment Attack: This is a reduced-knowledge push attack specifically designed for item-based ACF algorithms. The attack attempts to target (segment) users in the system who will be most susceptible to the attack. For example, targeting users who like comedy in the promotion of a comedy book.

3. Love-Hate Attack: In this simple attack an item (targeted for demotion) is given the minimum rating in the attacking profiles. Other items are given the maximum rating.

4. Random Attack: This can be a push or nuke attack in which random items are given random ratings and the target a max or min rating.

5. Reverse Bandwagon Attack: This is a nuke attack where selected items are those that tend to be rated poorly by many users. These items are assigned low ratings together with the target item. Thus the target item is associated with widely disliked items, increasing the probability that the system will generate low predicted ratings for that item.

6. Average Attack: This is a high knowledge attack in which the average rating is given to the items in the attack profiles in an attempt to maximise the degree of overlap with users in the ACF database. It can be implemented as either a push or a nuke attack.

Now that the motivation for attacking ACF systems and the approaches to attack have been discussed, as well as some existing solutions, we move on to discuss how the trust-based ACF algorithms from Chapter 3 can be applied to increase robustness in the recommendation process.

5.4 The *CITEM* Algorithm

We have shown in Chapters 3 and 4 how trust scores are calculated for individual producers of recommendations, and how these values can be put to work in the recommendation process. It has been shown in Chapters 3 and 4, and by Massa in [71] and Golbeck in [37] that these trust values can be utilised in a variety of ways during recommendation. For our evaluations of robustness in this Chapter, we will use our best-performing algorithm, *CITEM*, introduced in Chapter 4. This approach is a combination of *trust-based weighting* of producer profile ratings and *trust-based filtering* of producer profile ratings using trust values at the item level. The weighting approach simply adds the extra metric of trust to the standard similarity weighting in Resnick's prediction formula. This modified version of Resnick's formula is shown in Equation 5.1 to include the trust weighting. Here, $w(c, p, i)$ is the simple harmonic mean between the item level trust for producer p and item i, and the similarity between producer p and the profile receiving the recommendation, i.e: the consumer profile c. The formula for calculating the simple harmonic mean is shown in Equation 5.2. Figure 3.10 from Chapter 3 illustrates the weighted producer profiles as larger members

of the producer neighbourhood. In this figure, there are more shaded items in the final recommendation set since these have come from producer profiles with high trust values.

$$c(i) = \overline{c} + \frac{\sum_{p \epsilon P(i)} (p(i) - \overline{p}) w(c,p,i)}{\sum_{p \epsilon P(i)} |w(c,p,i)|} \tag{5.1}$$

$$w(c,p,i) = \frac{2(sim(c,p))(trust^I(p,i))}{sim(c,p) + trust^I(p,i)} \tag{5.2}$$

Trust-based filtering simply filters out the producer profiles that form a consumer's neighbourhood according to the trust scores of the producer profiles. If a producer profile has a trust value which is less than the threshold T, then it's inputs are not considered in the recommendation process for a particular item. The filtering process is illustrated in Figure 3.10, where the original producer neighbourhood P gets reduced to neighbourhood P^T containing only trusted profiles.

Combining these two approaches of filtering and weighting gives us the *CItem* algorithm, which is given in Equation 5.3. In this formula, P^T is the set of trusted producer profiles.

$$c(i) = \overline{c} + \frac{\sum_{p \epsilon P^T(i)} (p(i) - \overline{p}) w(c,p,i)}{\sum_{p \epsilon P^T(i)} |w(c,p,i)|} \tag{5.3}$$

5.5 The Reinforcement Problem

Including trust metrics in the recommendation process can have a positive effect on predictive accuracy in collaborative filtering [71] [37] . The issue of security against attack in recommenders is becoming increasingly relevant, giving cause for an assessment of the performance of our trust-based recommendation techniques in the face of malicious attack. An obvious flaw was noted in the basic technique, which is illustrated in Figure 5.1. This is a graph of the prediction shift for a "pushed" item with varying attack sizes.

It is clear that the *CItem* approach has been skewed much more than the benchmark Resnick algorithm by the false attack profiles, all of which give approximately average ratings for random items, and then a maximum rating for the pushed item "Toy Story".

The reason for this large prediction shift is intuitive. Using the trust approach we have described, say for example that 10 malicious profiles $(p_{a1}, ..., p_{a10})$ find their way into the system, all with a maximum rating for the pushed item and possibly with other similar ratings too. In this situation, when we are building our trust scores, each attack profile will reinforce the ratings of each other profile. For example, p_{a1} will generate recommendations (as the sole producer profile) for his nine counterpart attack profiles $p_{a2}, ..., p_{a10}$, and these recommendations will have zero error since all the attack profiles have given the same rating for the pushed item. This means that the attacking profiles actually get *higher* trust scores than regular profiles.

Without an explicit knowledge of which profiles are malicious, it is impossible to completely block the effects of attackers in collaborative filtering [89]. We can however increase the system's resistance to attack. In order to achieve this, and to tackle the reinforcement problem in our *CItem* algorithm, we propose that modifying the selection of the set of consumer profiles C during the trust-building process can reduce the effect of push attacks in the recommender system. This is achieved by minimising the chances of an attack profile serving as a consumer profile.

5.6 Consumer Selection Strategies

To overcome the problem of reinforcement of attack profiles in the trust modelling process, we filter the profiles to be included in the trust building process. Intuitively if we do not allow the attack profiles to serve as consumers in the trust building process depicted in Figure 3.12, then they cannot reinforce the ratings of the remaining attacking profiles. In a real world system however, this is not a trivial task. The performance of any recommendation algorithm will be effected by the nature of the attacking profiles. For example, a push attack where the attacking profiles are all recently entered in the system could be completely stopped by simply selecting older profiles

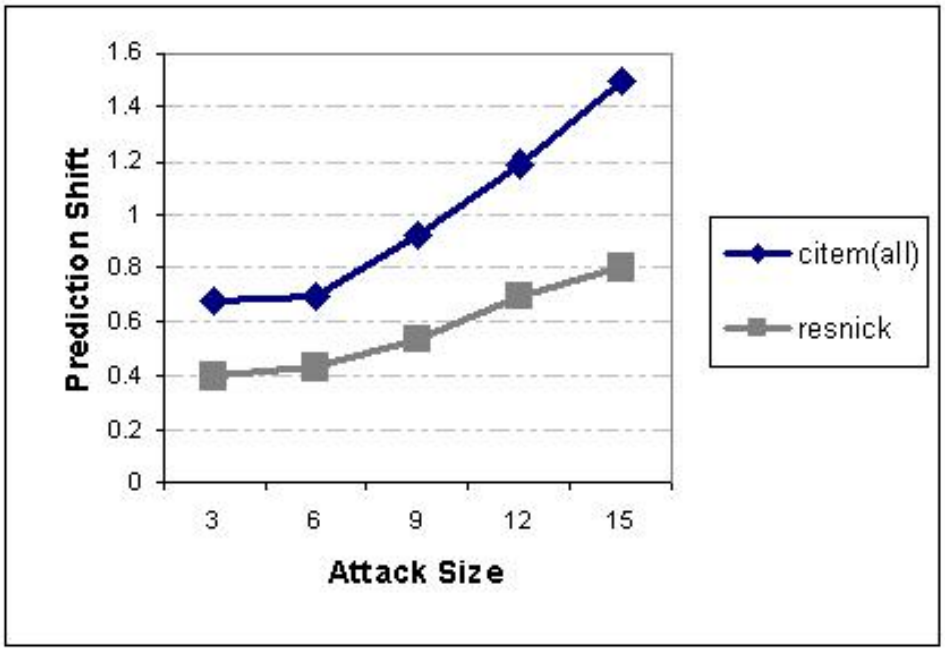

Figure 5.1: The Reinforcement Problem With Trust-Based Recommendation.

as consumers in the trust-building process. Unfortunately we do not always have the luxury of this information however, so here we explain five different consumer selection strategies and analyse in our evaluation the effects that attacks of varying size have on the predicted rating for a pushed item, and on the predictive accuracy of the recommender using each of the selection strategies. In the list below each strategy consists of a set of 100 consumer profiles, apart from *CItem(all)* which contains all 754 profiles involved in the trust building process.

1. *CItem(all)* - This is the basic *CItem* approach described in [87] which is susceptible to attack in terms of a prediction shift for a pushed item. In this selection strategy, all of the profiles in the training data are allowed to temporarily serve as consumer profiles.

2. *CItem(diverse)* - This approach selects a diverse set of consumer profiles from the training data during the model building stage. In this approach we follow the diversity formula for selection of profiles based on work in [115]. This selection strategy ensures that there are a broad

range of items rated differently in the profiles used to model trust.

3. *CItem(time)* - In this strategy, we simply select older profiles to form the consumer set C

4. *CItem(random)* - Selection of the consumer set C randomly across all of the training profiles.

5. *CItem(genuine)* - This is the "ideal" protection against malicious profiles. In an ideal situation we could simply remove the malicious profiles from the data altogether. In our experiments however, we are interested in examining the robustness of the trust based approach to recommendation, so we leave the malicious profiles in the data but do not allow *any* of them serve as consumer profiles during the trust modelling stage.

5.7 Evaluation

The argument presented in Chapters 3 and 4 was that trust models can be mined from ratings data in collaborative filtering systems and that these models can be used to improve the predictive accuracy of collaborative filtering. In this chapter we have highlighted an important robustness issue with the trust based models presented in the previous chapters. This problem can render the system susceptible to malicious attack, and is a security weak spot. We have proposed a solution to this problem by modifying the way we select our consumer profiles during the trust building process. In our evaluation we use the same dataset and experimental setup as in Chapter 4. We describe three experiments to assess the prediction shift for an attacked item, the change in recommendation accuracy for an attacked item and to evaluate the mean predictive accuracy of each of our consumer selection strategies under normal (non-attack) operation of the *CItem* algorithm. In each of our experiments, we also make comparisons to a standard Resnick collaborative filtering algorithm.

5.7.1 Experimental Setup

For this evaluation we again use the MovieLens dataset [102] which contains 943 profiles of movie ratings. The rating scale for this dataset ranges from 1 to 5, with a rating of 5 indicating a "liked" item. Profile sizes vary from 18 to 706 with an average size of 105. We divide these profiles into two groups: 80% are used as the producer profiles for trust building and the remaining 20% are used as the consumer (test) profiles.

5.7.2 Building the Trust Models

In a deployed system, the trust models which we have described can be built on the fly. Whenever a user rates an item, the system can allow producer profiles to generate predictions for that real rating, and build trust scores based on the accuracy of those predictions. For our experiments however we implement a "snapshot" of such a system in which the trust models are computed offline.

We build five different models of trust, each varying based on the consumer selection strategies described in the previous section. For each of the selection strategies we run a standard *leave-one-out* training session over all of the producer profiles. Accordingly, each profile temporarily becomes the consumer, and the remaining producer profiles generate predictions for that consumer. These predictions are generated by using Resnick's prediction formula, but with each producer profile serving as the sole recommendation partner in the prediction process. This recommendation partner is shown as p in Figure 3.12. Once a prediction is made by a producer profile, we then analyse the distance of the predicted rating from the known rating of the consumer. If this distance is within a threshold of the actual rating, we increment the trust score for that producer profile on the recommended item. In this manner we model trust for each producer of recommendations on an *item-level*. To find the *profile-level* trust for a producer we would simply average the item level trust across all of the items in the profile. For these experiments however, we are only interested the *CItem* algorithm which uses trust scores at the item level.

In the *CItem(all)* model, we allow every profile to serve as a consumer

during the training session. For each of the other strategies, *CItem(time)*, *CItem(diverse)*, *CItem(genuine)* and *CItem(random)* we select a subset of 100 profiles to serve as consumers, which is just over 13% of the training data.

5.7.3 Generating Attack Profiles

In the following two experiments, we attack the prediction algorithms by inserting false profiles to attempt to shill the predictions in a specific direction. The attack strategy we implement is a *push attack* as described in [89]. The goal of the attacking profiles is to create a favourable prediction shift for the target item "Toy Story", which has a mean rating of 3.87 and a set of 452 raters. We distributed ratings in the attack profiles according to the *average attack* strategy outlined by Burke in [15]. Attack profiles are assigned average ratings for a random set of items and then a maximum rating for the "pushed" item.

Attack size has important implications for e-commerce applications. When it comes to understanding the cost of an attack, for example, if an author must purchase many copies of his book, and possibly some other books from Amazon.com in order to build a valid attack profile, it will be very expensive to have any great effect on the products which get recommended. If the *cost of attack* is greater than the *rewards*, then there will be little point in launching an attack in the first place. Bearing this in mind, we analyse prediction shifts for each of the models with a varying attack size.

5.7.4 Prediction Shift

In this experiment we analyse prediction shift for the *CItem* algorithm while varying the size of the attacks from 0 to 15% of the data in intervals of 3%. The results of this experiment are presented in Figure 5.2. In this graph, the prediction shift (ΔR_i) is measured from zero. For instance, if an item rating was 2.5 before the attack, and 3 afterwards, then the shift would be 0.5. We can clearly see the susceptibility of the *CItem(all)* approach to the reinforcement problem described earlier. This manifests itself in terms of a prediction shift that is actually twice that of the Resnick algorithm for

an attack size of 15%. It may seem counter-intuitive that there can be a prediction shift greater than 1.23 (5 - 3.87). This occurs because the graph indicates an average shift.

However, Figure 5.2 clearly shows that the other consumer selection strategies for the *CItem* algorithm dramatically lower the prediction shift for the attacked item, in most cases to a lower amount than for the Resnick algorithm. Our *CItem(genuine)* model gives us the lowest prediction shift. For an attack size of 15%, this technique boasts a prediction shift of 0.2 in comparison with a 0.8 shift for the Resnick algorithm, giving a 75% improvement. We define this algorithm to be our ideal since in a large scale deployed system, identifying a suitable set of truly genuine profiles is not a trivial task. The results for the random, diverse and time models are within a distance of 0.2 from each other and the Resnick algorithm across all of the attack sizes. However, as the attack size tends towards 15%, the time, diverse and random models tend to shift much less than the Resnick technique, both *CItem(random)* and *CItem(diverse)* have a prediction shift of 0.2 lower than the Resnick technique for an attack size of 15%. The random technique actually does well in this experiment, considering the benefit of ease of implementation.

Although there is a notable reduction in the prediction shifts for the attacked item using different consumer selection strategies, we must consider what the implications of these techniques are for the user of the recommender system. To assess this we perform two experiments to evaluate the predictive accuracy of the *CItem* algorithm using each consumer selection strategy. Firstly, we examine the predictive accuracy on a single attacked item, and then we measure the mean predictive accuracy across all of the items in the test set without the attacking profiles. The latter experiment is similar to the accuracy test in Chapter 4, but using the new selection strategies.

5.7.5 Recommendation Accuracy

To analyse the predictive accuracy of our recommendation techniques for the attacked item "Toy Story", we evaluate the mean recommendation error generated by each technique for the items in our consumer profiles. For

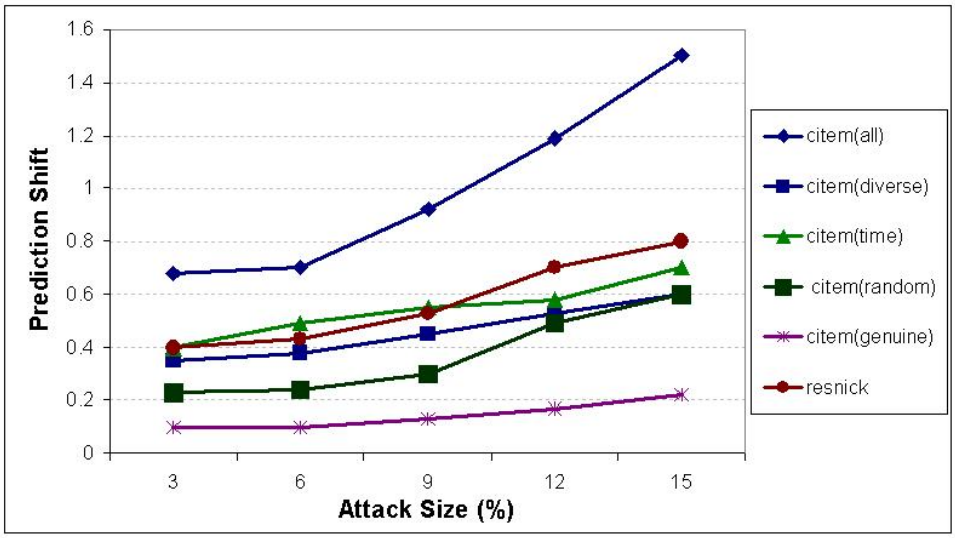

Figure 5.2: The Average Prediction Shift for Pushed Item "Toy Story" (Compared to Resnick) for each of the Trust-Based Recommendation Strategies, with Varying Attack Sizes.

each consumer profile we temporarily remove each rated item and allow the producer profiles to generate recommendations for the removed item. We record the error as the absolute distance between the predicted rating and the consumer's actual rating for that item. We record this information for each of the *CItem* models, using three attack sizes: 3, 9 and 15%. The results of this accuracy experiment are presented in Figure 5.3. It is immediately clear from these results that all of the recommendation techniques take a big predictive accuracy hit as the attack size increases, The most effected being the *CItem(all)* technique. This result is to be expected since if we look back to the prediction shift results we can see that the *CItem(all)* technique has a shift of around 1.5 for an attack size of 15%. If we consider that the mean rating for the attacked item is 3.87, and 3.52 for the whole dataset then for many profiles we can speculate that while the size of the push attack approaches 0.35 in the prediction shift graph we should be seeing a reduction in mean error compared with the Resnick algorithm. The accuracy should then drop as the predicted ratings are skewed towards the max rating of 5. By this intuition we should be seeing a greater drop in the prediction

errors for the *CItem(genuine)* and *CItem(diverse)* with attack sizes of 3 and 6, since the average prediction shift for these in Figure 5.2 is less than 0.35. We do not see this clear drop in absolute error from the graph in Figure 5.3 which leads us to consider one possibility that the attack for this experiment may have caused a larger prediction shift than that in Figure 5.2, possibly due to the fact that the distribution of ratings for "Toy Story" has large deviations lower than the mean rating, and that these profiles were used in the evaluation. These accuracy results do tell us that when attack profiles get into the recommendation process, they cause serious problems. We need to develop more intelligent ways for the system to select profiles with which to build trust models. The discussion at the end of this chapter proposes some avenues of research to assess this problem further.

However, the most accurate algorithm, although marginally, is still the *CItem(genuine)* which beats all of the others including the Resnick algorithm across all of the attack sizes. For an attack size of 6, our consumer selection techniques seem to prevent an increase in error when compared to the benchmark Resnick technique. At this attack level, our *CItem(genuine)* algorithm has a mean absolute error of 0.7, which is a 33% improvement on the benchmark. This result shows us that trust models can in fact preserve recommendation accuracy. It is clear that more research is needed to discover how to build these models for optimal performance.

5.7.6 Predictive Accuracy for a Non-Attack Situation

To make a comparison between the recommendation strategies we have described in Chapter 4, and the consumer selection strategies in this chapter, we perform a mean absolute error analysis of the predictive accuracy of each *CItem* technique in a non-attack situation. In this experiment accuracy is recorded in the same manner as for the previous experiment, but in this case we record the difference between predicted ratings and actual ratings across all of the items in the test data. Once again, for each of the *CItem* techniques we choose a set of 100 consumer profiles for the trust building session. The error metric in Equation 3.2 is set at 1.8 to give a good distribution of trust values. The results are presented in Figure 5.4. The black line indicates the

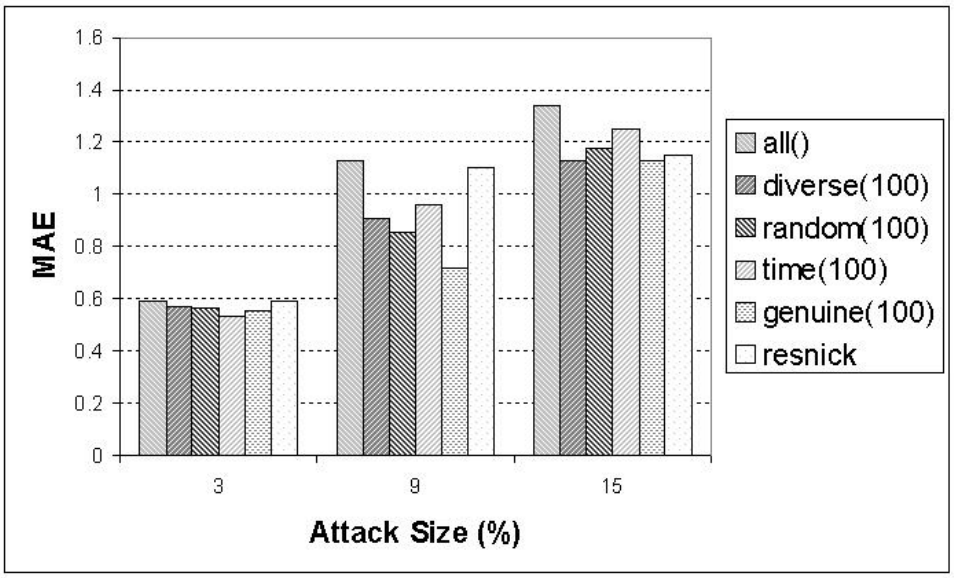

Figure 5.3: The Average Prediction Error (Compared to Resnick) for Attacked Item "Toy Story" *CItem* Algorithm Using Different Consumer Selection Strategies and Varying Attack Sizes.

relative benefit each prediction strategy shows over the benchmark Resnick algorithm. We can clearly see that every flavour of the *CItem* algorithm has a lower mean absolute error than the benchmark. As with the findings in Chapter 4, the *CItem(all)* technique improves on the benchmark by 22%. Interestingly, the benefit in accuracy is reduced across each of the new techniques by around 0.1, compared with a 7-fold reduction in the amount of consumer profiles used in the trust building session. The trust building technique has a computational complexity of $O(n^2)$ on the number of users in the system, so this reduction means that the model building process is 7 times faster for a reduction of 0.1 in accuracy.

5.8 Discussion

In this chapter we examined the robustness of trust based recommendation using different trust models, and an average attack strategy. It would be interesting to carry out a further examination of our trust based algorithms

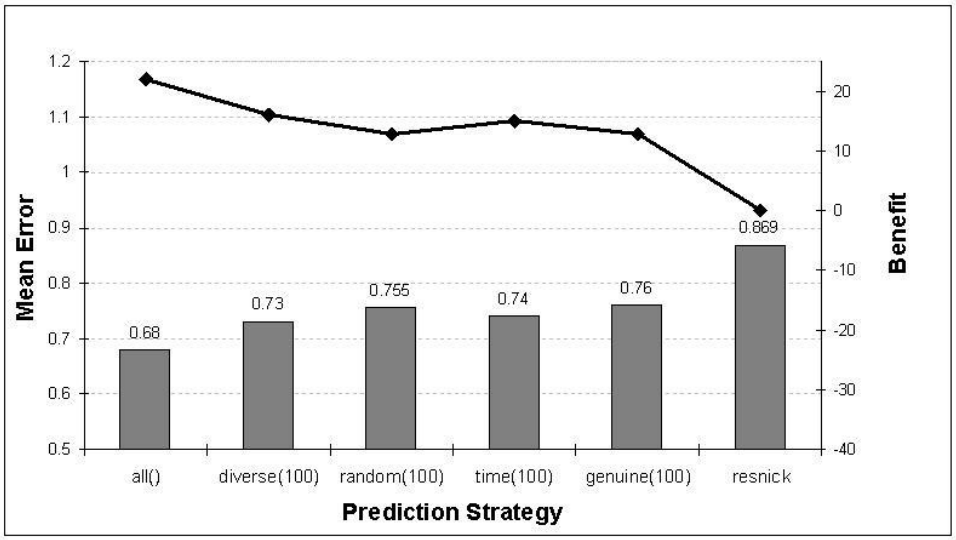

Figure 5.4: The Average Prediction Error and Relative Benefit (Compared to Resnick) for each of the *CItem* Algorithms Using Different Consumer Selection Strategies.

under different attack scenarios. For instance in the simple case of a sudden attack on a collaborative recommender in which all of the attacking profiles are recent, our *CItem(time)* strategy should perform as well as the *CItem(genuine)* technique since none of the attack profiles would be selected as consumers in the model building process. Another interesting question arises from our analysis of prediction shift: would the prediction shifts be similar for a *nuke* attack in which an item is targeted for *demotion* by the attacking profiles?

In many cases there are similarities and patterns in attacking profiles, for example in the push attack described in this document each attacking profile contained a rating of 5 for the pushed item "Toy Story". With the possible exception of the *diverse* and the *time* approaches to consumer selection which we have proposed, there is not much "intelligence" involved in the selection process as it stands. In all of the attacks described by O'Mahony in [89] and Burke in [15, 15] there are clear patterns in the attacking profiles, such as consistent average ratings in the case of the *average attack* and similarity to one profile in the case of a *favourite item* attack. An interesting avenue of fu-

ture research might be to examine each common attack strategy to highlight their salient features so we can minimise the chances that an attack profile will be used in the consumer selection process. For this examination we will use pattern matching algorithms (possibly the *apriori* rule generating algorithm described in [92]) and other machine learning techniques to attempt to identify the attacking profiles.

Research into trust modelling in collaborative filtering is no doubt still in its infancy, we believe however that as recommender systems proliferate on the web, the issue of who to trust and who to avoid will become vitally important to receiving good quality, reliable recommendations. In this chapter we have argued that conventional collaborative filtering techniques are susceptible to attack from malicious users. We have shown also that the trust based collaborative filtering algorithms presented in earlier chapters can be even more susceptible since attacking profiles can actually reinforce each other's opinions during the trust building process. We have proposed a solution to this problem by modifying the manner in which we choose consumer profiles during the trust building process. We have shown four new approaches to the trust building process and shown in our evaluation that the prediction shift for an attacked item can be reduced by modifying the selection of consumer profiles for trust building. Our best technique provides a 75% reduction in prediction shift when compared with the benchmark Resnick algorithm and 87.5% when compared with the *CItem(all)* algorithm presented in Chapter 4. We have also presented results of two experiments to evaluate the predictive accuracy of the new techniques in both attack and non-attack situations and shown that our best strategy consistently beats the benchmark Resnick algorithm in MAE tests.

5.9 Summary

To recap on the progress of this thesis up to this point: The overarching goal is an analysis of the issue of trust on the Social Web. Previous chapters have discussed the evolution of the Social Web as a participation platform, or a framework for user-authored content. A detailed discussion of the various meanings of trust was presented, using input from a range of disciplines.

Following this, a detailed account of recommendation systems was presented, with appropriate case studies to illustrate concepts. The previous two chapters analysed trust models for collaborative filtering based on historical data (or reputation) and on error tendencies. This chapter dealt with the robustness of such trust models in the face of malicious attack. The following chapter details a technique for visualising the trust relations in multidimensional space. The system presented gives a user an overview of the trust-space and an opportunity to interact with and manipulate the visualisation to inform the system of new trust information. The system presented is a tool for trust *management*.

Chapter 6

Interactive Visualisation of Trust in Recommender Systems

6.1 Introduction

The core focus of this thesis is an analysis of the role of trust in the context of Social Web applications such as recommender systems. So far, a range of algorithms for harnessing trust within ACF systems have been presented and evaluated. This chapter also deals with a ACF systems, but the focus is on the user interface as opposed to the recommendation algorithms alone. Specifically, this chapter documents the design, implementation and testing of an interactive interface for ACF in which peer *similarity* and peer *trust* can be visualised and manipulated by the user during the recommendation process to reflect their current recommendation requirements.

The proliferation of Social Web applications such as Facebook [64] and eBay is bringing about a need for new ways to present social information to the end user. These applications are constantly evolving and adding new functionality to keep users interested. For example, the social networking site Facebook recently added the Touchgraph [117] visualisation tool which expresses trust as a function of node-size. (See Figure 6.2).

This chapter introduces *PeerChooser*, an extension for Social Web applications which controls and manages trust in a user friendly manner. The system uses a novel technique for the visualisation of a neighbourhood space

for a social networking system, catering not only for a *similarity* metric, but also for *trust*. This chapter details the manner in which we present the end-user with an overview of their personalised *trust-space*, which can be tweaked to reflect current mood and requirements. The application presented can be viewed as a *glass-box* [46] approach to recommendation, in contrast to the black-box nature of most recommender systems which leave the user wondering how the system arrived at its recommendation.

PeerChooser, is a collaborative recommender system with an interactive interface which provides the user not only an explanation of the CF process, but the opportunity to manipulate their neighbourhood at varying levels of granularity to reflect aspects of their current mood and requirements. In addition to this, explicit expressions of trust are facilitated in the application's Social Web. *PeerChooser* models relations on the Social Web (in this case applied to a recommender system), and provides an interactive platform for a user to visualise and/or manipulate these relationships. By facilitating interaction during the recommendation process, we overcome the problem of redundant profile information which exists in most current recommenders, in addition to providing an explanation interface. The layout algorithm which acts on the nodes in the interface produces an exact, noiseless graph representation of the underlying correlations between users. PeerChooser's prediction component uses this graph directly to yield the same results as a benchmark ACF technique. User's then improve on these predictions by tweaking the graph to their current requirements. Figure 6.1 provides an annotated screenshot as an overview of the application. This is discussed in detail later in the chapter.

6.2 Motivating Example

In the real world people can usually choose who their friends will be, for example, if someone consistently provides bad advice, then we stop asking that person for advice. This choice is not given to the user in most current CF systems. Imagine you want a movie recommendation and can choose a group of peers in addition to your existing group. You might choose Stephen Spielberg, among others if your in the mood for a Sci-Fi flick. However, what

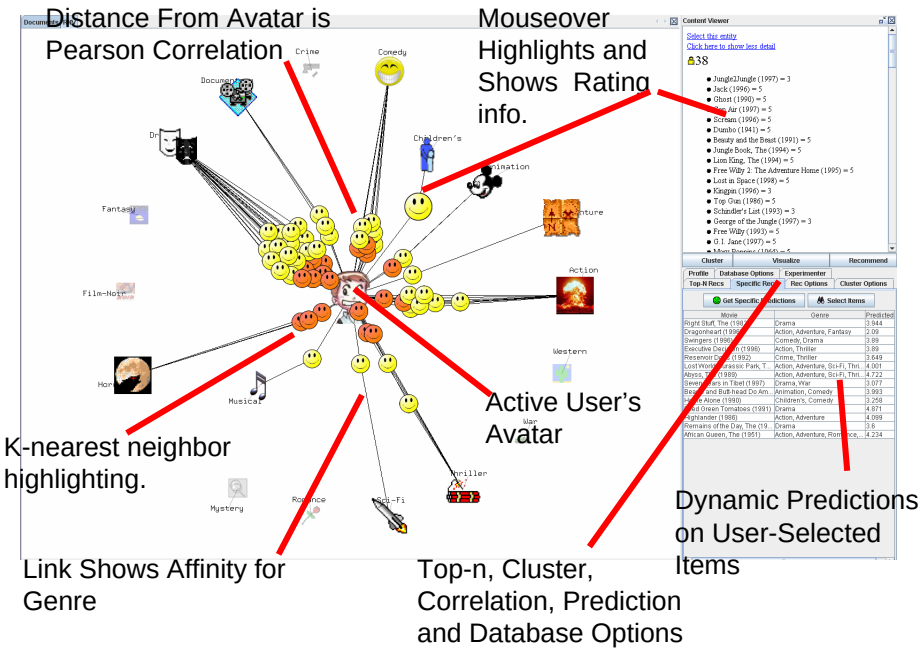

Figure 6.1: Annotated Screenshot of PeerChooser's Interactive Interface.

if you told Mr Spielberg your top 10 movies, and he hated all of them. With this knowledge, would you still want his contributions?

In this chapter it is shown that not only is it possible to manipulate CF peergroups on the Social Web based on dynamic and informative feedback, but doing so improves the user's experience and recommendation quality. The process is fast and more importantly fun for the end user. Dynamic feedback is achieved in *PeerChooser* by allowing a user to see what their existing peergroup would recommend as the user's top-n favourite and least-favourite items. User's can then tweak the neighbourhood graph to tune the predicted ratings on their movie list to the required level. In fact, an existing profile is not essential for the process to work well. By starting at an average position in the neighbourhood-space and manipulating neighbour icons to optimise predictions on the favourite movie list, the user can rapidly generate information upon which the system can base predictions on unseen items.

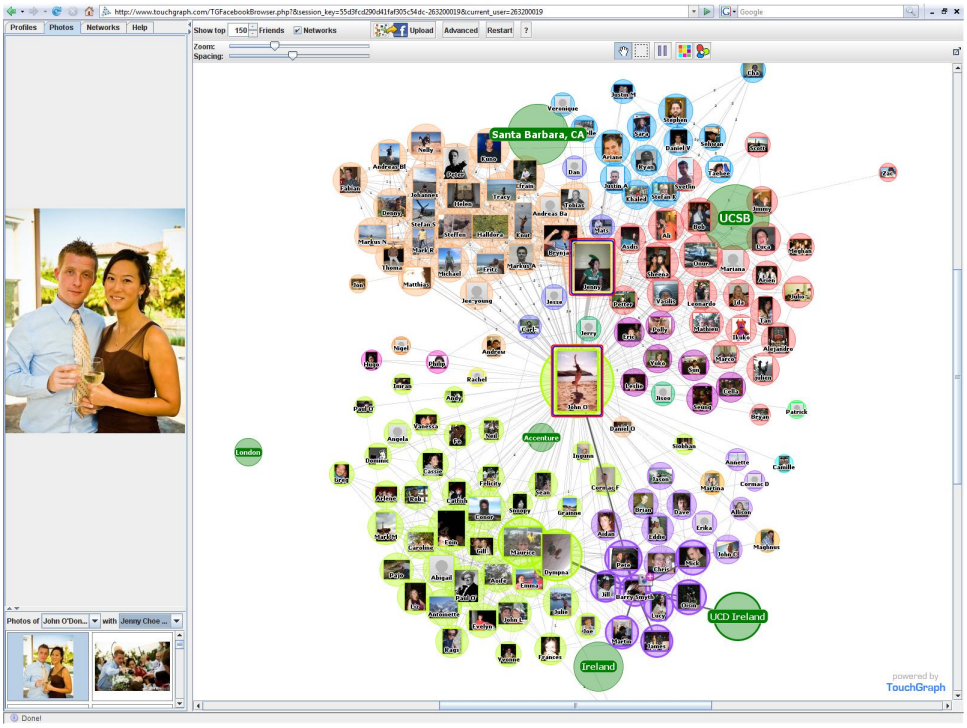

Figure 6.2: The Touchgraph graphing tool deployed on Facebook to visualise the underlying social network produced by photo tagging.

PeerChooser also serves as an explanation interface for CF, affording the user the opportunity to gain an understanding of the core trends in the data, and become more aware of where the final predictions are really coming from. Following from the work of Herlocker et al. in [46], we show that visualisation provides an openness to the recommender system, which in turn enhances overall user trust in the system– something which is lacking in traditional recommender systems, as discussed in chapter four.

Furthermore, *PeerChooser*'s interactive components provide the user with the option of telling the system about current requirements. Mood and persona are important factors to be considered in the recommendation process [67], but they are also highly ephemeral phenomena in the context of the web, making them very difficult to profile. For example, a user may purchase books on Egypt prior to a vacation, but not want recommendations of books on ancient Egypt 6 months later.

An overview of the design of the *PeerChooser* system is presented, along with a discussion of a range of approaches for peergroup selection. Novel ways to integrate data acquired from the visualisation interface back into the recommender system are analysed. Figure 6.4 shows the basic operation of a standard CF algorithm– build a similarity model then make a prediction based on that model. Figure 6.4 also shows two additional "tiers"– construction of an *ephemeral similarity model* based on the interaction with the visualisation, and computation of a prediction based on the updated model.

6.3 Background

Recent research in recommender systems has seen a shift in the focus, off large scale accuracy metrics such as MAE and precision-recall, and towards the general user satisfaction with the system. For example McNee et al in [74] posit that the "usefulness" of a recommendation should be determined by a users *current* need. Herlocker et al. [46] believe that "usefulness" of a recommendation is as important as accuracy. Ziegler et al construct a similar argument in [122] and analyse the importance of topical diversity within recommendation lists.

In this chapter it is shown how facets from visualisation research have been incorporated into a novel recommender system to improve performance and the user's overall experience with the system. Recent research has focused on developing synergies by applying visualisation an interactivity to existing applications, for example the application of friend-visualisation on the social networking web site Facebook [64]. (See Figure 6.2.)

6.3.1 Example Deployment Scenario

Figure 6.3 shows a sample deployment of the *PeerChooser* system on Amazon.com. This figure illustrates the simplicity of the interface in its intended environment, as opposed to the current research-based implementation. In Figure 6.3 recommendations are presented as images, possibly with some associated descriptive text and/or confidence levels. These images and confidence levels are updated dynamically as the user manipulates the person-

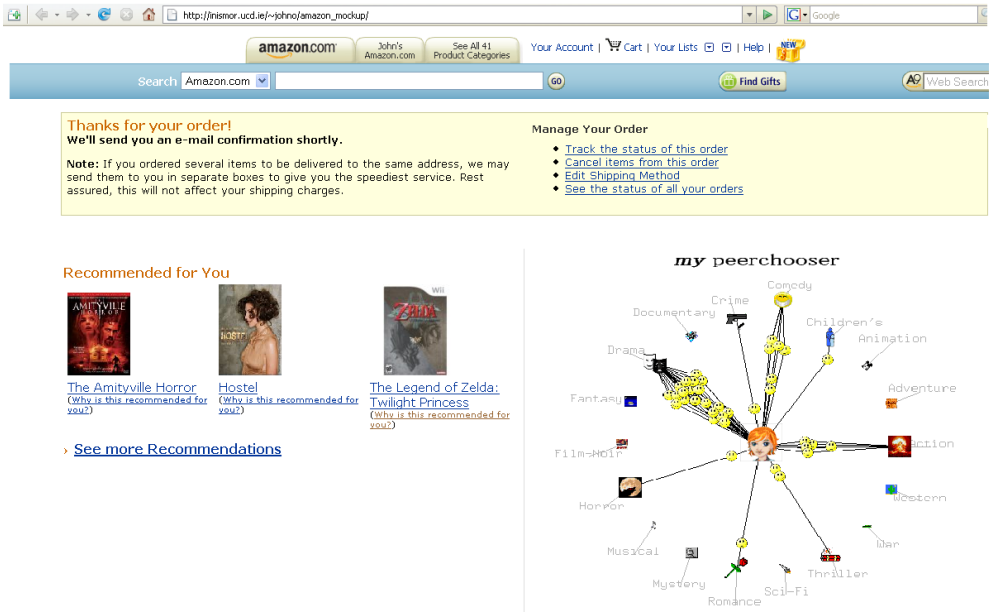

Figure 6.3: Prototype of a PeerChooser interface deployed on Amazon.com

alised graph on the right side. The avatar in the centre of the graph is personalised to the end-user.

6.3.2 Visualising the Social Web

The recent boom in the popularity of Social Web applications has stirred research interest into new ways to gain information about the underlying data in these systems. In a recent work Ciccarese et al. [95] introduce *GENIUS*, a graphical tool for visualizing genetic networks composed by a large number of genes. The system uses two popular visualisation tools, Touchgraph* and Apache Agora [81]. The focus of Ciccarese's work is the examination of ways to simplify exploration of relationships that exist in a network by using various clustering and visualisation strategies such as connection strength, i.e. varying edge values on network graphs, and multiple dataset visualisations by visually intersecting largely heterogenous datasets via a few interconnecting nodes. Ciccarese's approach of using multiple modes of visualisation is followed later in this paper when we describe our approach.

*www.touchgraph.com

The Agora visualisation tool [81] used in [95] operates on the assumption that every node can be treated as equal. By making this simplifying assumption, the underlying mathematical model becomes less complex and the application can easily scale to large numbers of nodes. The Touchgraph [117] model uses spring forces to direct the graph, resulting in vastly different and usually more interesting layouts at the cost of higher complexity and hence less scalability. Ciccarese's work in [95] details many interesting visualisations but lacks empirical evaluation on a specific application.

A very relevant work by Koshman [60] reports on the application of visualisation techniques to an information retrieval system, and more importantly, presents an analysis of user versus system-generated similarity values within the visualisation. We present a similar analysis in our evaluation section. Koshman performed a survey to assess users' opinions on the system, finding only 52% satisfaction with the visualisation approach to information retrieval (IR), 23% uncertain about their use of color and 41% unsure about the relations in Koshman's visualised IR system. However, people tend to be more comfortable with familiar methods and there is a natural familiarisation period in which users opinions on the system may change, so these initial results may not be definitive. We present a detailed user evaluation in Section 6.5 which correlates with the results in [60] in that users tend to prefer familiar systems.

Work by Ware et al. in [49] highlights the fact that visualisation of information relies heavily on aspects of preattentive processing which is lacking in text-based communication modes, this allows for faster and easier interpretation of data.

Middleton et al. present the *Foxtrot* system in [76]. Foxtrot addresses the problem of recommending research papers to users and employs a visualisation-based approach. A major contribution of the work in [76] is their year-long user evaluation assessing the benefits of adding visualisation and explanation to recommendations against a traditional approach to the recommendation task. In the experiment, 200 users received several recommendations every day for one year. One test group were shown visualisations along with the base recommendation, and the other were not. Middleton's core finding is that visualisations help the user to understand and accept the predicted

rating. Similar to the work in this paper, [76] explores the use of bootstrapping the recommender system using visualisation to overcome inherent recommender system problems such as sparsity and cold-start[92].

6.3.3 Visualisation and Explanation

Through visualisation we are creating an "explanation interface" for our recommender system. Some research has been carried out into the effects recommendation explanation has on the overall user experience with the system. A prominent work in this field is Herlocker's study of recommendation explanations [46]. Herlocker et al. evaluate a "white box" conceptual model of recommendation as opposed to the run-of-the-mill black box approach. They present a user study where 21 different recommendation interfaces are presented to users, explaining various types of information from within the recommender algorithm. Herlocker found that users preferred viewing histograms of their neighbours' ratings for the recommended item over all other approaches. Their general finding agrees with Middleton's findings [76] in that "explanation interfaces lead to improved acceptance of a predicted rating."

More recent work by Freyne et al. explores the value of explanation in the web-search domain. Their work in [30] presents a social search application in which the "influence of community wisdom" is presented and used to explain ranking in a search engine. Results from their user survey show that users generally find the social visualisation both interesting and relevant.

This brief examination of the related work shows that a synergy can be achieved by combining information filtering algorithms with visualisation tools to create an "explanation interface". However, the application we now present serves not only as an explanation, but as a fully interactive *control* over the neighbourhood data upon which our algorithms operate.

6.4 Visual Collaborative Filtering

We argue that in addition to the standard similarity assumption in ACF algorithms, users can benefit from incorporating facets of their current *mood* or

persona into the recommendation process. *PeerChooser*'s interactive neighbourhood graphs afford the user the opportunity to express current requirements by seeing how similarity and trust are represented within the recommender system, and to modify this representation where they see fit. Consider the following: A user might want a recommendation on a Sci-Fi related product, and feel that Steven Spielberg should be trusted, but this trust should be lowered if the user was seeking a recommendation on fashion, for example. An important distinction must be made between our definition of a user's current requirements such as mood and persona, and trust. In the algorithms presented in previous chapters, trust has generally been modelled from historical data and then combined with similarity in some way to improve recommendations. In the *PeerChooser* system presented here, the feedback that users provide by tweaking an ACF interface during recommendation takes the place of the trust values in previous algorithms. Trust information computed using the previous algorithms is displayed on the *PeerChooser* interface, and users can manipulate this data during recommendation, but the concept is entirely different from the algorithms previously discussed.

Before presenting a discussion of the actual visualisation of the CF process, we will discuss the high level architecture of the system. Figure 6.4 presents an outline of the *PeerChooser* architecture. User rating data is correlated to produce a model of similarity. Again, Pearson's correlation function is used to compute correlations between users. The resulting similarity model, ratings, and item genre information from the database are passed to a graph generator component. This is essentially a clustering algorithm which generates an XML representation of a personalised graph for one "active" user. The system can compute a range of representations, as discussed in Section 6.4.1.

Our application focusses on the way that CF systems identify user communities or *peer groups* as the basis for recommendation. Very often this is opaque to the users of a system. We suggest that users need more control in order to tailor recommendations to their current moods and influences. Accordingly we show how users can influence peergroup formation, by manipulating a visual representation of their current neighbourhood, in order to fine-tune their recommendations. Figure 6.4 shows the architecture of *Peer-*

Chooser with the feedback mechanism in the shaded area. At each tuning step, users are provided dynamic feedback in the form of recommendations on items well known to them.

6.4.1 Architecture

Figure 6.1 shows a visualisation of a CF process in *PeerChooser*. The main display includes the peer-graph which is centred on the active user. The active user's neighbours are represented by the connected nodes such that the length of the connection between the active user and a neighbour is based on the similarity of their ratings profiles (using the standard Pearson's Correlation Coefficient). The precise positioning of nodes is governed by using a force-directed graph layout technique. This models node connections as 'springs' which exert a force on their nodes that is inversely proportional to node similarity and the systems settles to a zero-energy equilibrium state as the graph is visualised; in practice we have found this to provide accurate graph layouts in which on-screen distance is closely correlated with node similarity, thus providing the system user with an accurate visual representation of their peer group or neighbourhood.

In this work, we are interested in allowing a user to manipulate a neighbourhood in order to allow then to benefit from improved recommendations that more accurately reflect their current preferences and mood. To achieve this the interface must convey meaningful information to the user about their neighbourhood structure and the interests of their neighbours. This is a challenging problem as each neighbourhood represents a high-dimensionality preference space which is difficult to render and even more difficult for a user to interpret. To address this, while maintaining ease of use for the end user, we have chosen to map the complex ratings data onto the space of movie *genres*, which are presented as a set of user-connected *genre nodes* alongside their neighbourhood representation; see Figures 6.1 and 6.5. An edge is created between a genre node g and a user u if u displays an affinity for g in their ratings; in practice we use a simple thresholding technique to identify affinities between users and genres if more than 10% of the user's positive ratings reflect a given genre. These genre nodes are also displayed using the

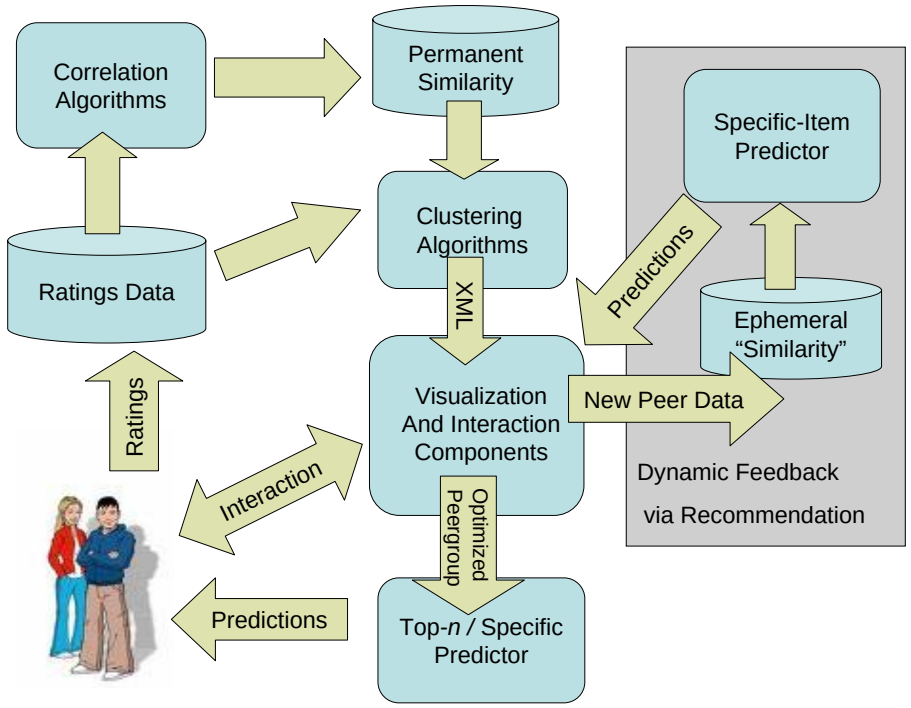

Figure 6.4: Architecture of the PeerChooser Visual Interactive Recommender System.

force-directed graph layout algorithm, which causes a natural clustering of users to genres as shown in Figure 6.5; in this example we allow each user to be connected to their two most dominant genres but in principle each user can be connected to any number of genres, subject to presentation clarity issues. *PeerChooser* provides highlighting functionality for nodes that are salient in the CF process. The k-closest nodes to the active user are highlighted in red. Any selected node is automatically scaled up to distinguish it from other nodes. Multiple node selection is enabled for group manipulation of neighbours. When a node is selected in *PeerChooser* the associated rating detail for that user is shown in a table. Genre nodes in *PeerChooser* are linked to an icon database to represent each genre clearly.

Labelling in *PeerChooser* is done on a per node basis. The label-positioning algorithm ensures that labels will not overlap on the graph and it attempts to place labels towards the outer edges of the graph wherever possible. Labels can be enabled or disabled on the basis of node type. For most of the

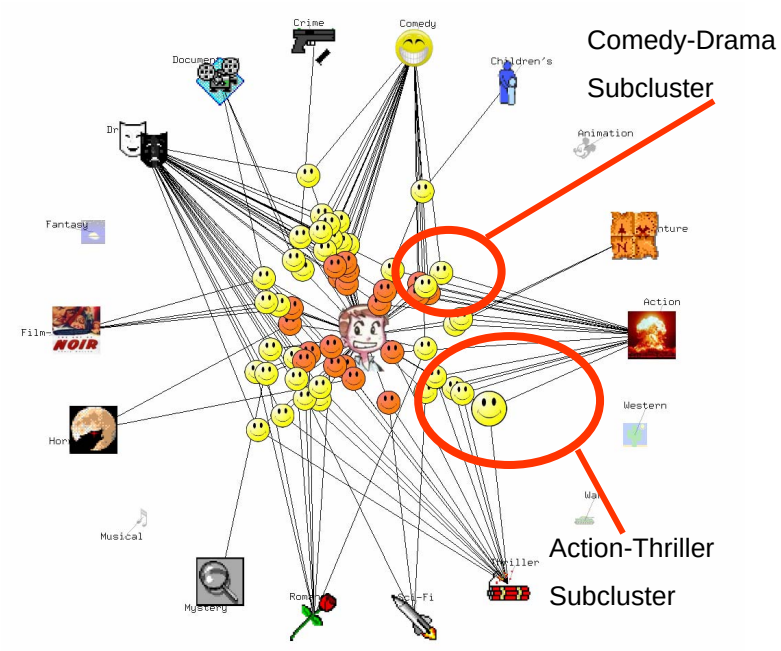

Figure 6.5: MovieLens data visualised using *PeerChooser* with multiple genre associations.

visualisations in this paper we have displayed labels only on the genre nodes.

PeerChooser uses OpenGL technology on a Java platform, making it visually appealing to the end user. When the spring forces come to rest, the active user can interact with the graph by moving or deleting icons. Genre nodes are positioned around the outside of the graph and can be clicked on and moved freely. When this occurs, each connected neighbour node is moved by a relative distance in the same direction. This allows the user to make a bold statement about her current opinion on a particular genre. For example, if the comedy icon is moved towards the active user's avatar then all users who like comedy will be drawn closer and become "more similar" to the active user for this session. We term this new value *ephemeral similarity*. Furthermore, users can make fine-grained statements by moving neighbour icons individually. On mouseover, nodes are highlighted and associated ratings are displayed in the right panel, shown in Figure 6.1.

6.4.2 On the Fair Representation of Genre Information

The genre distribution in the MovieLens data [78] has a massive bias towards three genres: Drama, Action, and Comedy. As a result, our initial clustered graphs tended to only show connections between non-active user nodes and nodes representing these three genres. To counter this problem and provide a more diverse set of neighbours we apply a scaling function to our edge drawing threshold based on number of occurrences of a genre in the database. Equation 6.1 shows how the system computes user to genre connectivity on a per-user basis. In Equation 6.1 G is the set of all genres, g represents one genre. Term $U_{liked,g}$ indicates the number of items user U liked in genre g. U_{total} represents the total number of ratings by U; g_{total} is the number of movies in genre g and c is a scaling constant.

$$Max_{(g \in G)}(\frac{U_{liked,g}}{U_{total}} + \frac{U_{liked,g}}{g_{total}} \cdot c) \tag{6.1}$$

Once our data has been clustered based on the algorithm described in the previous section, the graph is saved as an XML file. The advantage of using this format is that graphs can be saved and loaded repeatedly, meaning that the personalised recommendation graph is portable between different deployments of *PeerChooser*.

6.4.3 Visualising Trust Relations in *PeerChooser*

In addition to providing an explanation of the recommendation process to the end user, *PeerChooser* enables the user not only to visualise *correlation*, but also the *trust-space* generated from the underlying ratings data.

Following from the trust-mining experiments in Chapters 4 and 5, the trust matrix built on the MovieLens dataset was incorporated into the visualisation mechanism as a means to provide at-a-glance information to the end user on *trust* in conjunction with *similarity*. For this experiment, trust was computed using the algorithms from Chapter 4, and similarity was computed in the usual way, using Pearson's correlation over the raw rating data.

Figure 6.6 shows the *PeerChooser* application displaying trust and correlation in parallel. In this personalised graph, the active user is positioned

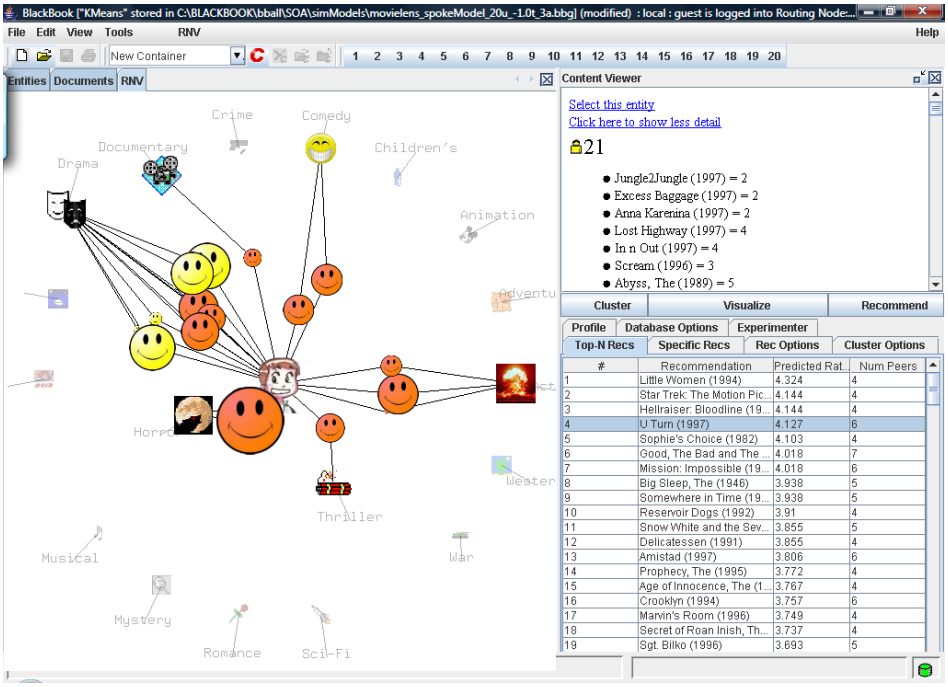

Figure 6.6: The *PeerChooser* OpenGL application showing trust values as node size and correlation as edge length.

at the centre with a personalised avatar. Non-active nodes are positioned around this, again with edge length fixed proportional to the Pearson correlation between the users. Node size is a function of the trust- smaller icons are less trustworthy and larger ones have higher trust. Using this graph the active user can easily discern the differences between similar users and trustworthy users. In this example, a highly trusted neighbour can be seen just below the active user's avatar. This neighbour is also highly correlated due to the close proximity to the active user node. Attached to the trusted peer is an icon representing the "horror" genre. In the right panel the top-n recommendation list contains the slightly esoteric movie "Hellraiser" in the top three, with a prediction of 4.1 from 5. This is most likely the influence of the trusted and correlated peer who likes horror movies.

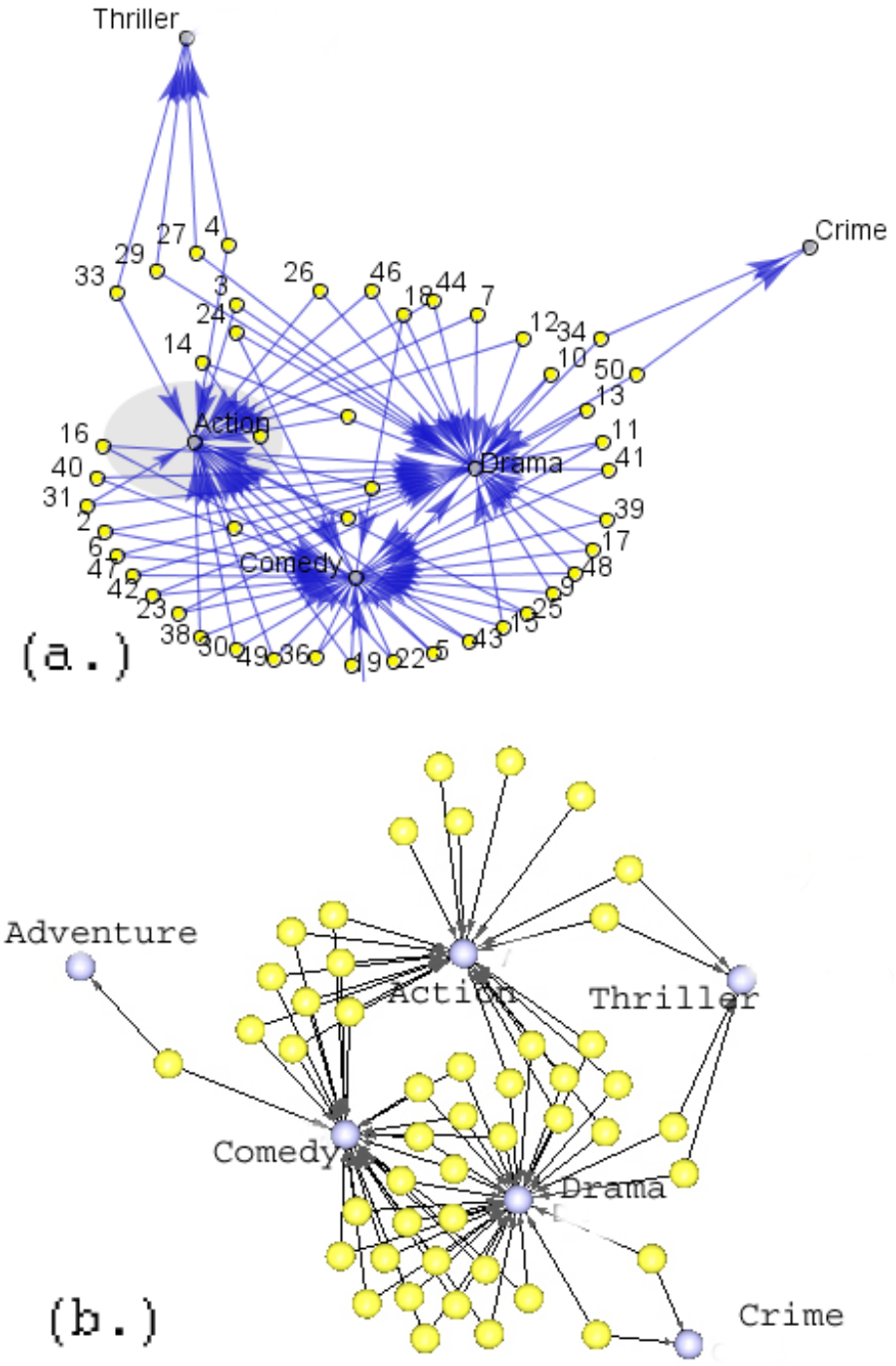

Figure 6.7: MovieLens data visualised using SNV (2D) and DNV(3D) OpenGL-based Visualisation Tools

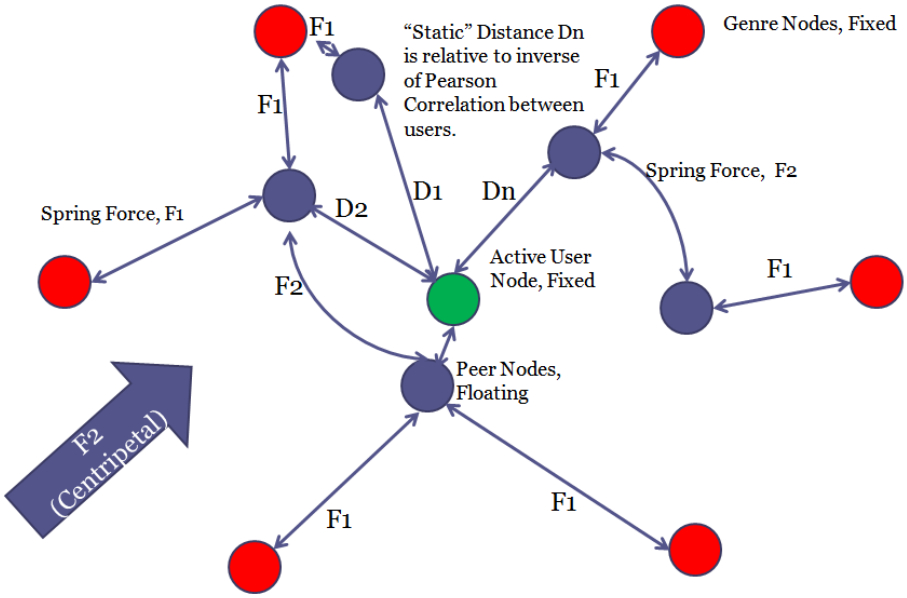

Figure 6.8: Node interactions and forces in RNV

6.4.4 Implementation

Three different attempts were made to produce an interactive visualisation for ACF, firstly a 2-dimensional graphing tool was used to plot our data, and secondly a 3-dimensional approach was used. However, the layout algorithms in both of these approaches (described respectively in Sections 6.4.4 and 6.4.4 below) introduced too much noise to accurately base ACF predictions on the laid-out graph of the underlying data. The current *PeerChooser* interface, which we call Recommender Network Viewer (RNV) was developed to address the problem of noisy graph layouts for collaborative filtering visualisation. See for example Figures 6.6 and 6.1. Section 6.4.1 has already described the layout of this graph in terms of node types, edges, and their respective roles in the visualisation. Now we provide some detail on the underlying implementation of the dynamic graph in RNV.

RNV Visualiser

RNV uses a simple force-directed spring layout algorithm [117]. Figure 6.8 provides an overview of the node interactions in the layout algorithm. Edges

between all nodes are represented as springs. All the springs have a natural length, measured in the same units as the screen co-ordination system, i.e: pixels, which they attempt to achieve constantly. Force $F1$ operates between genre nodes and non-active user nodes. Force $F2$ acts between the non-active users nodes and prevents them from collapsing in on each other. There is a constant centripetal force that prevents nodes from drifting too far from the center. The key point of this layout is that the distances $D1, D2, ..., Dn$ are static and cannot be changed by other forces in the algorithm. Only mouse movements from the user can change these distances, which represent the correlation between the active user and the individual peer nodes. RNV is a Java/OpenGL application and can automatically detect screen resolution and adjust its pixel map accordingly. If a spring in RNV is shorter than its natural length it extends, pushing the nodes at either end of the edge apart. If a spring is longer than its natural length it contracts, pulling the nodes at either end together. The force exerted by the spring is proportional to the difference between its current length and its natural length. Nodes linked together tend to form cluster so a repulsive force is also added so there will still be distinguishing space between the nodes. To keep the nodes in the RNV graph from drifting into too large a space, a weak centripetal force is applied to each node. Intra-node forces keep the graph from collapsing in on itself. The lengths are changed iteratively to obtain a well spaced out layout by minimising the total energy. During initial implementation it was found that nodes tended to "quiver", i.e: jump back and forward rapidly between two or more approximately equal resting positions. To address this issue a resting-heuristic was introduced to settle each note once it had minimised its energy to within a threshold. The spring layout, is initialised with a graph of nodes and edges, as shown in the following code snippet, and assigns X/Y locations to each node in the network. The nodes are moved iteratively to minimise the energy.

PeerChooser uses Java OpenGL (JOGL) which is a wrapper library that allows OpenGL to be used in the Java programming language. It is currently being developed by the Game Technology Group at Sun Microsystems, and is the reference implementation for JSR-231 (Java Bindings for OpenGL). JOGL differs from some other Java OpenGL wrapper libraries

in that it merely exposes the procedural OpenGL API via methods on a few classes, rather than attempting to map OpenGL functionality onto the object-oriented programming paradigm. Indeed, the majority of the JOGL code is autogenerated from the OpenGL C header files via a conversion tool named Gluegen, which was programmed specifically to facilitate the creation of JOGL. By using JOGL we harness all of the functionality of OpenGL from within a familiar Java environment. Use of this technology means that the *PeerChooser* interface is enabled to operate in either 2 or 3 dimensional space.

Our initial attempts to visualise ACF algorithms were unsuccessful because the correlations between user nodes were not preserved by the layout algorithms. We initially used two different layout algorithms, SNV and DNV. Sample visualisations from both of these layout algorithms are shown in Figure 6.7, and we now provide a brief discussion of each.

SNV Visualiser

SNV, or Social Network Viewer, is part of the Blackbook suite of visualisation tools [51]. It lays out the nodes in 2D and displays a static graph, which the user can interact with. A user can zoom in and out, pan, and move nodes around. Our initial experiments visualising CF passed xml data to the SNV visualiser tool, but the layout algorithm generated noise for the predictions and it proved inapplicable to the task. Figure 6.7(a.) shows our initial attempts visualising the MovieLens dataset in SNV.

DNV Visualiser

DNV, or Dynamic Network Viewer, is a viewer for Blackbook which scales to 10,000+ nodes. DNV was developed by Gretarsson and Hollerer at the University of California, Santa Barbara. DNV can lay out nodes in either 2D or depth-limited 3D using a spring model. The layout is done dynamically, i.e. if the user moves a node around then he or she immediately sees how the whole graph changes shape, based on the spring model. We carried out preliminary CF visualisation tests in DNV and found that users preferred to move genre nodes rather than user nodes to manipulate the graph. Following

from these tests we developed a CF-specific visualisation tool, *PeerChooser* based on the original DNV model. Figure 6.7(b.) shows a sample DNV graph of our MovieLens visualisation. The viewer for the *PeerChooser* interface, RNV, (shown in Figures 6.1 and 6.5) was based on the original DNV implementation.

6.4.5 Tweaking ACF Recommendations

In the previous section we focused on PeerChooser's visualisation interface, which allows the user to manipulate their direct neighbours and genre preferences; these manipulations are the user's *hints* for the recommender. Now we explain how these hints translate in to actual recommendation influences. In *PeerChooser* the k nearest user nodes are selected as the neighbourhood for the purpose of recommendation; these are the more shaded nodes in Figures 6.1 and 6.5. Our benchmark CF prediction algorithm again uses Resnick's formula which we have seen in earlier chapters as Equation 3.1.

To incorporate user hints into the recommendation process we simply replace the standard similarity values (based on user-user ratings correlations) with a new similarity value that is based on the inverse *Euclidean distance* between the active user node and each of the k peer nodes that have been manipulated by the user. This is our *ephemeral similarity* value and is given by Equation 6.2. Here, Euclidean distance between pixels on the graph is normalised to the Pearson's correlation range of (-1, +1), max_dist is the maximum possible distance between the active user node and a peer node, while $node\_dist$ is the distance between the active node (i.e: the centre of the graph) and each peer node. Equation 6.3 shows the original Resnick prediction formula using ephemeral similarity in place of the standard Pearson correlation. The nomenclature is similar to that in Equation 3.1 with $c(i)$ being the predicted rating for an active user c on item i.

$$eph\_sim(c,p) = 1 - \frac{2(node\_dist)}{max\_dist} \tag{6.2}$$

$$c(i) = \overline{c} + \frac{\sum_{p \in P(i)} (p(i) - \overline{p}) eph\_sim(c, p)}{\sum_{p \in P_i} eph\_sim(c, p)} \tag{6.3}$$

6.5 Evaluation

The majority of experiments involving recommender system algorithms are based on some form of automated testing, for example, predicting ratings for some "hidden" subset of the rated data. This is only possible in absence of interactive components such as the ones of our *PeerChooser* system. Visualisation and interaction are additional "tiers" to the process of collaborative filtering. The visualisation and interaction tiers bring many obvious advantages, however, it does prohibit the standard automated testing procedure since real people must explore and interact with the graph to attain results.

To evaluate the interactive visualisation components of the system, we performed controlled user trials. The objective of the trials was to ascertain as much information as possible about the users' overall experience using the visualisation techniques for profile generation and ratings prediction.

6.5.1 Experimental Data

For all of the evaluations in this chapter the small Movielens* [78] dataset was used. The base dataset consists of 100,000 ratings from 943 users on 1682 items. The dataset has a sparsity of 93.7%.

The dataset contains no demographic information, but does contain genre information for each movie. Movies can be associated with multiple genres. Data is stored in a relational database and cached into memory where possible.

*http://movielens.umn.edu

6.5.2 Rating Distributions

During user evaluations with our system two important comments were noted. Firstly, a 21 year old pointed out that he didn't know many of the movies in the dataset, and secondly, a 30 year old stated that there were a lot of "classics" in the movie list he had to rate. These were worrying comments- it was feared that users might only remember the better, more popular movies, and accordingly produce an unnaturally high rating scale. To assess this possibility we performed a comparative analysis between the distribution of dataset ratings and those from our user survey. Results from this analysis are presented in Figure 6.9. There is a definite increase in the number of ratings in the top bin, but we were happy to find that the difference in the overall rating trend was not significant ($p<0.01$).

6.5.3 Procedure

Our study involved 30 users, ranging in age between 21 and 30 years. Each trial took approximately 30 mins and consisted of three stages, a pre-study questionnaire in which participants provided general demographic information, and information relating to their experience using recommender systems and graph visualisations. We also asked the users to rank a list of movie genres according to their perceived popularity.

The second stage of the user trial required the participant to use the system to get recommendations.

Firstly, users were required to perform a familiarisation task. This involved users getting a recommendation for a movie from the list, interacting with the various user nodes and genre nodes and manipulating the predicted rating to their taste. For the next task in the trial participants were required to use *PeerChooser's* manual rating window to generate a ratings profile via explicit ratings. Users were asked to rate 30 movies that they had watched on a scale of 1 (strong dislike) to 5 (strong like). Users then clicked on a submit button and the system correlated their ratings profile with the existing profiles.

To test the performance of our system we examined four techniques for generating recommendations, two with interaction and two without. For

each approach we collected three values per item: the predicted rating, the actual rating, and the average rating in the whole database. The following list describes each approach:

1. *Average Layout* - The graph is laid out based on an "average user". We created this user by taking the average rating for the top 50 rated items. This technique was expected to yield the worst results as it contained no personalised information.

2. *Profile-Based Layout* - This is the benchmark CF algorithm. Correlations computed from a ratings profile are used to generate predictions.

3. *Profile-Based Layout with Manipulation* - Same as above but the user can manipulate the graph to provide information on current requirements.

4. *Profile-Based Layout with Manipulation and Feedback* - Same as above except the user receives dynamic recommendations on their salient items with each graph movement. We expected this to demonstrate the best performance.

For the *Average Layout* task users did not interact with the graph. They were simply shown a list of recommended items and asked to rate them (without showing them the predicted rating). Following from work by Swearingen et al. in [112] the predicted ratings were not displayed during this test as they could effect the users input. (Swearingen suggests that users tend to rate towards the machine-provided ratings.) Users then clicked on a submit button and the ratings were recorded.

A graph was then generated using our Pearson Clustering method described earlier. The cluster algorithm had a liked-item threshold of 3, meaning that the system only assumes an item is liked if the rating is 4 or 5. The number of neighbours to use in the computation was set to k=30 and the total number of users displayed on the graph was 400. The algorithm took the 400 most similar users from the entire database to display on the graph. The user then clicked on a button marked "visualise" to show their personalised graph on the *PeerChooser* visualiser.

Users were then told to click on a button marked "recommend" to a list of recommendations based on their personalised graph. Since the visualisation algorithm has zero effective noise for the collaborative filtering process, these recommendations are exactly equivalent to the benchmark Resnick CF algorithm from [102]. As with previous tasks, users rated 20 items and recorded their info. The penultimate recommendation task was similar to the above but with the addition of user manipulation on the graph.

In the final task the graph layout was as in the previous task. Users were asked to click checkboxes next to 5 movies they really liked and 5 that they hated. Users were then told to click on a button marked "specific predictions" and were provided with the benchmark CF predictions on those items. Users were then told to take some time to manipulate the graph to try and tweak the recommendations for their chosen items to a value that most suited them. All users in the survey reported that they had arrived at a satisfactory position within 2 mins without notification of a time constraint. At each movement users were provided with dynamic recommendations on their "salient" item-set according to the current neighbourhood configuration. This allowed the users to dynamically assess the goodness of each movement performed. Users were again presented with a list of recommended movies and asked to rate them and store the information.

The concluding part of the user trial was the post-study questionnaire. This contained the important questions regarding the users overall experience with the interface. Results are shown in Figure 6.11. Plotted are (as ratings between 1 and 5): Subjective accuracy of the different methods as perceived by the user (not based on actual prediction comparisons), the understandability of the used graph, user ratings for the four techniques in Figure 6.10. The next question is important: Would the user prefer a visualisation-based approach over the standard active ratings-based technique in real life CF applications? The answer was a resounding 3.75 on average, one of the highest values in the survey. Likewise high ratings were given for the questions if the users gained knowledge through their interaction with the visualisation, the clarity of the labelling, informativeness of the right-hand information panel, desirability of applying this technique in other domains, and usefulness of interaction (highest value of all). Only on the question of how much control

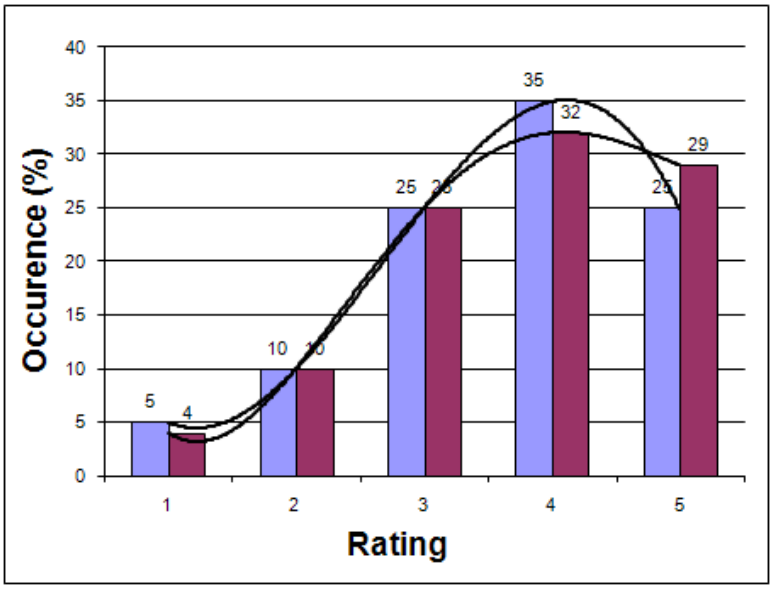

Figure 6.9: Comparison of ratings distributions between existing MovieLens data and data from the user trials.

the users felt they had, and on the perceived accuracy did the survey yield ratings below average.

Also, during this final questionnaire users were asked once more to rate a list of popular genres, as in the pre study to assess if there is any learning effect from use of the visualisation, which turned out to be the case. As but one example, on the post-study form, every user listed Drama, the most popular genre in the list of 5 most popular genres. On the pre-study form, only 8 out of 13 users had mentioned Drama.

6.5.4 Recommendation Accuracy

To evaluate the accuracy of the techniques, mean absolute error was computed between the predicted rating and the users actual rating for each of the methods. Results of our analysis are presented in Figure 6.10 for four techniques. As expected, the average predictions– that is, predictions based on the average user described earlier, exhibited the worst performance, producing a an accuracy of 80.5%. Predictions based on an average user (column 1 in Figure 6.10) have high accuracy, this is not surprising if we take a look

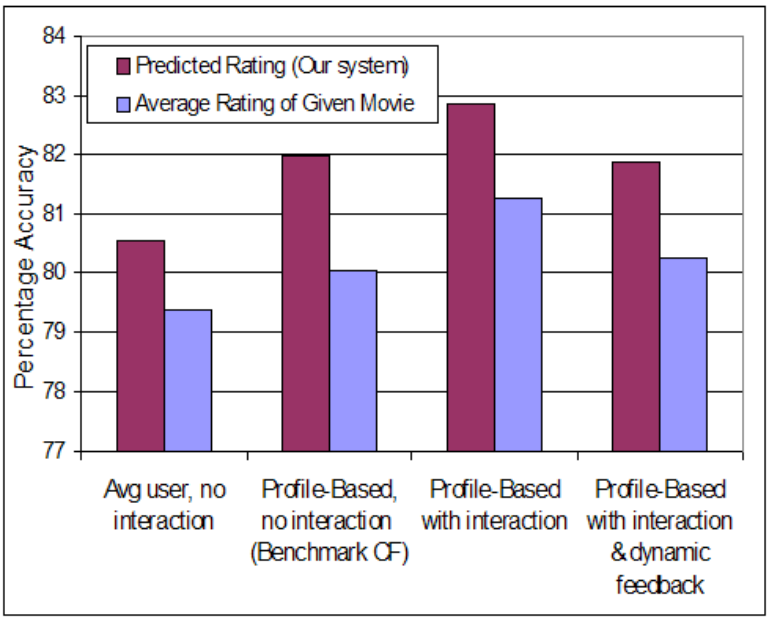

Figure 6.10: Accuracy results for each technique in the user trials.

at the rating distribution graph in Figure 6.9. Users tended to rated only movies they liked, and the average user was constructed from a list of the most commonly rated movies, which as it turns out were generally the most highly rated movies. This may be attributed to the fact that the movies were not new and users tended to remember them in a good light. Our profile-based technique (column 3) with manipulation beats the benchmark (column 2) achieving a small relative increase of 1.05%. A single factor between groups ANOVA shows that these differences are significant in each case with $p = 0.006$, $F = 3.87$. This small increase is an important result because it indicates that manipulation does increase recommendation accuracy.

The most surprising result was that the dynamic feedback technique performed worse than the other profile-based techniques. After much analysis of the graphs it was determined that users tended to *over-tweak* the system to achieve desired results for their salient item sets. In doing this, many of the profile-based correlations were overwritten and the resulting layout was overfitted to the specific item sets. A solution to this may be to ensure diversity within the salient item sets.

#	*QuestionDescription*
S1	I am familiar with recommender systems
S2	I am familiar with interactive computer graphics
S3	I am familiar with graph visualizations
S4	Which did you think was the most accurate
S5	I found the graph easy to understand
S6	Look at the 4 graphs and rate each one
S7	Did you prefer the visualisation approach
S8	I gained knowledge about the underlying data from the visualization
S9	I felt that the labelling was appropriate
S10	I felt that the information in the right panel was helpful
S11	I felt that the *visualisation* system gave me more control of the recommendation algorithm
S12	Would you like to see this interface on other domains (e.g: Amazon.com)
S13	I felt that the icons communicated the genres appropriately
S14	I felt that the *visualisation* system gave me more control of the recommendation algorithm
S15	I felt that I benefitted from *interaction* with the system

Table 6.1: Likert Scale Questions from the User Survey.

6.5.5 User Satisfaction

To assess the effects of users interaction with the system a pre and post study questionnaire was answered by each participant. Table 1 lists each question from the survey and references the columns in Figure 6.11. $S1$ to $S3$ indicate that all participants had experience with graphical interfaces, recommenders and visualisations. $S4$ tells us that, in contrast to our empirical accuracy tests, users felt that manual rating provided more accurate results. This was an interesting result which may indicate that users are more comfortable with familiar, manual rating systems. The four columns marked $S6$ represent participants' opinions on a range of different graph representations of the data, with column 8 being an exceptionally poor display. These are available for

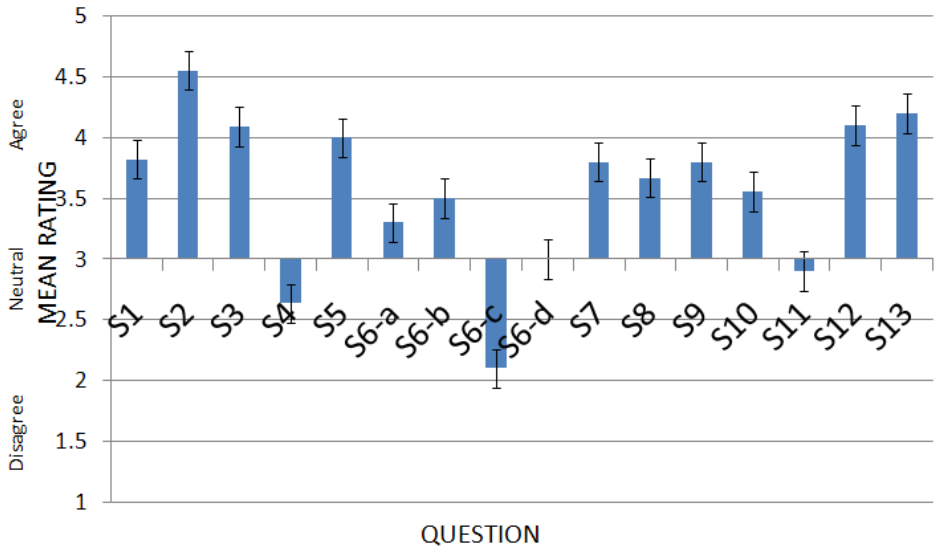

Figure 6.11: Results of the Questionnaire from the User Trials.

viewing at the URL above. $S7$ shows a clear preference for the visualisation approach, with 75% of users preferring *PeerChooser* over the traditional approach. The benefit of our technique as a recommendation *explanation* is shown by $S8$, where the majority of users felt they gained knowledge of the data from their interaction with the graphs. $S9$ and $S10$ indicate that users felt that labelling and node information were appropriate. $S11$ shows us that there were mixed views about the control that the interface provided on the CF algorithms. This response may be due to insufficient familiarisation time, since the technique does provide more access points to influence the CF process. $S12$ shows that more than 80% of participants agreed that they would like to see a *PeerChooser*-like interface on other domains such as Amazon.com. This is an encouraging result, given that there is a broad scope of applications for our technique. More importantly, $S15$ shows that more than 80% of participants felt they benefitted overall from interacting with the system.

6.6 Conclusions

Traditionally collaborative filtering systems have relied heavily on similarities between the ratings profiles of users as a way to differentially rate the prediction contributions of different profiles. We have shown how interactive visualisation techniques can be used in a standard collaborative filtering algorithm to greatly expedite the process of profile generation, helping to ease the "cold start" problem that exists in most collaborative filtering systems. [92]. Visualisation enhances the user experience by providing a user with at-a-glance information about the underlying structure of the trust and correlation data upon which predicted ratings are generated. Interaction with the visualisation affords the user the opportunity to express facets of current mood and requirements.

This chapter introduced and evaluated our visualisation-based approach against a benchmark collaborative filtering algorithm using mean error accuracy tests. A live user survey was carried out to assess a range of aspects of user interaction with the system from a HCI and predictive accuracy perspectives. Accuracy experiments indicate that there is a benefit to the visualisation approach to profile generation for collaborative filtering over each of the four techniques tested, more importantly however, by viewing trust values between individual peers, users felt that the system provided an *explanation* of the recommendation process, which led to increased overall trust in the recommendation system.

Several additions to the existing system are currently in progress. The Netflix prize* dataset consists of very recent movies and is a rich resource for ratings information. This dataset is currently integrated into the *PeerChooser* system. However, the publicly released data does not contain genre information for the items. We are in the process of implementing a web crawling algorithm to automatically tag each movie with its appropriate genre information, in a similar manner to MovieLens, so we can use *PeerChooser* on the data. An online version of *PeerChooser*† is planned (using Java WebStart), to test the system over much larger user trials. Another

*www.netflixprize.com

†www.johnodonovan.net/peerchooser

avenue to explore is the possible application of *PeerChooser* to the online auction domain. Chapter 7 documents our trust mining algorithm for eBay. *PeerChooser* would be very suited as a visualisation tool for this data, with the "genre nodes" mentioned in this chapter replaced by categories of goods from the eBay data discussed in the next chapter.

This concludes the research on trust based recommender systems. The following chapter introduces trust metrics into another popular Social Web application: online auctions. In the chapter a new algorithm for mining trust values from online auction feedback comments is presented. In keeping with the trust-mining techniques presented in earlier chapters, these trust values are built at a general level and a context-specific level (similar to *Item-Level* and *Profile-Level* trust.) The algorithm is tested on real data from eBay and evaluated against popular existing classification algorithms. The chapter deals with the possible methods for personalising the generated trust values through a propagation mechanism to achieve personalised trust for potential transactors on eBay who have not interacted directly in the past. Chapter 7 also presents a technique for visualising the underlying trust network that exists in the online auction, so the end user can gain an overview of their personalised *trust-space*.

Chapter 7

Extracting and Visualising Trust from Online Auction Feedback Comments

7.1 Introduction

Up to this point, discussion and analysis have focused on trust and its various applications to recommender systems, including their modelling, integration and visualisation. The second Social Web application area for our trust-based experiments the domain of online auctions. In this chapter a technique is presented to harness trust information which is hidden in freetext feedback comments on eBay. Similar to the ideas in the previous chapter, we show how the resulting trust model can be visualised and presented to the end user to provide an enhanced experience with the Social Web application.

Buyers and sellers in online auctions are faced with the task of deciding who to entrust their business to based on a very limited amount of information. For example, buyers in online auctions are forced to trust sellers to deliver goods without the chance to actually see those goods. Sellers, to a lesser extent, must trust buyers to pay, their burden of trust is less because they hold the product until payment is made, so they only stand to lose the time cost of re-auctioning the product. To facilitate this, eBay implements a community-based system of buyer-seller reviews and ratings. However, current trust ratings on eBay average over 99% positive [103] and are presented as a single number on a user's profile. This is a problem because it leaves the

EVENING HERALD WEDNESDAY 13 DECEMBER 2006 33

NEWS

CYBERSPACE FRAUD: 21 ARE CHARGED IN THE US OVER eBAY AUCTION SCAM

PROSECUTORS in the US have charged 21 people accused of swindling users of eBay and other internet auction sites out of $5m (€3.8m) by promising to send them goods they never received.

People who bid unsuccessfully on items including cars and tractors were told they had "a second chance" to buy and were urged to wire the money to pick-up points such as Chicago currency exchanges.

But the buyers never received the items and much of the money was sent overseas, most likely to Romania, according to prosecutors.

Eight of those accused in an 80-page affidavit that accompanied the complaint were arrested and remanded until Friday when they appeared in a Chicago court.

Five other defendants are already in custody, another will be summoned to appear later and the rest were fugitives, prosecutors said.

Most of those arrested were of Romanian origin and a Romanian interpreter translated for one defendant at the hearing.

Victims were typically contacted online by someone who claimed to be the "seller" or the "seller's agent" or an "eBay agent" and told that the successful bidder on an item had not followed through and that they could buy it for the amount they had offered.

INTERNET: Victims conned

Figure 7.1: Recent Media Article on Fraud Committed on eBay Resulting in 21 Prosecutions.

end user with a difficult task of deciding who to trust when almost everyone has high trust scores. This chapter further explores the issue of trust within the Social Web, with a focus on the online auction domain. A system is designed and implemented which is capable of extracting valuable negative information from the wealth of feedback comments on eBay. This information is used to compute *personalised* and *feature-based* trust, and the results of this computation are presented to the user graphically. This system highlights yet another rich source of trust on the Social Web, and another way in which trust can be used to benefit its users.

In 2005, eBay Inc. had an annual growth rate of 42.5% and in February 2006 was receiving over three million user feedback comments every day. [26] This growth reflects the increased access to online markets throughout the world. As a result of this rapid growth rate, users find themselves more frequently in the situation where they must trust a complete stranger to uphold their part of a business transaction. Traditional mechanisms for evaluating the trust of a potential buyer or seller do not apply in the online world, where in many cases the only information provided is a buyer or seller's username and there are orders of magnitude more potential transactors. [104] In this

chapter we address the problem of providing a reliable mechanism for trusting users online, this time within the domain of online auctions. We argue that the current trust system on eBay is massively biased towards positive comments and show how *AuctionRules*, a trust-extraction algorithm can be applied to the wealth of freetext comments to capture subtle indications of negativity, producing a more scaled range of trust based on the sentiment found in comments. To achieve this, the algorithm classifies freetext comments using a small list (less than 30 nouns) of salient features in the domain. Our evaluations show this algorithm outperforms seven popular classification algorithms by up to 21%.

Currently, eBay displays a non-personalised trust score for a potential transactor. We describe a trust propagation mechanism which uses output from *AuctionRules* to generate personalised trust between two potential transactors. For demonstration purposes, we present a replica interface to a standard auction site in which trust between specific users, and per-feature trust can be computed by *AuctionRules* on the fly and visualised as a social network graph with edge length and thickness as functions of both strength and goodness of the trust relationship.

7.2 The Interactive Online Auction Model

The online auction business model is one in which participants bid for products and services over the Internet. The functionality of buying and selling in an auction format is made possible through auction software which regulates the various processes involved.

When one thinks of online auctions they typically think of eBay, the world's largest online auction site. Like most auction companies, eBay does not actually sell goods that it owns itself. It merely facilitates the process of listing and displaying goods, bidding on items, and paying for them. It acts as a marketplace for individuals and businesses who use the site to auction off goods and services.

Several types of online auctions are possible. In an English auction the initial price starts low and is bid up by successive bidders. In a Dutch auction, multiple identical items are offered in one auction, with all winning

bidders paying the same price – the highest price at which all items will be sold (treasury bills, for example, are auctioned this way). Almost all online auctions use the English auction method.

The following list outlines some of the main strengths and benefits of the online auction business model.

1. No time constraints. Bids can be placed at any time (24/7). Items are listed for a number of days (usually between 1 and 10, at the discretion of the seller), giving purchasers time to search, decide, and bid. This convenience increases the number of bidders.

2. No geographical constraints. Sellers and bidders can participate from anywhere that has internet access. This makes them more accessible and reduces the cost of "attending" an auction. This increases the number of listed items (ie.: number of sellers) and the number of bids for each item (ie.: number of bidders). The items do not need to be shipped to a central location, reducing costs, and reducing the seller's minimum acceptable price.

3. Intensity of social interactions. The social interactions involved in the bidding process are very similar to gambling. The bidders wait in anticipation hoping they will "win" (eBay calls the successful bidder the "winner"). Much like gambling addiction, many bidders bid primarily to "play the game" rather than to obtain products or services. This creates a highly loyal customer segment for eBay.

4. Large number of bidders. Because of the potential for a relatively low price, the broad scope of products and services available, the ease of access, and the social benefits of the auction process, there are a large numbers of bidders.

5. Large number of sellers. Because of the large number of bidders, the potential for a relatively high price, reduced selling costs, and ease of access, there are a large number of sellers.

6. Network economies. The large number of bidders will encourage more sellers, which, in turn, will encourage more bidders, which will en-

courage more sellers, etc., in a virtuous circle. The more the circle operates, the larger the system becomes, and the more valuable the business model becomes for all participants.

7. Captures consumers' surplus. Auctions are a form of first degree price discrimination (i.e: each consumer paying the maximum that he or she is willing to pay). As such, they attempt to convert part of the consumers' surplus into producers' surplus. On-line auctions are efficient enough forms of price discrimination that they are able to do this. (Consumers surplus is defined as the area above the market price line but below the firm's demand curve, which is the graph depicting the price of a certain commodity and the amount of it that consumers are willing and able to purchase at that given price.)

It is clear from the above list that there are many advantages to using online auctions over the traditional type, the most obvious being ease of accessibility. However, recent research [104] has shown that users are still much more comfortable with the "old-fashioned" or "corner-store" business model, wherein people can interact face-to-face and trust and reputation can be assessed using traditional methods such as word-of-mouth and human intuition. One of the primary goals of this research is to explore new ways in which we can make online transactions more similar to the traditional "corner-store" experience. We propose that use of trust metrics is a step towards achieving this aim.

7.2.1 Trust in Online Auctions

Resnick highlights some relevant points which affect the current eBay reputation system in [103] and [104]. Buyers reputation matter less since they hold the goods until they are paid. Feedback can be affected by the person who makes the first comment, i.e: feedback can be reciprocated. Retaliatory feedback and potential for lawsuits are strong disincentives for leaving negative comments. Anonymity is possible in eBay since real names are not revealed and the only thing validated at registration is an email address. Users can choose not to display feedback comments. "Unpaid item" buyers

cannot leave feedback [13], and users can agree to mutually withdraw feedback [13]. Furthermore eBay inc admit that negative feedback is removed from newer users of the system [13] All of these points help to explain the lack of negative comments on eBay. Of course this does not mean that customers are satisfied. The eBay forums* highlight the fact that false advertising does occur on eBay. This should lead to more negative comments, but they are not being displayed. Xiong et al. [120] provide further reasoning for the imbalance of positive comments on eBay. Our proposed model of trust for eBay should provide a more realistic scale than the existing system. This is presented in Chapter 6.

Case Study: eBay

The online auction web site was founded in San Jose, California on September 3, 1995 by computer programmer Pierre Omidyar as AuctionWeb part of a larger personal site that included, among other things, Omidyar's own tongue-in-cheek tribute to the Ebola virus. The very first item sold on eBay was Omidyar's broken laser pointer for $13.83.

Founded in September 1995, eBay (Nasdaq: EBAY) is The World's Online Marketplace. eBay is a platform for the sale of goods and services by a diverse community of individuals and businesses. Today, the eBay community includes more than 114 million registered members from all around the world.

Globally, eBay has more than 25 million items listed for sale at any one time, and an additional 3.5 million items are added daily. There are more than 50,000 categories including collectibles, antiques, sports memorabilia, computers, IT and office, art, antiques, toys, dolls, stamps, comics, magazines, music, pottery, glass, photography, electronics, jewellery and gemstones. eBay provides an international platform where practically anyone can trade practically anything. Figure 7.2 shows a screenshot of the antiques section of eBay.

*http://forums.ebay.com

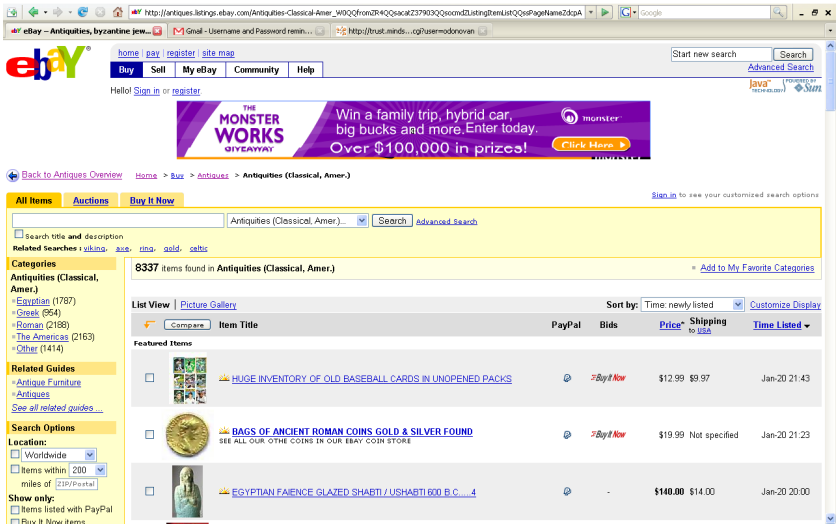

Figure 7.2: Screenshot of eBay Online Auction

Case Study: Yahoo Auctions

Yahoo! Inc. is a global Internet media company that offers a branded network of comprehensive information, communication and shopping services to 60 million users worldwide. Yahoo Auctions is a leading person-to-person interactive marketplace connecting buyers and sellers and is part of the Yahoo! global network. As the first online navigational guide to the Web, http://www.yahoo.com/ is the leading guide in terms of traffic, advertising, household and business user reach, and is one of the most recognised brands associated with the Internet. They are second in popularity in the online auction business only to eBay. Yahoo is headquartered in Santa Clara, California. Figure 7.3 shows a current screenshot of the antiques page from Yahoo auctions. Later in this thesis we present comparative analysis of our trust-mining algorithm on data from Yahoo auctions as eBay for the purpose of demonstrating portability and consistency of the technique.

Online Auction Survey

In order to gain a better understanding of the online auction domain and the potential roles which trust might have within it, a survey was conducted

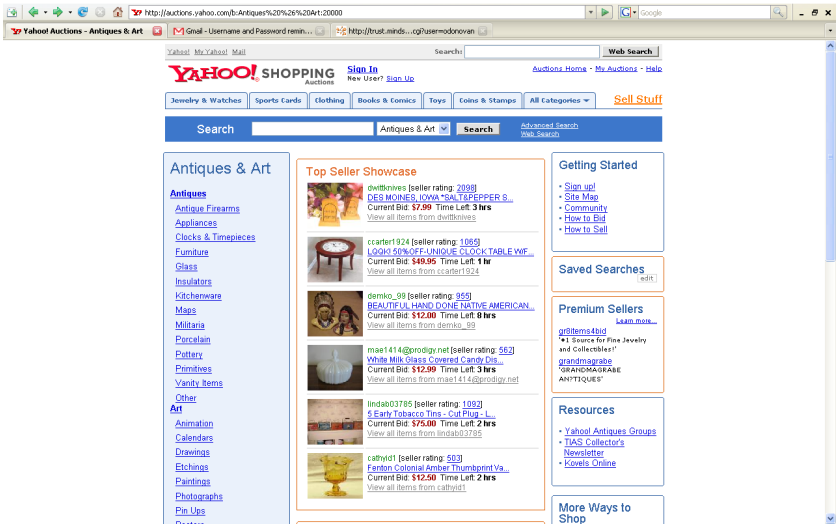

Figure 7.3: Screenshot of Yahoo Online Auction

of 10 online auction/retail sites. Table 7.2.1 shows a summary of the main findings. Full results of this survey are available on the web*. For each site we asked several specific questions relating to the potential applicability of a trust modelling and propagation module to the system.

- What are the user roles in the system. (Buyer/Seller/Both) This will be relevant because each role in an auction transaction commands varying trust issues. For example, a seller does not need to be trusted with respect to payment.
- Is there an existing trust value? (y/n) How is this trust value computed.
- If so, is it personalised (y/n) i.e. does every user get the same trust value for the active user.
- What is the percentage of positive comments? If there are too many positive comments, the site falsely promotes the idea that 'everything is fine'. This can actually lead to more business for the site [26] so in some cases may be done on purpose.

*http://www.johnodonovan.net/research.jsp

- What are the review types (Product/People/Both). For this work we are interested solely in reviews of people, although there is probably information that can also be harnessed from product reviews that can aid in a trust building process.

- What are the requirements for making a rating (Purchase/Registration/None). This question relates directly with attacking of trust based systems. How easy/difficult will it be to skew the system in some way. The cost of making a rating is strongly correlated with the cost of attacking the system for malicious or other reasons.

- What types of sale are provided (Auction/Retail/Both). For this work we are only interested in auction style sites where users can make reviews.

Interestingly, none of the sites surveyed provided any personalisation of trust scores. The trust propagation system required that users could be both buyers and sellers, comments were provided on transactions, and that the comments be mostly genuine. For this reason we enquired about the potential for malicious attacking [88] by assessing the cost of making a rating. We found that our application would be deployable on 7 out of 10 of the surveyed sites. Users have the option of performing buyer or seller roles on 8 of the 10 sites. On 6 of the sites, reviews were made about other users, and on 1 site reviews were on products only. 3 sites allowed reviews of both. An important statistic we gathered was that on average user feedback was over 94% positive across all the sites reviewed. This reaffirms the need for a negativity-capturing algorithm if we are to build a new trust model for use on auction sites which use text feedback comment systems.

7.3 Background

Some background material is required for this chapter as it deals with a slightly different form of trust than discussed in the previous chapters. This brief background is in three main areas. Firstly, we will examine related research in the area of trust and reputation in online environments. The

Site Name	User Roles (Buyer, Seller)	Review Types (Product, People)	Applicable to our Trust Model.	Existing trust value (y/n)	If so, is it personalised (y/n)	Percentage of positive comments	Requirements for rating (purchase, reg, none)	Sale Type (auction, retail)	URL
eBay	Both	People	Very	Yes	No	98	Purchase	Both	eBay.com
Amazon	Both	Both	Very	Yes	No	89	Purch/Reg	Retail	www.amazon.com
Epinions	Buyer	Product	No	No	No	N/A	None	None	epinions.com
TradeMe	Both	People	Very	Yes	No	99	Purchase	Both	trademe.co.nz
Bidz	Both	None	No	No	No	N/A	N/A	Auction	www.bidz.com
BidEra	Both	People	Very	Yes	No	98	Registration	Both	bidera.com
Overstock	Both	People	Yes	Yes	No	99.5	Purchase	Auction	overstock.com
Yahoo Auctions	Both	People	Very	Yes	No	99	Purchase	Both	auctions.yahoo.com
Auctionfire	Both	People	Very	Yes	No	90	Purchase	Both	auctionfire.com
MSN Auctions	Buyers	Both	Not Very	Yes	No	80%	Purchase	Both	auctions.msn.com

Table 2: Comparative survey of popular auction sites. For a breakdown and analysis of results presented, see Section 7.2.1

core algorithm in this chapter uses techniques borrowed from the field of natural language processing (NLP), specifically *sentiment extraction*. Next we discuss relevant work in this area of NLP. Lastly, a brief examination of research on constructing and visualising social networks is presented.

7.3.1 Classifying Natural Language Comments

For many years research effort has been dedicated to the use of lexical techniques to classify free responses in areas ranging from online auctions [47] to movies [94] and for correcting students short-answer exams. [17]. Processing of freetext comments requires four main components, morphology (prefixes and suffixes), syntax checking, deciding on a semantic meaning and finally, taking some action based on that meaning. In this chapter we are especially interested in the classification of comments with *negative* sentiment. Yukari et al. [50] proposed a method to extract potentially unsatisfied customers by applying text mining to data from a customer satisfaction survey. This work introduced the idea of salient *satisfaction factors* as a mechanism for classifying negative comments. Later in this chapter a *feature-based* trust metric is introduced, which was inspired by work in [50]. Pang et al. attempt to classify freetext documents according to *sentiment* in [94], highlighting bad performance of machine learning algorithms such as naïve bayes and support vector machines in the task of sentiment classification. Pang et al. conclude that classification of free text according to sentiment is a challenging problem. Gamon et al address the same sentiment classification problem in [34], using data from car reviews. Gamon et al. also applied machine learning techniques to this task and found poor performance. They developed a tailor-made clustering algorithm which uses a small amount of domain knowledge in the form of a *stop-list* and *go-list* of features known to be salient in the domain. The *AuctionRules* algorithm described later in this chapter defines small negative and positive lists in a similar manner to that in [34].

Gamon et al. [34] introduce the concept of varying the level of granularity during the classification procedure, finding that "varying the level of granularity of analysis allows the discovery of new information". The *AuctionRules* algorithm can record finer grained information during classification, to

present extra information to users.

Hijikata et al. [47] focus on summarisation of freetext comments in online auctions, proposing a technique for analysing the underlying social networks in online auctions and using this information to present a summary of comments to users. In this work, the social network for their test data is defined by a crawler algorithm which takes all the feedback comments written by a buyer who wrote a comment for a target seller. As a result of this data crawling technique it appears that the data collected does not represent a natural social network, and if large portions of freetext data come from single individuals there is a potential for overfitting during computation. A web application is presented in [47] where users can see comment summaries represented as a set of bar charts for easy analysis.

7.3.2 Construction and Visualisation of Social Network Graphs

As people conduct larger proportions of their daily business on the web, the ability to analyse the resultant social networks is becoming more important. Technologies such as FOAF (Friend-Of-A-Friend) are becoming more popular in blogging communities, allowing users to directly specify their social networks. Visualisation tools such as *PieSpy* [84] (used later in this chapter) allow for dynamic display and fast assessment of complex graphs. Mitton [84] assesses a range of social networks mined from IRC channels. Golbeck uses social networks to compute trust scores and shows resulting visualisations derived from the *FilmTrust* movie ratings web site in [41].

The FOAF Project

Social networking meta-data, using the FOAF (Friend-of-a Friend) vocabulary, has emerged as a leading Semantic Web technology. A Semantic Web is a type of "weak AI". The concept of machine-understandable documents does not imply an artificial intelligence which allows machines to comprehend human-defined concepts, but it does indicate a machine's ability to solve a well-defined problem by performing well-defined operations on existing well-defined data [7].

```
<foaf:Person>
  <foaf:name>John O'Donovan</foaf:name>
  <foaf:gender>Male</foaf:gender>
  <foaf:title>Mr</foaf:title>
  <foaf:givenname>John</foaf:givenname>
  <foaf:family_name>O'Donovan</foaf:family_name>
  <foaf:mbox_sha1sum>cf2f4bd069302febd8d7c26d803f63fa7f20bd82</foaf:mbox_sha1sum>
  <foaf:homepage rdf:resource="http://www.johnod.eu"/>
  <foaf:weblog rdf:resource="http://www.johnod.eu/blog.jsp/"/>
 </foaf:Person>


 <foaf:Person>
  <foaf:name>Joe Bloggs</foaf:name>
  <foaf:knows>
    <foaf:Person>
     <foaf:name>John O'Donovan</foaf:name>
    </foaf:Person>
  </foaf:knows>
 </foaf:Person>
```

Figure 7.4: FOAF File Excerpt Showing RSA Encoded Email Address and foaf:knows Syntax.

The FOAF "Friend of a Friend" project* is a community driven effort to define an RDF vocabulary for expressing metadata about people, and their interests, relationships and activities. Founded by Dan Brickley and Libby Miller, FOAF is an open community-lead initiative which is tackling the wider Semantic Web goal of creating a machine processable web of data. [24] Users can add their metadata to the FOAF network by publishing a FOAF file on their web server. A FOAF file is an RDF containing personal information such as name, email etc. The social network is defined in FOAF using a link tag `foaf:knows`. Figure 7.4 shows an excerpt of such a file. To preserve privacy, email information in FOAF files is encoded using RSA encryption.

*http://www.foaf-project.org/

FOAF facilitates the creation of the Semantic Web equivalent of the archetypal personal homepage: My name is Leigh, this is a picture of me, I'm interested in XML, and here are some links to my friends. And just like the HTML version, FOAF documents can be linked together to form a web of data, with well-defined semantics.

7.4 Extracting Trust From Feedback Comments

To address the problem of unnaturally high trust ratings on eBay, we look to the freetext comments and apply a classification algorithm tailored for capturing subtle indications of negativity in those comments. The situation arises frequently where users are afraid to leave a negative comment for fear of retaliatory feedback comments which could damage their own reputation. [103]. In many of these cases, a positive feedback rating is made, but the commenter still voices some grievance in the freetext comment. This is the type of subtle but important information the *AuctionRules* algorithm attempts to extract.

7.4.1 The *AuctionRules* Algorithm

To achieve our goal of computing a broader range of trust values for eBay, we developed an technique to predict whether individual comments were positive or negative overall, or whether individual components of those comments had positive or negative sentiment attached. *AuctionRules* is our classification algorithm with the goal of correctly classifying online auction comments into positive or negative according to a threshold. *AuctionRules* capitalises on the restrictive nature of online markets: there are a limited number of salient factors that a user (buyer or seller) is concerned about. This is reflected in feedback comments. We define a set of seven core *feature sets* for which the algorithm will compute granular trust scores. The following sets have a coverage of 62% of the comments in our database. The algorithm can obtain semantic information from 62% of the comments at a fine grained level. It is

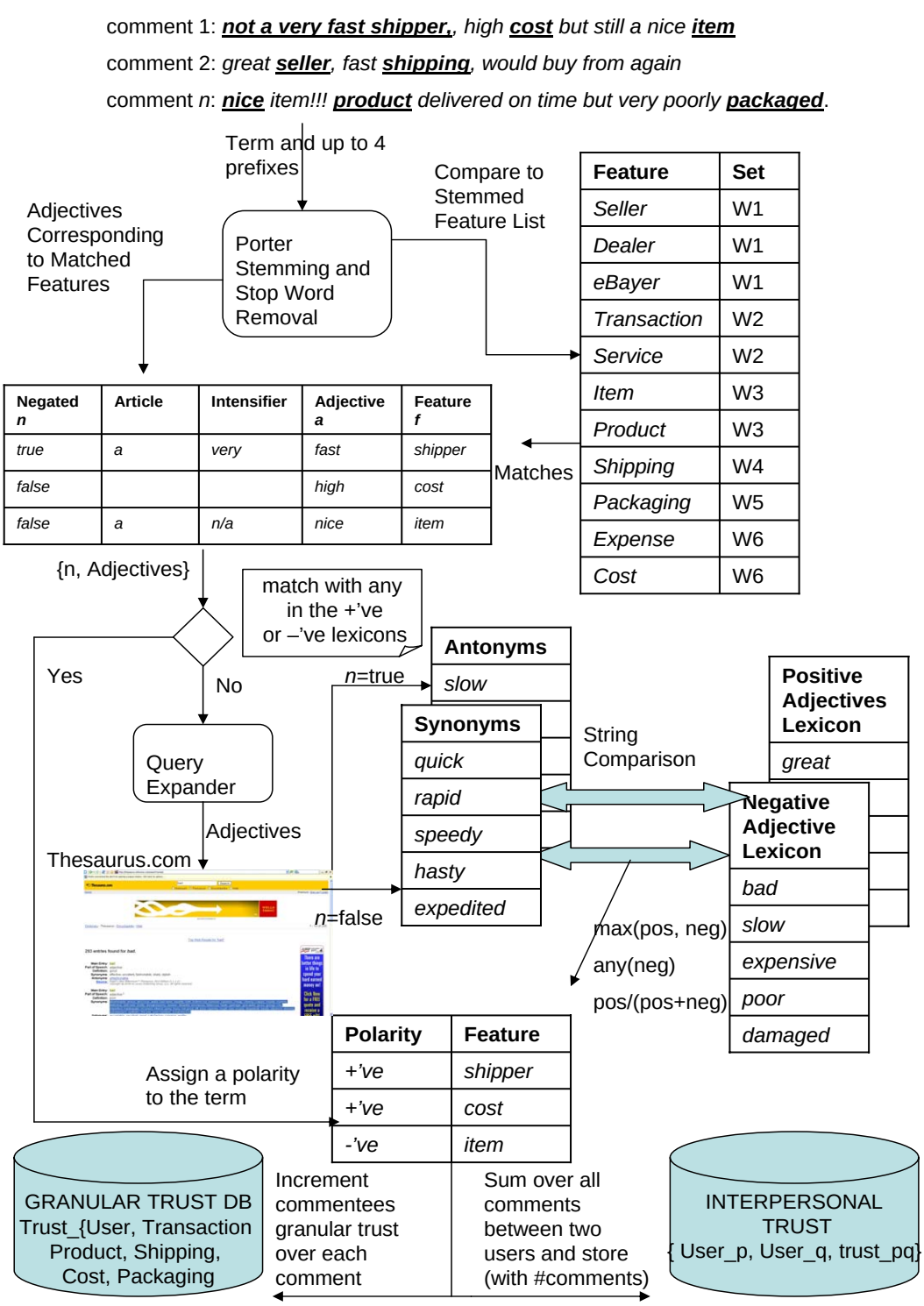

Figure 7.5: The *AuctionRules* Comment Classification Algorithm.

shown in our experimental analysis how we can maintain over 90% coverage using this algorithm. The terms in brackets are contents of each feature set.

- *Item* - The quality/condition of the product being bought or sold. (*item, product*)
- *Person* - The person the user makes the transaction with. (*buyer, seller, eBayer*, *dealer*)
- *Cost* - Cost of item, cost of shipping, hidden costs etc. (*expense, cost*)
- *Shipping* - Delivery of the item, security, time etc. (*delivery, shipping*)
- *Response* - Communication with the other party, emails, feedback comment responses. (*response, comment, email, communication*)
- *Packaging* - The packaging quality/condition of the item (*packaging*)
- *Payment* - how the payment will be made to the seller, or back to buyer for return (*payment*)
- *Transaction* - the overall transaction quality (*service, transaction, business*)

This technique enables us not only to compute a personal trust score between individual users, but also to provide more granular information on a potential transactor. For example: "User x is very trustworthy when it comes to payment, but shipping has been unsatisfactory in the past", This granular or *contextual* trust draws on the wealth of information in comments and can uncover hidden problems which the current trust system on eBay might overlook. For example if John in New York wants to order a crate of Italian wine from Lucia in Italy, it would be very beneficial for John to examine Lucia's history of shipping quality.

Figure 7.5 details *AuctionRules* working on a sample comment which had a positive rating on eBay. (All of the explanation in this section refers to Figure 7.5.) Each term in a comment and up to four preceding terms are passed into an implementation of the Porter stemming algorithm [97]. The standard porter stemmer uses many rules for removing suffixes. For example, all of the terms in c are conflated to the root term *"connect"*. c = *connect*, *connected*, *connecting*, *connection*, *connections*. This reduces the number of terms and therefore the complexity of the data. The stemmer algorithm was

modified to also stem characters not used by *AuctionRules*, such as "?, !, *, (,)" for example.

Data dimension is reduced by removal of stop-words. Google's stop-word list* was used as the removal key. A future addition to the algorithm would put an automatic spelling corrector at this point. There were 11 different spelling occurrences of the word 'packaging' for example. Each stemmed term is compared against the stemmed terms from the feature list. If a match is found the algorithm examines the preceding term types. This is shown graphically in Figure 7.5. The algorithm recognises five term types:

- *nouns* the words contained in the feature sets.
- *adjectives* (e.g: 'nice', 'good') from a web list of 150
- *intensifiers* (e.g 'very', 'more') list of 40.
- *articles* 'a' or 'the'
- *negators* (e.g. not, 'anything but') from a manually generated list (currently 17).

The table on the left in Figure 7.5 shows the order in which these terms are analysed. From the five terms, two can provide semantic meaning about the feature: adjectives and negators. If an adjective is found without a negator, it is compared to an arrays 20 positive and an array of 20 negative adjectives. If a negator is found as the second or third preceding term, the process works with positive and negative arrays switched. If a match is found, the polarity for that feature is recorded.

If no match is found, *AuctionRules* uses query expansion, by calling an interface to an online thesaurus which returns an array of 20 synonyms. If a negator is present, the interface returns an array of 20 antonyms, and these lists are compared in a similar manner to our short lexicon of positive and negative adjectives. The matching results are recorded in three ways: a) $max(pos, neg)$ b) $any(neg)$ and c) $\frac{neg}{pos+neg}$. In the case of (c) the polarity is recorded according to a configurable threshold α. Two separate trust databases are maintained: *granular* or *contextual* trust which is the trust computed for each feature for a user over all of the comments analysed. Equation 7.1 shows contextual trust t as a multi valued set of trust scores

*http://www.ranks.nl/tools/stopwords.html

associated with each feature. Here, f denotes a particular feature and t_fn is the associated trust score. The second trust database is *interpersonal* trust which is the average trust value recorded for all features on every comment made between two users.

Of course not every comment will contain words from the feature lists above. In cases where no features are found, the algorithm performs a straightforward count of positive and negative terms using the query expansion module where necessary. In this manner, coverage is maintained at over 90%, and many of the unclassifiable comments are foreign language or spelling mistakes. For example, from all of the (10k) eBay comments stored by our crawler, 68 different misspellings of the word "packaging" were found. It would be interesting as future work to analyse the effects of automatic spelling correction on our results.

One drawback with the current implementation is that negative sentiment which is represented in complicated grammatical expressions is often misinterpreted by the algorithm. For example, using the current implementation the sentence `the item was anything but good, shipping was ok` gets misinterpreted to `good shipping`. To overcome this more complex natural language processing rules need to be developed to account for more complex statements. However, the vast majority of comments crawled from eBay are expressed in simple grammar, so instances of the above problem are relatively few. There are several available "black box" NLP classifiers, for example, the Stanford university NLP tools*. Future work on this algorithm should include a comparison between *AuctionRules* and these NLP classifiers.

7.4.2 Propagating Trust

Interpersonal trust is given by Equation 7.2. This is a personalised trust score between two users. If users have been directly involved in a transaction, this is simply the classifiers value of their transactions (a, b, or c above). Equation 7.2 shows how these trust scores can be computed along a path connecting the two users. In Equation 7.2 $\oplus$ represents some combination of the scores

*http://www-nlp.stanford.edu/software/index.shtml

```
<?xml version='1.0' encoding='utf-8'?>
<graph>

 <node name="011stephen">

 <node name="111laz111">
      <edge name = "venturefox" value = "904.0" id = "43675"  />
      <edge name = "venturefox" value = "999.0" id = "43613"  />
      <edge name = "venturefox" value = "932.0" id = "43551"  />
      <edge name = "venturefox" value = "948.0" id = "43489"  />
      <edge name = "venturefox" value = "947.0" id = "43427"  />
      <edge name = "venturefox" value = "946.0" id = "43365"  />
 </node>

 <node name="11thmiss">
      <edge name = "ptolemyxvi" value = "949.0" id = "42678"  />
 </node>

</graph>
```

Figure 7.6: XML Representation of Trust-Graph Information (Input to the Graphing Tool).

at each node in the path between the target and source users. Currently this can be any of the following four techniques below. The problem of combining values in the trust graph is a research in itself. This will be dealt with in more detail in a future work.

- *weightedDistance* - The average trust score over all the edges in the shortest path, discounted by the distance from the source.
- *meanPath* - The average trust score over all the edges in the shortest path between the source node and target node.
- *twoPathMean* - The average of the *meanPath* of the shortest path in the graph and the *meanPath* of the second shortest path.

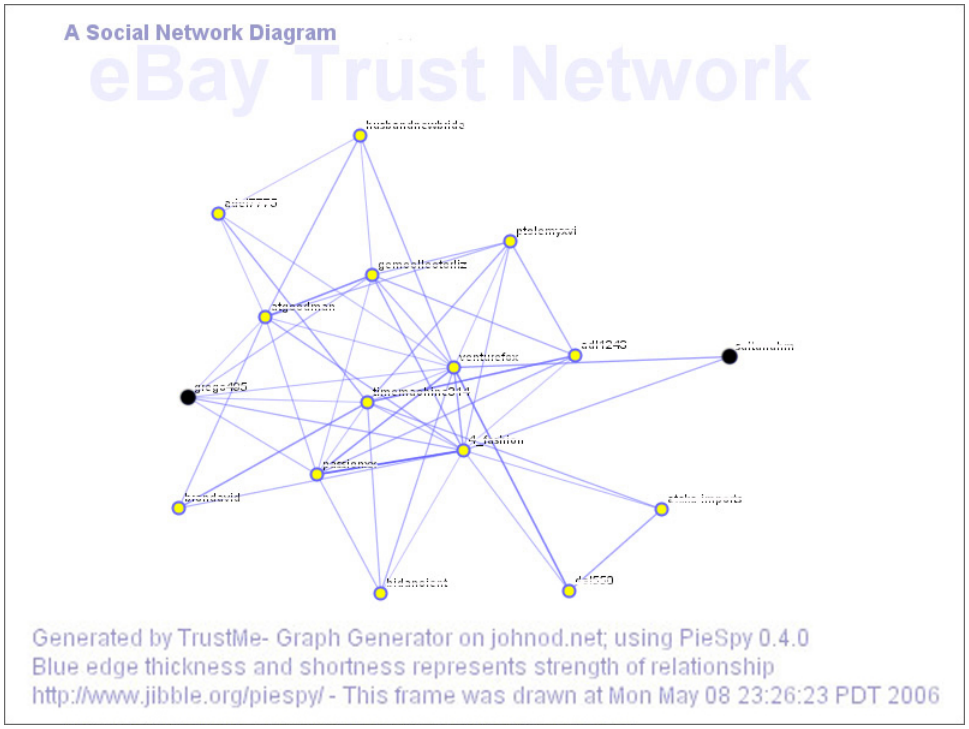

Figure 7.7: Visualising the Trust-Network Generated by *AuctionRules*. (Usernames have been Obfuscated at the Request of eBay Inc.)

- *SHMPath* - The simple harmonic mean of the trust scores over all edges in the shortest path.

$$t_{granular} \varepsilon \{t_{f1}, t_{f2}, ...t_{fn}\} \tag{7.1}$$

$$t_{i,n} = t_{i,j} \oplus t_{j,k} \oplus ... \oplus t_{m,n} \tag{7.2}$$

7.4.3 Visualising Trust Information

Before introducing our presentation strategy, we define two important metrics, firstly a scalar trust *value* which is represents trust either between two users, or for one user on a particular feature. Secondly, we define trust *strength*. This is based on the number of transactions between two users

in the case of interpersonal trust, or as the number of comments used to calculate the feature trust score in the case of contextual trust.

To generate on-the-fly visualisations of the resultant trust network, we modified a version of *PieSpy* [84], an open source graphing tool by Mutton for visualising social networks. Our interpersonal trust database is a set of triples of the form: $e_{(i,j)} = (user_i, user_j, t_{(i,j)})$, where $t_{(i,j)}$ comes from Equation 7.2 and is the trust value generated by *AuctionRules* for those users i and j. This was translated into XML and fed to the graphing tool. Modifications were made to the existing code to highlight target and source users in the graph. Figure 7.7 shows the visualisation output from a demonstration set of users in our database. (this graph has much higher than average interconnectedness.) The graphing tool caters for trust *value* and trust *strength* by representing value as the thickness of an edge and strength of the trust-relationship as the length of a line. In the graph, shorter lines mean a higher number of comments were used in the classification. The demonstration system in Figure 7.8 shows the trust graph popup when a mouseOver occurs on a username. To visualise contextual trust, a popup table with feature names, associated trust scores and strengths is used. This is shown as the smaller popup in Figure 7.8.

Unlike the *PeerChooser* visualiser discussed in the previous chapter, *PieSpy* was not specifically developed for this visualisation task. *PieSpy* is open-source software and has a range of other functionality which is not used in this work. *PieSpy* can track conversations over IRC channels and plot meaningful visualisations of the discovered social network. This section provides some detail on the forces and interactions within the *PieSpy* layout algorithm, which was modified slightly for our visualisations. *PieSpy* operates using a spring embedder which is based those of Fruchterman [31]. This version of the spring embedder is effective and widely used. It requires a minimal set of parameter values that can be adjusted to achieve good automatic layouts. In this model, the repulsive force acting on a pair of nodes is $k2/d$ and the attractive force between two nodes caused by an edge is $d2/k$, where d is the distance between two nodes and k is a constant. With this force model, it is worth noting that an edge connecting the only pair of nodes in a graph will have a natural length of k. The graph drawing process is started by

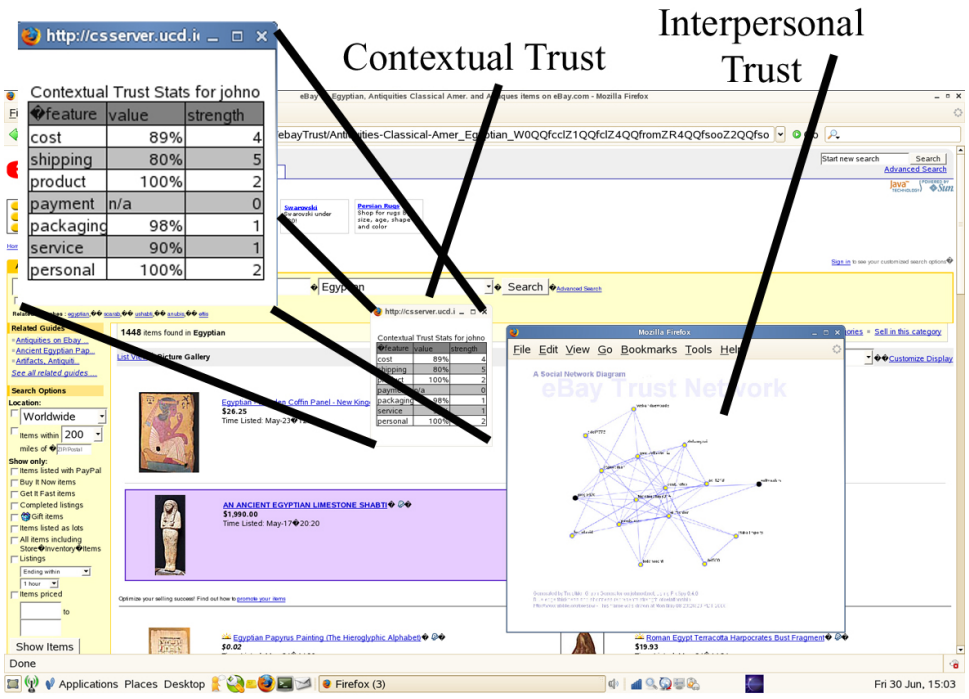

Figure 7.8: Integration of the Trust System into an Online Auction. (Usernames have been Obfuscated at the Request of eBay Inc.)

allocating each node to a random location on a two-dimensional plane. The iterative calculation of the forces begins and nodes are moved accordingly at the end of each iteration step. This results in a layout where connected nodes are close together, yet no pair of nodes are too close to each other due to the repulsive forces acting between them. The graph eventually settles to a static position which represents the minimum energy state of the system. The spring model can cause the layout to expand rapidly, as there is nothing to counter the repulsive forces acting between each maximally connected component. Limiting the distance over which repulsive forces may act easily solves this problem. The force model is modified so that a pair of nodes with separation greater than m does not exert a repulsive force.

7.5 Evaluation

We examine four factors in our evaluation of *AuctionRules*, including the *accuracy* of the *AuctionRules* classifier with respect to other techniques from machine learning. We also examine accuracy from a Mean Absolute Error perspective and by creating confusion matrices to examine performance with respect to false negative and positive classifications. As the system uses a very small amount of domain knowledge, and a limited number of features, we must examine the *coverage* of *AuctionRules* over our comments database. Finally we make a comparison between the *scale* of trust achieved by *AuctionRules* against the current eBay scale.

7.5.1 Setup

Figure 7.9 explains the environment in which we test our algorithm. Data is crawled from the online auction site according to the crawler algorithm below. Importantly, unlike it's machine learning counterparts, *AuctionRules* requires *no* knowledge of the comment set it has to classify. The feature lists used by the algorithm are generic and should work on any set of online auction comments. Arguments have been made that *AuctionRules* relies heavily on domain knowledge and that this is very restrictive. The *AuctionRules* algorithm uses little domain knowledge in the form of lexicons of positive and negative words, intensifiers etc. *AuctionRules* will work on any system where users buy and sell from each other and leave textual feedback comments using no more domain knowledge than is used in the eBay system. Relative to the size of the application domain, the knowledge required by the algorithm is very minimal.

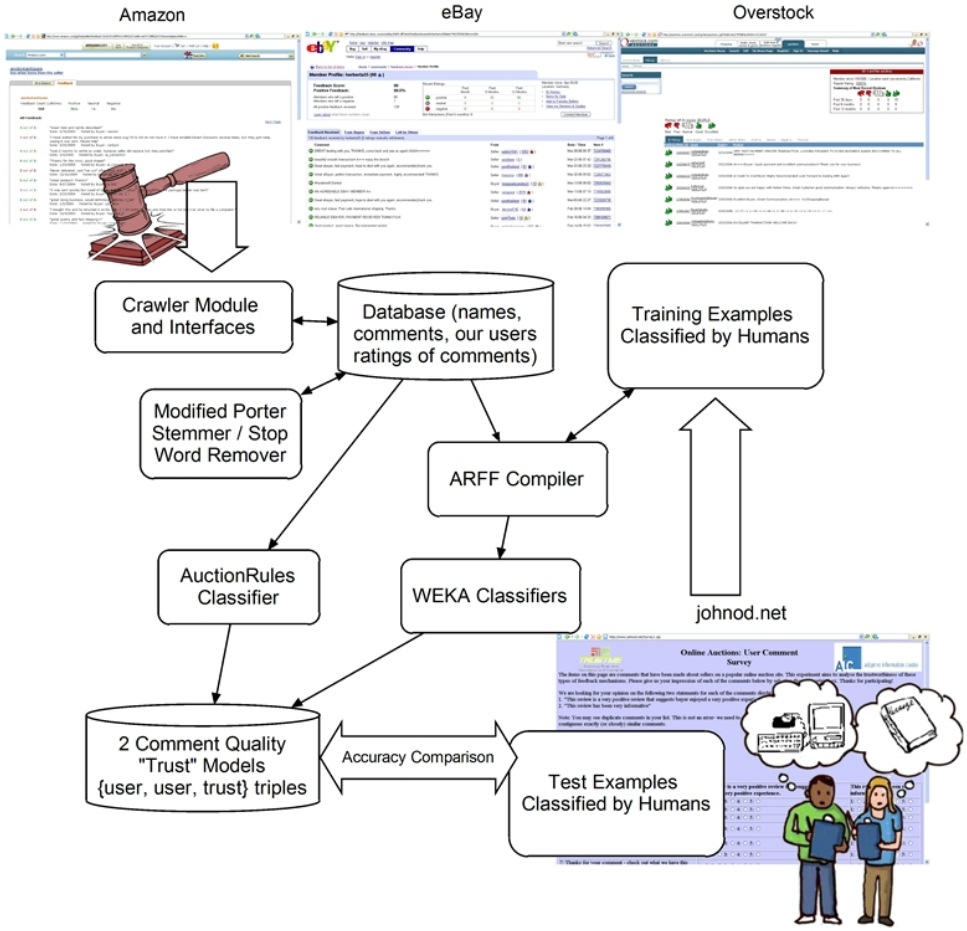

Figure 7.9: Graphical Overview of the Trust-Modelling Process. (Current Implementation Only Uses eBay as a Source for Ratings.)

Algorithm 7.5.1: CRAWL($String\ url\_list,\ int\ maxbound$)

while $n < maxbound$
for $i \leftarrow 1$ **to** $\#Items\_on\_page$
  $followSellerLink()$;
  for $j \leftarrow 1$ **to** $\#Comments\_on\_page$
    $db.add(bId, sId, comment, sTrust, bTrust)$;
  $n \leftarrow n + 1$;
return ;

10,000 user comments were collected from the auction site using the

crawler. As a social network visualisation was planned, data was collected from within a domain with highly interconnected and contained repeat-purchase users. After a manual analysis of a range of sub-domains of the auction site, Egyptian antiques was chosen as the data domain as it appeared to meet the prerequisites to a reasonable degree. Although a large number of comments were collected, only 1000 were used in our experimental analysis.

User-Provided Training Data

In order to test any comment classification algorithm a benchmark set was required. 1000 comments were compiled into an online survey* and rated by real users. In this survey, users were asked to rate the positiveness of each comment on a Likert scale of 1 to 5. 10 comments were presented to a user in each session. Each comment was made by different buyers about one seller. Users were required to answer the following:

- How positive is the comment (Average rating: 3.8442)
- How informative is the comment (Average rating: 3.1377)
- Would you buy from this seller (Average rating: 4.0819)

Currently only results from the first question are used to develop and test *AuctionRules*. For future experiments we may incorporate results from the other questions. Permission was sought from eBay inc. to use the information from the eBay web site in our experiments.

7.5.2 Comparing *AuctionRules* With Machine Learning Techniques

To examine classification accuracy of *AuctionRules*, it was tested against 7 popular algorithms. We chose three rule-based learners, *Zero-r*, *One-r*, *Decision Table*, one tree learner *C4.5 rules*, two Bayes learners, *Naive Bayes* and *BayesNet* and a lazy learning algorithm *K-Star*. Table 7.5.2 provides a brief explanation of each algorithm used.

*www.johnod.net/Surveyone.jsp

AlgorithmName	*AlgorithmDescription*
Zero-*r*	a simple majority class predictor.
One-*r*	produces very simple rules based on a single attribute
Naive Bayes	implements the probabilistic Nave Bayesian classifier
Decision Table	uses wrapper method, finds a good subset of attributes for inclusion
C4.5 Rules	120Set of rules that produces decision trees.
BayesNet	finds an appropriate Bayesian network given a data set D over U.
K-Star	an instance-based classifier

Table 7.1: Brief explanation of classification algorithms used.

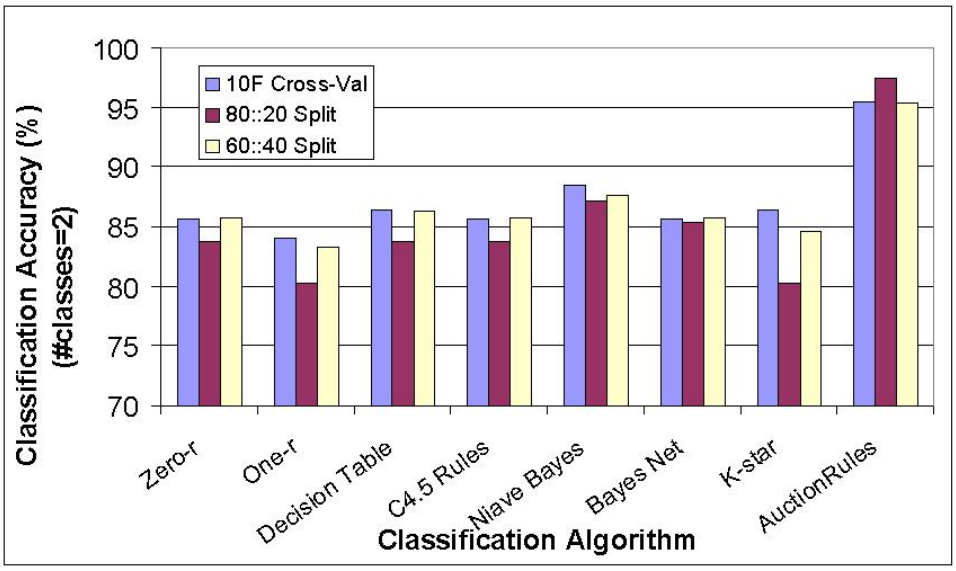

Figure 7.10: Classification Accuracy [Classification Distribution from User Evaluations: 36% positive, 63% negative, using a threshold of 4 or higher for a positive comment.]

Figure 7.10 shows results of this experiment. For each algorithm we performed three runs. a 60:40 train-test split, an 80:20 split, and a 10-fold cross validation of the training set, which randomly selects a training set from the data over 10 runs of the classifier and averages the result. In the experiment each algorithm made a prediction for every value in the test set, and this prediction was compared against the training set. *AuctionRules* beat all of the other classifiers in *every* test we performed, achieving over 90% accuracy in all of the evaluations, 97.5% in the 80:20 test, beating the worst performer *K-Star* by 17.5%, (relative 21.2%) and it's closest competitor *Naive Bayes* by 10.5%, giving a relative accuracy increase of 12.7%.

In addition to numerical accuracy, we examined where the high accuracy results were coming from more closely by assessing the confusion matrix output by the algorithms. This was necessary since prediction of false negatives would have an adverse effect on the resulting trust graph. This phenomenon has been discussed by Massa in [71] with respect to the *Moleskiing* application, and Golbeck in [41] with respect to the TrustMail application. Table 7.5.2 shows *AuctionRules* outperforming all of the other algorithms by predicting no false negatives. This is taken as a good result since in a propagation computation negative values should contain more weight because of their infrequency. When a value is presented to a user directly however, false positives are more damaging for an auction environment. *AuctionRules*also performs very well for false positives with a rate of 4.5%, half that of the closest competitor *One-r*. All of the algorithms displayed similar trend to the ones in Table 7.5.2, which shows results of the 80:20 classification experiment which had a test set of 234 comments. It was found during accuracy evaluations that there was a strong correlation between the number of feature terms recognised and the final accuracy of the classification. For our coverage experiments, we addressed the number of *hits* found with respect to coverage. This is detailed in the following section.

7.5.3 Coverage and Distribution Experiments

To assess the coverage of the *AuctionRules* feature-based trust calculator we examined the number of feature term hits that occur during classifica-

	AuctionRules		*NaiveBayes*		*Decision Table*		*One-r*	
	+'ve	**-'ve**	+'ve	**-'ve**	+'ve	**-'ve**	+'ve	**-'ve**
+'ve	91.4	*0*	84.1	1.2	84.6	1.2	77.3	8.1
-'ve	4.7	4.7	11.1	2.9	12.3	1.7	8.5	5.9

Table 7.2: Confusion matrices showing percentage true and false negatives and positives for four of the algorithms tested. [All of the other algorithms had similar results to the ones displayed.]

tion. Coverage was tested in two modes. Firstly, the standard feature-based mode. In this case, 62% of the 1000 comments tested contained at least one hit from the feature list. This provides us with semantic knowledge about a large number of comments. However there is a sharp reduction in coverage when we look for comments with more than one hit. To increase coverage *AuctionRules* uses simple term counting to supplement its feature-based approach in cases where it fails to find any terms. When counting is applied the coverage is greatly increased with over 90% of the comments getting at least one hit. After manual examination, it is clear that majority of the other 8% can be attributed to misspellings, foreign language and other noise.

A distribution analysis was performed between the *AuctionRules* generated trust values and the current and found that the new trust values do capture some unseen negativity from feedback comments. This is manifested in a slightly more scaled distribution relative to the current eBay values. From 600 comments that were analysed in the distribution experiment, 90 comments which had been rated as 93 to 100 percent positive were taken out of that bracket by *AuctionRules*. This may seem like a small amount, but compared with the current system which has less than 1% negative ratings, *AuctionRules* produced 16%.

7.5.4 On the Generalisation Of *AuctionRules*

During the course of this work we have argued that the small amount of domain knowledge required by the AuctionRules algorithm is not specifically tailored to eBay and in fact can work on any online auction where users leave feedback comments. An interesting future avenue of research in this

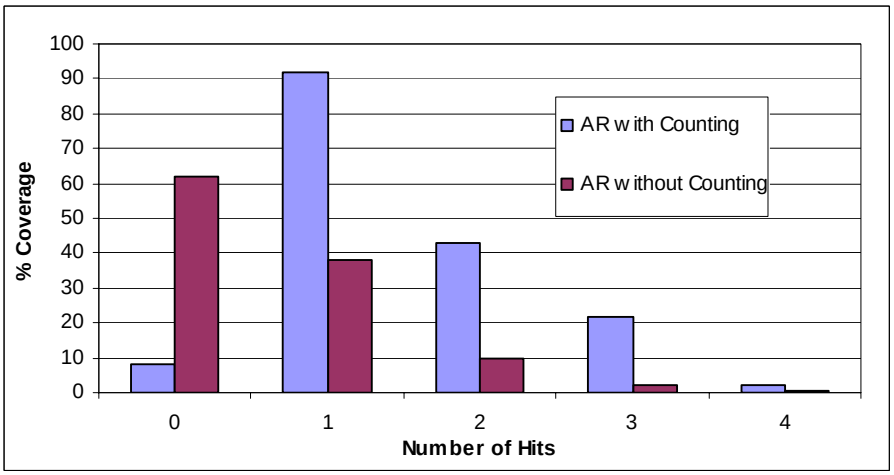

Figure 7.11: Comparison of the Coverage and Hit Ratio in *AuctionRules*

area might be to elicit similar data from another auction site, such as Yahoo auctions and run the same tests as with the eBay comment data. Furthermore, it is possible that user details may overlap between the two sites. For example, usernames or their business names. If this were to occur it might even be possible to propagate generated trust values across different applications on the Social Web. However, for this to really work efficiently, some standardisation should occur. For example, users should carry a unique id across different Social Web applications. This is a difficult issue for designers of applications, because users become concerned about privacy issues [114].

7.6 Summary

The main contribution of this chapter is an algorithm for extracting *personalised* and *contextual* trust from the wealth of freetext comments on online

auction sites. The algorithm operates on the assumption that online auction transactions can be categorised into a relatively small set of features. *AuctionRules* can extract context-specific and personal trust both retroactively and on the fly as new comments are added. There are a range of uses for the extracted trust scores. In this chapter we have shown one such use in a pop-up visualisation of the resulting trust network for a demonstration subset of the eBay marketplace. This visualisation also shows per-feature trust in a pop-up window for a buyer or seller. In our evaluations we show that even using a small lexicon of key features, coverage is still maintained above 90%. We show that the *AuctionRules* classifier beats seven popular learning algorithms at this classification task by up to 21% in accuracy tests using very minimal domain knowledge. *AuctionRules* has a favourable false negative rate for the classifier of 0% compared to up to 6% from benchmark algorithms and a false positive rate of 4.7% compared with 8.5% to 12.3% for the benchmark algorithms.

The following chapter presents a summary and overview of the research in this thesis, followed by a detailed discussion of the goals and achievements of the work, and a list of limitations associated with each approach. This is followed by some concluding remarks.

Chapter 8

Discussion, Future Work and Conclusion

8.1 Introduction

The Social Web is a relatively new phenomenon, derived from and a subset of the traditional html-based and relatively static Web. The advent of the Social Web has brought with it a major shift in the flow of information on the Web. In place of traditional, webmaster-authored content, new applications have evolved which provide a framework for user-generated content (UGC). This shift in control of information authorship brings with it the problems of authoritativeness, author reputation and trust analysis.

Traditionally, information on Web pages has been authored by site owners and the reputation metrics for these sites were (and indeed still are) computed by algorithms such as PageRank [93]. PageRank was designed to calculate the importance of a static Web page by analysis of the incoming links to a page, so in a sense, the core structure of the Web defined the importance of its pages. Nowadays, the Social Web has hundreds of applications with millions of user accounts and each user contributing multiple fragments of information [40], potentially across many different Social Web applications. The network of links that existed in the the traditional web still exists today, and PageRank-like algorithms can still successfully operate by analysis of link topology, but the new participation platform of the Social Web has

brought about an entirely new network for us to consider: a network of *users*. Since it is the users and not webmasters that provide content, the traditional approach for evaluating the "goodness" of information on the Web may not be as successful on the Social Web, and should be broadened to cover new forms of "social content and context". Trust and reputation can play an important role in this.

The main objective of this thesis has been an examination of novel ways to tackle the problem of trust and reputation of the users within Social Web applications. By harnessing new and historical information from these applications, it becomes possible to make inferences about the trustworthiness of the people who use them. Learning about the trustworthiness of individual users on the Social Web is important because the percentage of all human interactions conducted via Social Web applications is increasing at a rapid rate.

8.1.1 Trust and Reputation on the Social Web

Throughout this work a distinction has been drawn between trust and reputation. For example, a user on the Social Web may have a good overall reputation, but may not be trusted by certain individuals for whatever *subjective* reasons they choose. In general, users in Social Web applications are trusted *as a result of* their good reputation, and conversely distrusted because of a bad reputation. Take online auctions as an example: a crucial question is whether or not one user decides to do business with another. This decision is a function of reputation, where reputation is an empirical value that can be calculated over a users transaction history. It is also a function of "trust", which is a subjective notion, based on reputation. Trust, to quote Marsh [69] is a "subjective user-centered notion", which must be above a certain subjective threshold (lets say "action-threshold") for a transaction to occur. Chapter one provided discussion and analysis of this concept from a Social Web perspective, the "Nintendo" experiment showed that Josang's [55] assertions that trust varies greatly with context hold true for trust in Social Web applications. The algorithms presented in this thesis focus primarily on trust in a computational sense, that is, on computable values that

can be mined from new and historical information within Social Web applications. The following section outlines the initial objectives and the overall contributions of this thesis.

8.2 Objectives and Contributions

As outlined in Chapter 1, the core objectives of this thesis are to consider the importance of trust in the context of the Social Web, to develop novel ways to harness trust information, compute new models of trust and to present trust information in a manner that can benefit users of the Social Web. The following list provides a more detailed look at the initial objectives which were defined at the outset of this work:

1. Assessment of the importance of trust within Social Web applications.
2. Identification of domains on the Social Web in which trust could be modelled from historical/new data.
3. Construction of algorithms that benefit Social Web users by integration of such trust information into a variety of computational processes.
4. Development of new ways to visualise trust and explain its concept to the end-user.

To meet the above objectives, two suitable Social Web application areas were chosen and a range of algorithms were developed to model, integrate and visualise trust information in various ways. The list below outlines the general contributions of this thesis to research in trust modelling on the Social Web.

1. Technique and application for building and using trust models in recommender systems
2. Technique and application for building and visualising trust relationships based on online auction feedback comments.
3. Algorithms for computing recommendation ranges based on error history in recommender systems.

4. Identification of reinforcement problem in trust based systems and analysis of attack strategies for trust based recommender systems.

5. Technique and application for visualisation of and interaction with trust and similarity values during the recommendation process.

6. Development of a visual *explanation interface* for recommender systems which are based on collaborative filtering.

The main concepts behind this work can be aptly described from a graph-theoretic perspective: the initial objective of this work was to examine the topology of the network that connects people on the Social Web, and to glean information about the nodes in this graph and the edges connecting them. Considering a node on this graph as a user, and an edge that links between two nodes as *any* piece of information which connects two users, we can construct a topology of the interactions that occur within the Social Web and make decisions about the trustworthiness of individual users with respect to specific tasks by taking trust as some function of the edge value between two nodes. Once this graph has been discovered, we can harness the power of rich collaborative information, communal or "social" wisdom, to enhance Social Web applications in a variety of ways. Examples of the connecting edges on this type of graph include item ratings, feedback comments, bookmarks, tags, blog comments, product purchases etc.

Even in cases where direct interactions between users have not occurred, it is still possible to use these "links"to artificially generate interactions between users. For example in our eBay experiments, the individual links that were used were taken from feedback comments between two users. In this case the link is a product of an existing direct interaction between two users. In our ACF experiments however, we do not have such direct links and need to artificially generate the interaction in the form of a recommendation.

Following is a brief discussion of each of the contributions made in this research, and analysis of the benefits and limitations of each contribution. Many of the limitations discovered apply across all of the techniques and applications which were developed. Accordingly, a discussion of the general limitations to trust based systems on the Social Web is presented before addressing issues specific to each contribution individually.

8.3 General Limitations

In traditional marketplaces and social groups, reputation was built over time and propagated by word of mouth. Modern day internet marketplaces and social networking applications attempt to mirror this mechanism because people naturally tend to be more comfortable with traditional methods [103] [104]. Examples of this mirroring include eBay's rating system, Flickr's photo tagging and Facebook's "friend" policies. Despite our technological advancements, the Social Web is still very much in its infancy when it comes to accurately modelling human trust mechanisms. This section details the major challenges for modelling trust on the Social Web that were encountered during this research. These limitations are listed below:

1. Data Sparsity
2. Transitivity
3. Portability/Contextual Issues
4. Reinforcement
5. Commutability
6. Combinatorial Problems

8.3.1 Data Sparsity

There is a lack of data available as the basis for any trust model due to the fact that most users/agents do not interact directly with other users/agents, and the fact that these interactions are generally very limited. However, the recent proliferation of social networking applications (spearheaded by Facebook) produces a rich data resource. With more than 67 million active and highly connected users*, the Facebook API will be a powerful data resource for future social networking research.

*http://www.facebook.com/press/info.php?statistics

However, in many existing Social Web applications data availability is not as high. In some online auction applications, for example eBay, information upon which trust decisions are made online is often limited to no more than a username, making it very difficult for a user to make informed trust decisions. Largely due to the economic implications of this, the problem is gaining increased attention from researchers around the world, but unless some serious standardisation occurs across major web applications, the problem of sparse information will continue. One ideal solution might be to enable a user to carry their profile around with them as they use different Social Web applications. Introducing a cross-application profile sharing mechanism would greatly improve the quality of trust decisions made on the Social Web, while at the same time would introduce a host of security and privacy related issues [103].

8.3.2 Transitivity

Given the sparsity issue and the lack of direct interactions between the majority of Social Web users, one potentially rich source of information may be found by the propagation of trust information via user connections in order to evaluate *indirect* trust which exists between potential transactors. Various recent works have dealt with the issue of propagation of trust values around a network. These research efforts cover areas from social science such as Coleman in [29], multi-agent systems as in Ramchurn et al in [100] to research in social networks like that of Golbeck in [38] [37] [39] [40], Massa [71] [70] [5] and [4], Ziegler in [123] and [121] and Josang in [55] and [57]. Throughout these works there are many different models and approaches mentioned, all of which discuss in some way the transitive properties of trust (or lack thereof). This thesis described a method for propagation of trust through the process of recommendation. The core limitation with this technique, and with all of the above is that as trust information is passed along a chain of users, the reliability of the information is significantly weakened. For example, if Mary trusts Bob and Bob trusts Jim, then we can assume that by the transitive property, Mary trusts Jim, but this information is not as reliable as the information in the direct link: Mary trusts Bob. A challenging problem

in systems which use transitive trust information is to identify the number of "hops" that can be made along a chain of people while still maintaining meaningful and useful trust information.

8.3.3 Trust in Context

Trust is tightly related to its context. In his 1994 work [69] Stephen Marsh defines "context-specific interpersonal trust" as the trust that exists between two people with respect to a specific situation. Deuctch talks about the importance of context with respect to trust in his 1962 work [22] but it is referred to as an "ambiguous path" which a person is faced to make a trusting decision on. In a similar manner, Luhmann [68] states that trust is given in a situation in which the circumstances are uncertain and it is clear that one can either act or not act, but that acting involves risk. Trust in Social Web applications is no different, for example, sharing personal information on Facebook or paying for a product on eBay before you have seen it. Trust decisions made by users in Social Web applications are determined by a wide range of influences. For example, if I were to purchase a laptop on eBay, I would be more concerned about a sellers reputation for packaging than for speed of delivery. However, if I were purchasing concert tickets from the same user, I would be more concerned about timely delivery than about the packaging. In these two scenarios, my trust for one particular seller might vary greatly based on his reputation for packaging and for speedy delivery of goods.

Leading works which attempt to define, formalise or just explore the concepts of human trust tend to agree that trust varies greatly with context. This is highlighted in very simple terms by Josang in his 2007 work [55]. Josang's example shows an image of a thin rope and asks the question "would you trust this rope?", in the following situations: firstly, during a fire drill and secondly during a real fire. Of course, during a drill people would be concerned that the thin rope may break and therefore say they would not trust the rope. However, during a real fire its either get burned or use the rope, in which case the rope becomes highly trusted. The "reputation" of the rope in this case will stay the same. (i.e: its history of breaking in similar

situations), but in the face of adverse conditions the person becomes forced to trust the rope despite its bad reputation.

8.3.4 Portability

The Social Web has great potential for the study of trust between its constituent users. However one of the biggest challenges to the development of reliable and robust trust models is the fact that the Social Web, despite its highly interconnected link and social topologies, consists largely of a set of independent, non-standard applications. As a result, we have a rich diversity of tools and services, all of which operate in different ways with respect to trust information. For example, on eBay, trust is defined by user feedback comments after transactions, on Facebook, Flickr and LinkedIn trust is explicitly stated by friendship or association statements, and on Wikipidia trust is defined by general acceptance of authored content. These diverse trust mechanisms work to some degree within their limited domains, but as users move about the Social Web they must build up their reputations for each new application they use. If it was to become possible to interrelate trust across multiple Social Web applications then we could (a:) rely much more on our trust models since they are compiled using a much larger window of information, and (b:) avoid the problem of "starting afresh" for each new application on the Social Web. However, this is not currently the case, and there are two salient challenges that arise when we attempt to port trust information across platforms:

1. **Trust is highly contextual.** The previous section has presented discussion on how trust can vary greatly with context. This applies to the portability argument. For example, just because I would trust seller "bidAincent03" to sell me Egyptian antiques on eBay, definitely does not mean I would trust him as a friend on Facebook. Future research in this area may benefit from a look at enabling *reputation* metrics to be passed between applications, and maintaining local computation of trust, at an application-specific or better yet, context-specific level. These local trust calculations can easily be based in some way on an imported reputation model. (Note: "reputation" in this case means

historical, non-subjective, statistical data.) In this sense, Facebook could be aware that "bidAincent" appears to be a reliable user, and this information could be used to influence a trust decision which I might make in the future.

2. **Making personal information portable is an infringement on *privacy.*** In general there exists a sweetspot between privacy and the performance of intelligent personalisation systems, as highlighted by Smyth in his 2005 work [114]. People are willing to give a certain amount of personal information to a system in return for better performance of that system. The concept of an all-encompassing, global trust or reputation profile that spans the Social Web is a terrifying prospect for those concerned about privacy, i.e: everyone. Nobody wants a "big-brother" scenario where a users every action is watched intensely. A clear trade-off occurs between maximising the benefits that can be gained from better trust and reputation models, and the privacy concerns of users in the Social Web. Systems such as eBay maintain privacy through anonymity, in that a user's real information is hidden behind a pseudonym, but in some cases users may not want the reputation associated with their eBay pseudonym ported to another site. For example, an eBay user with a trust score of 45% probably would not want this information displayed on their professional LinkedIn profile. Maintaining the sweetspot between personalisation and privacy is and will continue to be an ongoing effort. The core challenge to the portability of information upon which we can make trust decisions is intrinsically linked with issues of benchmarking and standardisation. If Social Web users can be convinced that sharing certain data through a well established, reliable reputation-sharing protocol can be beneficial, it would be a great step towards development of robust and reliable trust models.

8.3.5 Commutability

Marsh's 1994 findings on the commutableness of trust are still applicable to the Social Web: trust is *not* commutable. Knowing that Bob trusts Mary

with respect to a certain context, or in general, does not necessarily have a bearing on the trust Mary has for Bob with respect to that situation, or any other, as expressed by the following simple formula:

$$trust_i(j) \neq trust_j(i) \tag{8.1}$$

The lack of commutableness of trust values increases the sparsity of trust graphs on the Social Web, and less importantly but still relevant, it will increase the computational processing time for trust calculations on large-scale graphs, as the link between two users will have to be calculated twice. It would be convenient from a computational standpoint to assume that trust had the property of commutableness since we could check trust models calculated over diverse data by ensuring the property remained constant between two given users. However, this is not the case for human trust or most scenarios on the Social Web because the actions of two users are generally independent of each other, so this technique cannot be used to propagate trust information around a network.

This concludes the discussion on the general limitations that are communal to all of the trust-based algorithms which were developed and/or used in this research. Following is a description of each of the contributions that were developed from the initial set of objectives, and a brief discussion of the limitations that are specific to each one.

8.4 Contribution 1: Trust in Recommender Systems

The first main contribution of this thesis, a technique for modelling trust in a collaborative recommender system, was introduced in chapter 3. Detail of design, implementation and evaluation was presented for each of our trust models. Chapter 3 presented the argument that profile similarity on its own may not be sufficient to compute recommendation partners, that other factors might also have an important role to play. Specifically, the notion of trust has been applied in reference to the (computable) degree to which one might trust a specific profile when it comes to making a specific rating

prediction on either a specific item, or in general.

Two different trust models have been developed, one that operates at the level of the profile and one at the level of the items within a profile. In both of these models trust is estimated by monitoring the accuracy of a profile at making predictions over an extended period of time. Trust then is the percentage of correct predictions that a profile has made in general (profile-level trust) or with respect to a particular item (item-level trust). Several ways in which these different types of trust values might be incorporated into a standard collaborative filtering algorithm were presented. Each trust-based prediction technique is evaluated against a tried-and-test benchmark approach and on a standard data set. In each case it has been found that the use of trust values has a positive impact on overall prediction error rates with the best performing strategy reducing the average prediction error by 22% relative to the benchmark Resnick CF algorithm [102]. One issue with our implementation and testing of this technique was that the age of the dataset used may have influenced the result. The dataset contained only pre 2000 movies, and it was pointed out by a young participant in one of the later experiments on visualisation of trust information that the dataset contained a lot of "classics". It became a concern that users were tending to remember only movies that they liked. To check this possibility we asked users to rate the movies in a small user survey, (attached to the interaction surveys from Chapter 6). The distribution of these ratings were compared against those in the dataset, and no significant difference was found.

The key factor for the acquisition of a robust and useful trust model for a recommender system is a rich resource of historical data with high ratings overlap. Collaborative filtering systems which rely on ratings are subject to all of the problems discussed in the general limitations section above. The system will fail to compute good trust values if ratings data is too sparse. Trust between users is not necessarily transitive, just because Bob trusts Mary as a recommendation partner does not mean that Mary is a good partner for Bob. Trust is highly contextual: for example, Bob might be a good recommender for Mary when it comes to recommending cars, but he might be a bad recommender for clothing. Furthermore, due to the contextual nature of trust, and lack of standardisation across Social Web

applications, it is difficult to port trust data across different recommender systems. Integrating trust values back into a collaborative recommender system also poses some interesting challenges, for example: how should trust be combined with a producers contribution? what type of weighting should be used, should trust have properties such as temporal decay, or temporal strengthening?

8.5 Contribution 2: Study of Attacks on Trust-Based Recommenders

The second contribution of this work is an analysis of the resistance of different trust metrics to various forms of attack on trust-based recommender systems. This analysis identified some inherent drawbacks with trust-based systems in the face of such attacks. Much recent research has concentrated on attacking such systems. For example, O'Mahony's push and nuke attacks in [90], Kushmerick's attack evaluations in [62], Burke's bandwagon attacks in [15] and Mobasher's shilling attacks in [79]. However, even with continued research to find methods to identify and protect against attacks on personalisation systems, the threat will continue to exist since people stand to gain by manipulating the output of personalisation systems for a broad range of reasons. Chapter 4 of this thesis identifies a security problem that is inherent to trust-based recommender systems. Chapter 4 shows that this problem can manifest itself in terms of a larger prediction shift for an attacked item in a trust-based system than in a standard CF system. We now discuss this "reinforcement" problem as a limitation to trust-based systems.

During the analysis in chapter four the reinforcement problem was defined and highlighted in trust based systems as a robustness fault during attack situations. The reinforcement problem is the "backing up" effect that occurs between malicious users acting together in a collaborative system. For example, if there are 10 attacking users in a trust based recommender system that are all trying to push the rating for one target item. The attacking profile set could consist of a set of random popular items, given the same rating in each attack profile, and an unnaturally high rating for the target item in all

of the attack profiles. In this scenario, the attack users would all be considered to "back each other up" during the trust building process, and thereby achieve higher trust and a greater weight in the final output than regular users. Chapter 4 outlined several possible solutions that reduced prediction shift for an attacked item, including using an initial trusted set of users to build trust models: *CItem(genuine)*, ensuring a broad range of profiles in the modelling: *CItem(diverse)*, and enabling temporal weighting of profiles during the modelling stage *CItem(time)*. Results in chapter 4 showed that, although by a small margin, *CItem(genuine)* produced less of a prediction shift than the benchmark Resnick algorithm under a nuke attack on a target item. A key challenge to understanding robustness of computational processes within the Social Web is *uncertainty*. During these experiments we are aware that there is a specific type of attack going on. There are many different ways to target collaborative systems for manipulation, and no one defense to protect against all methods of attack. The Catch-22 of collaborative systems is that perhaps the only true way to ensure protection of these systems from potentially malicious users is to get rid of *all* the users. However there are ways to make attack more difficult, for example, it is generally more difficult to skew the results from a large system than a small one, and if cost of attack outweighs the benefit, then attacks are less likely to occur.

8.6 Contribution 3: Algorithm to Mine Trust from Recommendation Error

The next contribution of this work also applies to recommendation strategies but this time on the theory behind and our implementation of a trust-based recommender system based on history of errors in predictions generated by user profiles. This information is harnessed to compute a range value to be presented to the end-user. These recommendation ranges were presented in a prototype online recommender system, *BoozerChooser*. The goal was to show that presentation of a range of rating predictions is more likely to increase user trust in the recommendation system than presentation of absolute rating predictions. To evaluate the trust benefits resulting from the transparency of

our recommendation range techniques, we performed user-satisfaction trials on *BoozerChooser*, which recommended nightspots in Dublin city. Our user-satisfaction results show that the recommendation range techniques perform up to twice as well as the benchmark scalar predictions.

A recommendation range was computed by leveraging under/overestimate errors in users' past contributions in the recommendation process. When a producer profile overestimated a prediction, how much was it by, on average. Along with similar inforamtion for underestimates, the bottom of a range value was created by reducing the point recommendation by the average underestimate and the the top was calculated by increasing the point rating by the average overestimate. Three different models to compute this range were presented and evaluated. It was shown how this trust-based technique can be easily incorporated into a standard collaborative filtering algorithm through different weighting mechanisms. A "fair" comparison was defined in which our technique outperformed a benchmark algorithm in predictive accuracy. Results from the evaluations indicated that trust-aided recommendation based on error models increases predictive accuracy over the benchmark by up to 15%. It was also shown that there is a clear increase in the trust a user places in the recommendations they receive when they are presented in the form of a recommendation range derived from our error model. Looking back to the discussion on trust in earlier chapters, the recommendation range approach may provide an increase in system trust as defined by Marsh in [69]. This can enhance the experience a user has with recommendation system.

The notion of representing error and uncertainty in complex systems is not a new one, in fact, groups such as Conference on Uncertainty in Artificial Intelligence (UAI) have been studying this topic for a considerable length of time. Modelling uncertainty is especially important in multi-phase systems where errors and uncertainty can be multiplicative. Our study of this within the context of collaborative recommender systems was very limited, primarily due to our sparse dataset. The *BoozerChooser* recommender system was designed to collect ratings on entertainment spots in Dublin city, and present recommendations in the form of ranges to users. Because the data collected was highly sparse, the quality of recommendations generated tended to be

poor. As a result of this it was difficult to judge whether user satisfaction was simply a result of the mode of presentation, (ie: a range as opposed to a scalar prediction) or a result of the predictions generated by the underlying trust model. A remaining challenge for evaluation of this technique is the definition of a truly fair comparison between a scalar recommendation and a range of values.

8.7 Contribution 4: An Interactive Interface for Recommendation

The fourth core contribution of this work is *PeerChooser*, our interactive, graphical approach to recommendation. Chapter 6 detailed design, implementation and evaluation of the explanation interface and interactive feedback techniques of the application, and showed how both similarity and trust can be visualised in parallel. *PeerChooser* is a collaborative recommender system with an interactive interface which provides the user not only an explanation of the CF process, but the opportunity to manipulate their neighbourhood at varying levels of granularity to reflect aspects of their current mood and requirements. In addition to this, explicit expressions of trust are facilitated in the application's Social Web. *PeerChooser* models relations on the Social Web (in this case applied to a recommender system), and provides an interactive platform for a user to visualise and/or manipulate the underlying relationships between users on two dimensions: similarity and trust. By facilitating interaction during the recommendation process, we also overcome the problem of redundant profile information which exists in most current recommenders. The layout algorithm which acts on the nodes in the interface produces an exact, noiseless graph representation of the underlying correlations between users. PeerChooser's prediction component uses this graph directly to yield the same results as a benchmark ACF technique. User's then improve on these predictions by tweaking the graph to their current requirements.

PeerChooser proved to be popular with users, however, the current implementations have some limitations. Our initial attempts to produce a co-

herent visualisation of the CF process failed since the graphs produced were too complicated for users to understand. Representing complex information about the underlying network topology in a recommender system, as well as semantic information about each node in a two, or even three dimensional layout is a challenging problem. Our initial attempts used two and three dimensional layouts and tried to represent connections between users and items. This resulted in overly complicated visualisations. To get around this problem, either thresholding or some form of dimensionality reduction needs to be applied.

A further limitation was discovered relating to the interactive nature of our system. Interaction affords the user the opportunity to provide information about current requirements. It was found in many cases that users tended to over-tweak the visualisation, meaning that the influence from collaborative peers was reduced to an ineffective amount. This limitation can easily be addressed through further testing to correctly define weights for existing data, and user "tweaks". As a collaborative filtering system, the *PeerChooser* recommender is subject to general limitations such as data sparsity, early rater and cold start.

8.8 Contribution 5: Trust from Feedback Comments in Online Auctions

The contribution in Chapter 7 dealt with trust in online auctions using eBay as an example application and introduced *AuctionRules*, a new algorithm for capturing trust information from sentiment information found in user feedback comments on eBay. Chapter 7 also detailed the presentation of trust information to the end user, using a trust-based extension to eBay as our application. Values presented to end users included a graph of the trust relations between an active user and a target seller or buyer, the standard eBay trust score of the target, the AuctionRules trust score of the target (which was usually a lower value) and the context-specific trust scores of the target. For example, if user "bidAincent03" had a few comments that contained negative adjectives (possibly with intensifiers) around the keyword "packaging"

or any related synonym, then the contextual trust score for "bidAincent03"'s packaging was significantly reduced. It was also shown empirically that the AuctionRules classifier beats seven popular learning algorithms at comment classification task by up to 21% in accuracy tests using very minimal domain knowledge. These experiments were based on "ground-truth" obtained by manual classification of comments. AuctionRules has a favourable false negative rate for the classifier of 0% compared to up to 6% from benchmark algorithms and a false positive rate of 4.7% compared with 8.5% to 12.3% for the benchmark algorithms. This was a good result, but would have been even better if the false positive rate was lower, because saying an eBay seller/buyer is trustworthy when they are not is generally worse than the reverse.

The contributions from Chapter 7 included an algorithm for extraction of sentiment information from freetext comments, use of this information to build a trust model, and analysis of ways to facilitate propagation of trust information around the underlying social network. Limitations of NLP techniques used have been the subject of much research [97] [83] [50], and our implementations were lightweight compared to many existing NLP algorithms. The core issue with a propagation system for trust information is the assumption that trust, at some level holds the property of transitivity. Intuitively, if Bob trusts Mary and Mary trusts Joe, then Bob should probably trust Joe more than a complete stranger This is how it works socially, however Bob may trust Mary highly with respect to specific instances, but not trust Mary to give good recommendations about buyers and sellers. In this situation, the system loses the transitive property of trust. The work in Chapter 7 follows the findings of Josang in [57] in that propagation of trust is a regular occurrence in the real world, however, human trust has been generated through complex human interactions which are far too intricate to be expressed in full by our current computational models.

The notion of variance of trust across contexts also applies in our eBay experiments. Our findings show that there is a clear variance from our "granular" analysis of feedback comments, i.e: lots of users in the system are consistently good at one aspect of the transaction, such as timeliness for example and consistently bad at another, say packaging for example. Bearing this result in mind, we can see that although two uses may have the same overall

trust score in the online auction, they have a large contextual variance in reputation, and most likely also in the trust that people place in them with respect to specific situations. Consider for example, shipping a case of wine across the country, people may tend to trust the seller with better packaging reputation, whereas if a concert ticket was being shipped close to the concert date, people may tend to trust the seller with the history of, or reputation for expedited delivery. This is expressed mathematically in the following simple equation, for users i and j and contexts p and q where $p \neq q$.

$$(trust_i(j)|p) \neq (trust_i(j)|q) \tag{8.2}$$

This concludes the discussion of the core contributions of this thesis, and limitations associated with each. The following section details some potential avenues of research on the topic of modelling trust in Social Web applications.

8.9 Future Work

This research has explored a broad range of techniques and applications related to the computation of trust information on the Social Web. As with the vast majority of theses, this research remains "unfinished" as there are countless new techniques and technologies to be tested on the ever-evolving platform of the Social Web. The following sections discuss a few potential directions for future research in this area, at two distinct levels. Firstly, a discussion of the more immediate and obvious progressions related to the specific algorithms developed in this work, and secondly a more broad and speculative discussion of the future of Social Web trust models is presented.

8.9.1 Trust Models in Recommendation

Our trust models for recommender systems can be can also be used to generate a broad *recommendation explanation*. By using trust scores we are able to say to a user (in a car recommender, for example) "You have been recommended a Toyota Landcruiser; This recommendation has been generated by users A, B and C, and these users have successfully recommended Toyota Landcruisers X, Y and Z times in the past, and furthermore, P, Q,

and R% respectively of their overall recommendations have been successful in the past." This recommendation accountability is a very influential factor in increasing the faith a user places in the recommendation. One possible direction to further this work would be the implementation of a "transparent" recommender system in which users can easily access such information.

8.9.2 Trust Models in Online Auctions

A new technique for trust modelling within feedback-based online auctions was developed, using eBay as an example domain. In the implemented system, numerical trust values are mined directly from user comments using *AuctionRules*, a new classification algorithm which captures negativity in user comments. Our system also facilitates the propagation of the trust values throughout the social network that is formed by the eBay marketplace. Propagation of trust values allows us to compute a *personalized* trust score on a prospective seller to be presented to a buyer. Further research in this area includes testing the accuracy and coverage of the trust-propagation mechanisms, testing the query expansion module and refining the *AuctionRules* algorithm to produce higher accuracy.

A second avenue of exploration stems from the portability discussion presented earlier. It would be very interesting to see a trust model built up on one Social Web application and used to benefit another in some way. For example, taking the trust or reputation information built up from eBay's feedback system and using this information somewhere like Amazon.com or Yahoo auctions. One of the first hurdles of this task would be the mapping problem. How does the algorithm relate profiles between disparate applications? One approach may be to use a centralised trust-server where willing participants can enter multiple usernames from diverse applications in exchange for rich trust information on other users in each of those applications. Another approach, in the online auction domain might be to link users by product descriptions and prices. Frequently, sellers use more than one platform to peddle their wares. A relatively simple web crawler could easily detect repeated itemsets from diverse online auctions and determine probabilities that the associated usernames are in fact the same person or

company. The danger with this approach however is that it could easily be viewed as an infringement on privacy. Another drawback is that buyer trust information is difficult to attain in this manner since the majority of buyers do not perform seller roles.

8.9.3 Interaction and Visualisation of Trust Information

Several additions to the existing *PeerChooser* system are currently in progress, and this work will be continued. The Netflix prize * dataset consists of very recent movies and is a rich resource for ratings information. This dataset is currently integrated into the *PeerChooser* system. However, the publicly released data does not contain genre information for the items. We are in the process of implementing a web crawling algorithm to automatically tag each movie with its appropriate genre information, in a similar manner to MovieLens, so we can use *PeerChooser* on the data. An online version of *PeerChooser* has been made available to test the system over much larger user trials. However, this implementation used Java WebStart technology, and the current implementation is large and has a long download time. A more lightweight, web-based version of the application would be required to attract a larger user base, for example a lightweight Java applet or preferably an Adobe Flash application.

PeerChooser could benefit from more diverse mappings from the complex rating information space into a two or three dimensional graph that is simple enough to be understood by the average shopper on Amazon.com. This is a very challenging problem which is the subject of much research attention in Graphics communities such as SIGGRAPH and InfoViz. Any such mapping is bound to be a lossy computation, but the key is to maintain the most salient information that can be used to generate well informed recommendations.

*www.netflixprize.com

8.9.4 Trust Models and Facebook API

At the outset of this research, a student at Harvard University, Mark Zuckerberg was developing a Social Web application to "rate peoples looks". Now in 2008 his application, Facebook, has become a social networking giant with a user base of 67 million. It is currently growing at a rate of 250,000 new users per day. An interesting point of note is that this ubiquitous social networking application, which is bound to have a major impact on social networking research (and society in general), has its roots in the same place that Stanley Milgram performed his small world social networking experiments in the 1960's. A quick search on Facebook reveals many attempts to repeat Milgram's early experiments by getting massive numbers of users to join groups and perform simple tasks. The membership of these groups alone runs into millions of users. To put the growth of social networking into perspective, Milgram posted 296 letters to test his small worlds theory, and only a percentage of these got through to their destination. Facebook has a highly interconnected network of 67 million users who have expressed explicit trust statements about each other. This information has recently been made readily accessible through the Facebook API.

Facebook API consists of a list of functions which allows an application to access information from the Facebook databases. Of course, not all of the information is directly available due to privacy restrictions. However, information about the topology of the social network is accessible, in that any registered user can access the social links of any other user. This information alone is sure to be a highly valuable resource in the study of social networks. More specific to the work on recommender systems presented in this thesis, Facebook contains rich information about the movie and music preferences of its users. Accordingly, the Facebook API should have a major impact on recommender system research, and future experiments on collaborative filtering should harness this rich data resource. In order to get user preference information from Facebook, users will have to add a recommender application and "agree" to let the application see the data. Unlike many existing systems which require users to sign up in some way or other, Facebook have made the process extremely simple and "familiar" to users, and adding an application

is done with a single click. From a the perspective of trust research for recommender systems, the availability of this wealth of data raises some interesting questions: if a user has explicitly expressed trust in another user by adding that user as a friend, will that user be any better than a complete stranger as a recommendation partner? Because movie preferences are co-located with music and social preferences, can ratings information from these diverse items be used to compute better recommendation partners? What effect will demographic, geographic, religious, political and other information have on the quality of recommendation partners. Since all of this information is readily available in easily accessible formats, it will be interesting to see if we can find answers to some of the above by testing new recommender systems in the form of Facebook applications.

8.10 Conclusions

This thesis has explored the varying concepts of trust and reputation on the Social Web. The research has focused on two main application areas, namely recommender systems and online auctions. Throughout this work, the problems associated with using trust in both application domains have been explored theoretically, and numerous technical solutions been analysed, designed, implemented and tested. In particular, this work has contributed a technique for modelling trust in recommender systems based on users' history of contributions to the recommendation process. This trust model can be used to improve the accuracy of collaborative filtering recommendations by up to 22% over the benchmark Resnick prediction algorithm. This work was extended to include a trust model for recommenders based on errors in predictions from profiles producing recommendations, and a detailed examination of the robustness of trust based recommenders was carried out. To study the role of trust within the online auction domain, a trust-mining algorithm was developed to harnesses the wealth of information hidden in online auction feedback comments and provide new information about the reputations of potential transactors. The *AuctionRules* trust model is based on lightweight natural language processing over free text comments. To further investigate ways in which trust information can be used in recommendation

applications, a system was developed to provide a visual explanation of the computational processes involved in generating recommendations, and to allow users to visually interact with the computation to manipulate its output.

Since the outset of this work in 2004, there has been a huge growth in the Social Web and the research associated with it. Despite the large amount of recent research dedicated to the Social Web and reputations of users within it, there is a lot to be done before we can assess trust in the online world as we do in our everyday lives. I believe that Social Web applications will proliferate through industry, government and scientific research communities alike, and that the ability to trust the providers of information in these systems is vitally important for society as a whole.

Appendix A

PeerChooser User Trial Questionnaire

A.1 Introduction

This appendix contains the survey questions given in the experiments with *PeerChooser* in Chapter 9. On average the survey took about 20 minutes and was fully supervised. Firstly, participants answered the pre-study questions, then they were guided through the application section with the help of an instructor. This was done to ensure the participants understood their tasks properly. In all cases, participants quickly understood the tasks to be performed. On completion of the application section each user was required to answer a post-survey questionnaire. Full detail is given in this appendix.

A.2 Pre-study Questionnaire

1. Age:
2. Gender: (Male — Female)
3. Have you used a recommender system with before (where you rate items such as movies, books, or music for example and then get recommendations for other items)? (in some cases making ratings can be as simple as 'purchasing' an item on a web site. (YES or NO)
4. If yes, which specific ones do you remember using.

I am familiar with the concept of recommender systems

Strongly Agree	Agree	Not Sure	Disagree	Strongly Disagree
1	2	3	4	5

I am familiar with interactive computer graphics (e.g. computer simulations, games, visualisations)

Strongly Agree	Agree	Not Sure	Disagree	Strongly Disagree
1	2	3	4	5

I am familiar with the concept of graph visualisations (systems of nodes and edges, representing relationships among entities)?

Strongly Agree	Agree	Not Sure	Disagree	Strongly Disagree
1	2	3	4	5

What do you think are the 5 most popular movie genres in order? Please write 1 to 5 in front of the genres in Figure A.1:

Action	Adventure	Animation	Children's	Comedy	Crime
Documentary	Drama	Fantasy	Film-Noir	Horror	Musical
Mystery	Romance	Sci-Fi	Thriller	War	Western

Table A.1: Common Categories of Movie Genre.

A.3 Study Instructions

You will be shown a computer program called PeerChooser. This is a program for visualising the process of recommendation –predicting items that you, the user, will like based on your similarity with other users in the system. The unique advantage of this system is that it lets you "look inside" recommendation algorithms such as that on Amazon.com.

The system will be explained by the supervisor (edges, icons, ratings etc) and you will be asked to perform simple manipulation tasks to familiarise yourself with the system.

1. Familiarisation Task: Get a specific recommendation for user 10 for the movie "Home Alone". Now assume you are in the mood for comedy, and move some of your neighbours who like Comedy towards your avatar. What happens to your predicted rating? Now move some Comedy peers away. Again, note the change in the predicted rating.

2. Profile Generation: Explicit Rating Please go to the profile panel on the right side of the PeerChooser application. Rate 30 movies that you have seen from the given list. Press the "update profile" button. The system will now correlate your information with every other user in the database.

3. Average Recommendation Task You will be shown a list of generic recommendations. Go through the list and rate any 10 of the recommended movies on a scale of 1 to 5 (1 = strong dislike, 5 = strong like) . Click Submit. For this each of the following tasks you will be shown the generated predictions at the end of the task.

4. Graph Manipulation Task A You will be shown the same list of generic recommendations, now take 1 min to manipulate the graph to your liking. New recommendations will be generated. Go through the list and rate any 10 of the recommended movies on a scale of 1 to 5 (1 = strong dislike, 5 = strong like) Click Submit. (supervisor then saves graph)

5. Profile-Based Recommendation. You will be shown recommendations based on your profile data. Go through the list and rate any 10 of the recommended movies on a scale of 1 to 5 (1 = strong dislike, 5 = strong like) Click Submit.

6. Graph Manipulation Task B You will be shown the same list of profile-based recommendations, now take 1 min to manipulate the graph to your liking. New recommendations will be generated. Go through the list and rate any 10 of the recommended movies on a scale of 1 to 5 (1 = strong dislike, 5 = strong like) (supervisor then saves graph) Click Submit.

7. Result-Based Recommendation. Check the box beside 10 movies you have seen. (liked OR disliked). Now, get recommendations for these movies. Manipulate the graph until you are satisfied with the predicted ratings for your selected movies. Now you will be shown a list of predictions on other movies based on your optimised graph layout.

8. Go through the list and rate any 10 of the recommended movies on a scale of 1 to 5 (1 = strong dislike, 5 = strong like) (supervisor then saves graph)

A.4 Post-Study Questionnaire

Which of the techniques for predicting your movie ratings did you prefer (considering time spent, accuracy etc). Please order 1-4 where 1 is your favourite. (Please give a short reason for your choice.)

- Manipulating an average Graph
- Manipulating your personalised Graph (based on your 30-ratings profile)
- Rating 30 movies only.
- Tweaking the graph to optimise predictions on the movies you knew, then getting new ones.

Did you feel that the improvement by manipulation (if any) was worth the time spent examining the graph.

Visual Placement				30-Rating-Profile
1	2	3	4	5

Please rate the following statement: I found the neighbourhood graph easy to understand

Strongly Agree	Agree	Not Sure	Disagree	Strongly Disagree
1	2	3	4	5

Look at the following graphs and rate them on a scale of 1 to 5 depending on which one you would prefer for "placing yourself in"?

Strongly Agree	Agree	Not Sure	Disagree	Strongly Disagree
1	2	3	4	5

Which of the techniques for predicting your movie ratings did you feel predicted your ratings more accurately: Please order 1-4 where 1 is your favourite.

- Manipulating an average Graph
- Manipulating your personalised Graph (based on your 30-ratings profile)
- Rating 30 movies only.
- Tweaking the graph to optimise predictions on the movies you knew, then getting new ones.

Any other Comments/Issues/Suggestions?

Above is an example of how PeerChooser could be deployed on Amazon.com. Would you like to have control over your predicted ratings by a system such as PeerChooser in a real-life recommender system such as Amazon, or even on Facebook? (Please give a short reason for your choice.)

Strongly Agree	Agree	Not Sure	Disagree	Strongly Disagree
1	2	3	4	5

Please rate the following statement: I gained knowledge about the underlying data from the visualisations.

Strongly Agree	Agree	Not Sure	Disagree	Strongly Disagree
1	2	3	4	5

Based on the visualisations you have seen, what do you think are the five most popular movie genres in order? Please fill in 1 to 5 in the boxes:

Action	Adventure	Animation	Children's	Comedy	Crime
Documentary	Drama	Fantasy	Film-Noir	Horror	Musical
Mystery	Romance	Sci-Fi	Thriller	War	Western

Table A.2: Common Categories of Movie Genre.

I felt that the labelling was appropriate.

Strongly Agree	Agree	Not Sure	Disagree	Strongly Disagree
1	2	3	4	5

I felt that the icons communicated the genres appropriately.

Strongly Agree	Agree	Not Sure	Disagree	Strongly Disagree
1	2	3	4	5

I found the user information in the right panel helpful.

Strongly Agree	Agree	Not Sure	Disagree	Strongly Disagree
1	2	3	4	5

I found the visualisation system gave me more control over the operation of the recommendation algorithms. (Please give a short explanation of your answer.)

Strongly Agree	Agree	Not Sure	Disagree	Strongly Disagree
1	2	3	4	5

I feel that I benefited from the interaction with the system

Strongly Agree	Agree	Not Sure	Disagree	Strongly Disagree
1	2	3	4	5

I feel that the graph interface provided an explanation of the recommendation process.

Strongly Agree	Agree	Not Sure	Disagree	Strongly Disagree
1	2	3	4	5

Appendix B

Supplementary Background on Trust

B.1 Introduction

The idea of using trust metrics in Social Web applications is very recently conceived. Despite this, there has been a high level of research interest placed in this subject area, most likely as a result of the massive growth in online marketplaces and social networks. Results of a recent survey [40] show that there are nearly half a billion user accounts among hundreds of Social Web applications. Furthermore [40] points out that these accounts are not just one-time participators, since several of the top ten most visited sites on the Web are social networking sites. This appendix presents an analysis of past and present research on trust metrics especially where they are included, or have relevance to Social Web applications.

The appendix is supplementary to the background presented in chapter three. Analysis of past and current research into the varying notions of trust is presented. This analysis covers varying concepts of trust across several relevant research disciplines, from computational, social and psychological perspectives. For each discipline, ideas from the leading theoreticians are discussed and compared. This is relevant to our work as it presents a more broad view of what exactly is meant by "trust", and it places our *computational view of trust* into context.

In psychology and sociology, a trust metric is a measure of how a member of a group is trusted by the other members. Trust metrics may be abstracted

in a manner that can be implemented on computers, making them of interest for the study and engineering of virtual communities, such as Friendster and LiveJournal. Attack resistance is an important property of trust metrics which reflects their ability to handle agents who participate in bad faith (i.e. who aim to abuse the presumption of trust). The first forms of trust metrics in computer software were in applications like eBay's Feedback Rating. Slashdot introduced its notion of *karma*, earned for activities perceived to promote group effectiveness, an approach that has been very influential in later virtual communities. Across different disciplines, definitions for trust vary. The following sections give a brief synopsis of some of the prominent notions of trust from the major relevant disciplines.

Although this thesis is concerned with trust in a computational sense, it is important to consider the range of meanings covered by the term "trust". Despite the large amount of research carried out in this area across many disciplines, an exact definition for trust has remained somewhat elusive [69]. The word trust is derived from the old Norse word for "strong". The Oxford English dictionary defines trust as:

1. *noun*: Firm belief in the reliability, truth, ability, or strength of someone or something.
2. acceptance of the truth of a statement without evidence or investigation.
3. the state of being responsible for someone or something.
4. Law an arrangement whereby a person (a trustee) is made the nominal owner of property to be held or used for the benefit of one or more others.
5. a body of trustees, or an organisation or company managed by trustees.

1. *verb* have trust in.
2. (trust with) have the confidence to allow (someone) to have, use, or look after.

3. (trust to) commit (someone or something) to the safekeeping of.
4. (trust to) place reliance on (luck, fate, etc.).
5. have confidence; hope: I trust that you have enjoyed this book.

B.1.1 Trust in Sociology

Sociological definitions of trust play an increasingly important role in complex systems on the web as users begin to provide a larger percentage of the content, and consequently the web becomes more "social". There are a large number of writings on issues of trust from sociology researchers. In this section, some of the leading opinions are detailed.

James Coleman

In sociology, a good definition is given in by James S. Coleman in "Foundations of Social Theory" [29]. Coleman offers a four part definition:

- Placement of trust allows actions that otherwise are not possible.
- If the person in whom trust is placed (trustee) is trustworthy, then the trustor will be better off than if he or she had not trusted. Conversely, if the trustee is not trustworthy, then the trustor will be worse off than if he or she had not trusted.
- Trust is an action that involves the voluntary placement of resources (physical, financial, intellectual, or temporal) at the disposal of the trustee with no real commitment from the trustee.
- A time lag exists between the extension of trust and the result of the trusting behaviour.

With the exception of the third point, all of the above list can be applied and are relevant to the Social Web. In the context of online auctions for example, Coleman's third point above regarding the truster placing faith in the trustee with *no* real commitment from the trustee is not valid. A trustee who defects in an online auction situation will most likely be given a bad review which

will have a detrimental effect on the trustees reputation. This can be viewed as a cost/commitment for the trustee. A similar argument may be applied for other Social Web sites, such as the Facebook social network site [64]. At a system level, trust is placed in anyone with a university email address as this is the basis upon which accounts are granted. Coleman might see it that there is no real commitment from the user in this situation. The counter argument is that trust is universally positive and highly coveted and that the potential for loss of trust itself is a commitment from the user.

Using the Facebook as an example, if a user abuses account privileges, trust will be lost in that user and the account will be suspended. However, a big difference between the real and online world is the ability to create a new persona. In a Social Web application such as Facebook, or auction site such as eBay, the only thing required for a distrusted user to make a completely fresh start is an email address. It is not quite so easy in the real world! To think that any new user in a Social Web application like eBay could have been distrusted or banned the previous day is a troubling thought. However, there is an upside: For "well-behaved" users, as profiles mature, so too does reputation. As this value increases, the owner of that profile should be less and less likely to want to tarnish that reputation, and hence be more trustworthy.

Keeping with the Facebook application as an example, trust is placed in users by other users by explicitly specifying a "friendship" connection with them. When this 'trust' is earned, users are granted access to other users information. The potential for loss of trust by a trustee, and therefore loss of all the related privileges which accompany the trust can be considered as a cost/commitment from the trustee in itself. One may also consider the concept that being generally much less frequent in Social Web sites, news of the more rare examples of *distrust* may propagate further around the network than standard trust which exist between most users.

Gaia diLuzio

DiLuzio [23] performs a sociological analysis of client trust in [23]. DiLuzio develops axioms of client trust and use that model to explain why and un-

der what conditions the professional/client relationship (in contrast to other service relationships) involves a certain kind of trust. During the analysis in [23], DiLuzio defines an interesting viewpoint on trust which will be useful later in this work in our online auction studies. This viewpoint, in contrast to Marsh's view on interpersonal trust [69] (discussed in Section B.1.3) is that *impersonal* trust exists in the interpersonal relationship between a client and a professional. When the client relies on the special competencies and the confidentiality of the professional, it is not the latters personality or personal trust/reputation that plays the leading role in the anticipation of a positive outcome of treatment, but the trust and reputation that the client associates with the professional because he is a member of that profession.

In the context of the Social Web, DiLuzio's argument about the *impersonal* trust existing between two users in *interpersonal* communication is very relevant. Take for example a "Power Seller" on eBay. This group of sellers is perceived as elite amongst regular eBay sellers. Buyers in the marketplace prefer to conduct business with Power Sellers. However, just because a seller is a member of this group does not automatically mean that the seller should be highly trusted.

The following are the criteria upon which eBay designates power sellers:

- Have been an active member for 90 days
- Average a minimum of $1000 in sales per month, for three consecutive months
- Achieve an overall Feedback rating of which 98% or more is positive
- Have an account in good financial standing

DiLuzio suggested that there may be increased trust between a client and a professional, because he is a member of a professional group. In the real world, this works well, because there are measures in place to ensure the quality of members of such groups. In the online world, there are many cases where the requirements for membership of "elite" groups are not so clear. Looking again at the Power Seller example: Many buyers would glance at the range of important looking icons beside a sellers name and assume he is

trustworthy. Applying diLuzio's argument about the existence of impersonal trust to the relation between a buyer and a power seller on eBay and looking in particular at the third point in the above list, (a required feedback score of 98% positive or more) an interesting point arises: On eBay, a buyer might trust a seller because he is a member of an "elite" group of power sellers. However, an analysis of the feedback comments on eBay shows that 98.5% of feedback comments were found to be positive, meaning that although a buyer may hold an elite impression or *impersonal trust* for a Power Seller because of his association with that group, it is very possible that the seller may actually be of *less than average trustworthiness*. The results and details of the survey are presented at the end of this chapter in Section 7.2.1.

Diego Gambetta

Diego Gambetta's essays on trust were originally published in 1988 and republished in an online version in 2000 [33]. This work contributes greatly to the literature on the subject of trust. The closing essay in this collection, by Gambetta himself is entitled "Can we trust trust", and provides and interesting and relevant discussion. Gambetta proposes that trust can be assigned a probabilistic value between 0 and 1. This probability can be dependent on a range of external factors that define the context of the trust. Marsh gives a very detailed discussion and analyses of Gambetta's work in [69] concluding that it is interesting because of the similarity in theoretical approach (they both assign a value range to trust, and agree on the contextual variance), but they have different contexts. (Marsh discussed trust in [69] in the context of intelligent agents, whereas Gambetta's viewpoint purely sociological.) Gambetta proposes some relevant definitions in the 2000 work. [33]. At the end of the first definition the importance of context of trust is mentioned, which is highly relevant to the methods presented later in this work.

1. *Trust (or, symmetrically, distrust) is a particular level of the subjective probability with which an agent assesses that another agent or group of agents will perform a particular action, both before he can monitor such action (or independently of his capacity ever to be able to monitor it) and in a context in which it affects his own action.*

2. *When we say we trust someone or that someone is trustworthy, we implicitly mean that the probability that he will perform an action that is beneficial or at least not detrimental to us is high enough for us to consider engaging in some form of cooperation with him.*

3. *Correspondingly when we say that someone is untrustworthy, we imply that probability is low enough for us to refrain from doing so.*

Gambetta also argues that there are many cases where trust is not relevant. According to [33] trust is only relevant if a possibility exists of distrust, betrayal, exit or defection. If these possibilities are not present then trust itself is not necessary. In the context of the Social Web there are cases where this may apply to some extent, for example Resnick points out in his analysis of the eBay marketplace in [103] that the trust a seller must place in a buyer is small compared with the buyers trust in the seller, since the buyer will hold the item until payment has been made by the seller. In theory, Gambetta's point is relevant because there is no grounds for distrust, defection or exit once the seller has received the money. In general however, this is not the case. Where users provide content in the Social Web at the very least there is the possibility of misinformation, which is grounds in itself for distrust. This argument implies that trust is relevant throughout the Social Web

Gambetta's point is relevant in part to the relationship between a buyer and a seller in an online auction such as eBay for example. Generally, in the buyer-seller relationship the seller is not required to perform any action until money is received from the buyer. Once this is done, the seller relies on no further action (barring a feedback comment) from the buyer, so largely, seller trust in a buyer is not as *relevant* as its reciprocal.

B.1.2 Trust in Psychology

Norton Deutsche

Many research papers on trust have used Deutsche's work [22] as a starting point for their analyses. (e.g: Marsh [69], Josang [56]). Following is an extract from Deutsche's 1962 work which outlines his three main ideas for trust:

If an individual is confronted with an ambiguous path, a path that can lead to an event perceived to be beneficial (Va^+) *or to an event perceived to be harmful* (Va^-)*; he perceives that the occurrence of* (Va^+) *or* (Va^-) *is contingent on the behaviour of another person; and he perceives that(sic) the strength of* (Va^-) *to be greater than the strength of* (Va^+)*. If he chooses to take an ambiguous path with such properties, I shall say he makes a trusting choice; if he chooses not to take the path, he makes a distrustful choice.*

Deutsche's work on the subject of trust is probably one of the most important contributions to the subject. His 1973 work [22] outlines core facets of trust. These facets are listed below, with addenda marking relevance to the work in this thesis.

- Trust as despair: this occurs when the negative consequences of not trusting, or of staying in the present situation, are so great or so certain that the trusting choice is made out of despair. [Trust] is the lesser of two evils. As our online markets grow, users are being forced to conduct more of their daily business online, with very little information by which to judge the other party. In a sense, users are forced to trust people they are conducting their online business with.

- Trust as social conformity: in many situations, trust is expected, and violations lead to severe sanctions. [An individual decides to trust otherwise] he may end up socially ostracised or labelled a coward.

- Trust as innocence: the choice of a course of action may be made upon little understanding of the dangers inherent in the choice. This innocence may be rooted in lack of information, cognitive immaturity, or cognitive defect (pathological trust). Assuming that people are aware there is some risk involved in online business this should not be an issue, however, there are always some who will be oblivious to the fact that fraud does occur in the online world just as in the real world.

- Trust as impulsiveness: inappropriate weight may be given to the future consequences of a trusting choice.

- Trust as virtue: cooperative action and friendly social relations are

predicated upon mutual trust and trustworthiness. Thus trust is naturally considered a virtue in social life.

- Trust as masochism: pain and unfulfilled trust may be preferable to pleasure. Since people tend to try to confirm prior expectations, they will trust negatively, and generally find their expectations satisfied.

- Trust as faith: [the trusting person] may have faith in preordained paths which mean that whatever awaits is fated and thus to be welcomed. Having faith to a large extent removes the negative consequences of a trusting decision.

- Risk-taking or gambling: if the potential gains of winning are subjectively far greater than the potential losses from losing even should the risk be great, the gambling individual would be prepared to take that risk. This form of trust outlined in Deustch's work is highly relevant in the online world. Recently there has been explosive growth in the area of online gambling. Online poker sites are the envy of dot-com observers: Online poker revenues have grown from $82.7 million in 2001 to $2.4 billion (all numbers US) in 2005; last year, more than $60 billion was gambled on poker sites; and every day, 1.8 million players toss their ante into the virtual pots of the Internet.*

- Trust as confidence: here one trusts because one has confidence one will find what is desired rather than what is feared. When Deustch uses the word fear here, he is implying that in order to trust, one must first take a risk.

Niklas Luhmann

Luhmann contributed greatly to many disciplines, including economics, law, mathematics and psychology. His 1979 work [68] provides a groundwork for research on trust. Luhmann conforms to the idea that trust is highly contextual, and defines trust as "a risky investment" in that a trusting agent can never be sure of the actions of the trusted agent.

*http://www.digitaljournal.com/news/?articleID=4388

Luhmann agrees with Gambetta's theory that trust can be assigned a probabilistic value varying from "distrust" to "blind trust". Luhmann argues that trust is used in everyday life to reduce complexity and that it is built over very complex human information [68]. He poses the question of how to use trust in complex systems to reduce complexity and provides a functional strategy and an intentional strategy in [68]. Work in 2004 by Anders Bordum [9] furthers Luhmann's ideas on trust as a complexity-reducing mechanism, arguing that knowledge-based reduction of complexity should occur in complex systems but (in contrast to Luhmann [68]) where "an empirical analysis of trust and its social distributions is always connected with validity claims." Essentially, Bordum disagreed with Luhmann's omission of using existing knowledge to help in the trusting decision. These ideas are applied to collaborative filtering later in this work, where existing system knowledge is harnessed to aid in trust-based decisions.

An argument for the importance of trust in recommender systems, dealt with in detail in Chapter 3 follows from Luhmann's 1979 work: In highly complex systems such as recommenders, and indeed the Social Web in general where there is definitely an information overload [85], the complexity of the problem of finding relevant information can be reduced for the end-user by the addition of trust metrics to the system.

B.1.3 A Computational View of Trust

This work is concerned with the semantic web, social networking, virtual communities and recommender systems: these are all examples of research areas where issues of trust, reputation and reliability are becoming increasingly important, especially as we see work progress from the comfort of the research lab to the hostile real-world.

Marsh's work in [69] goes some way towards formalising trust in a computational sense, taking into account both it's social and technological aspects. Marsh's unpublished but highly cited Ph.D thesis deals with trust over a network of distributed agents. One factor in this theoretical work which makes it applicable to the Social Web is the assumption that agents can "remember" events and behaviours of other agents in the system. On the Social Web,

it is possible to record *every* action, event and behaviour that an agent has been associated with.

B.2 Summary

The focus of this research is an analysis of the role of trust on the Social Web. This supplement has analysed the different meanings of trust across a range of disciplines, and helps to defined the focus in the thesis to be on the *computational* properties of trust.

Appendix C

Code Snippets of Implemented Actions in the *PeerChooser* Interface

```
/**
 * Recommender Network Visualizer that draws entities, relationships and
 * communicaitons
 */
public class RNV extends Distractibilities
{
//=====================================
public static boolean labels_switch = true;
//=====================================
private static final String PAN_DOWN = "Pan Down";

private static final String PAN_UP = "Pan Up";

private static final String PAN_RIGHT = "Pan Right";

private static final String PAN_LEFT = "Pan Left";

/** Name of an action. */
public static final String ARRANGE = "Arrange";

/** Name of an action. */
```

```
public static final String ARRANGE_INCREMENTAL
                                        = "ArrangeIncremental";

/** Name of an action. */
public static final String ZOOM_FIT = "Zoom Fit";

/** Name of an action. */
private static final String ZOOM_SELECTION = "Zoom Selection";

/** Name of an action. */
public static final String ZOOM_OUT = "Zoom Out";

/** Name of an action. */
public static final String ZOOM_IN = "Zoom In";

/** Name of an action. */
public static final String INCREASE_DETAIL = "Increase Detail";

/** Name of an action. */
public static final String DECREASE_DETAIL = "Decrease Detail";

/** Name of an action. */
public static final String CHANGE_EDGE_RENDERING
                                        = "Change Edge Rendering";

/** Name of an action. */
public static final String TOGGLE_DISPLAY = "Toggle Display";

/** Name of an action. */
public static final String TOGGLE_ACTIVE = "Toggle Active";

/** Name of an action. */
public static final String INCREASE_RESTING_DISTANCE
                                    = "Increase Resting Distance";

/** Name of an action. */
```

```
public static final String DECREASE_RESTING_DISTANCE
                                    = "Decrease Resting Distance";

/** Name of an action. */
public static final String INCREASE_REPELLING_INTENSITY
                                    = "Increase Repelling Intensity";

/** Name of an action. */
public static final String DECREASE_REPELLING_INTENSITY
                                    = "Decrease Repelling Intensity";

/** Name of an action. */
public static final String INCREASE_K_VALUE = "Increase K Value";

/** Name of an action. */
public static final String DECREASE_K_VALUE = "Decrease K Value";

/** Name of an action. */
public static final String TOGGLE_3D = "3D / 2D";

/** Name of an action. */
public static final String TOGGLE_VIEW = "Toggle view";

/** Name of an action. */
public static final String INCREASE_CLUSTERS
                                    = "Increase number of clusters";

/** Name of an action. */
public static final String DECREASE_CLUSTERS
                                    = "Decrease number of clusters";

/** Name of an action. */
public static final String DEFAULT_CLUSTERS
                                    = "Default number of clusters";

/** Name of an action. */
```

```
public static final String INCREASE_NODE_SIZE = "Increase Node Size";

/** Name of an action. */
public static final String DECREASE_NODE_SIZE = "Decrease Node Size";

/** Name of an action. */
public static final String HIDE_DOCS = "Hide Docs";

/** Name of an action. */
public static final String HIDE_LEAVES = "Hide Leaves";

/** Name of an action. */
public static final String HIDE_TEXT = "Hide Text";
```

Bibliography

[1] Fortune 500. 2006 fortune 500 listings. *http://money.cnn.com/magazines/fortune/fortune500/snapshots/69.html*, Apr 17 2006.

[2] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *Proceedings of the 20th International Conference on Very Large Data Bases, VLDB*, pages 487–499. Morgan Kaufmann, Spetember 12th 1994.

[3] M. K. Albert and D. W. Aha. Analyses of instance-based learning algorithms. In *Proc. of AAAI-91*, pages 553–558, Anaheim, CA, 1991.

[4] Paolo Avesani, Paolo Massa, and Roberto Tiella. Moleskiing: a trust-aware decentralized recommender system. *1st Workshop on Friend of a Friend, Social Networking and the Semantic Web. Galway, Ireland*, 2004.

[5] Paolo Avesani, Paolo Massa, and Roberto Tiella. A trust-enhanced recommender system application: Moleskiing. In *SAC '05: Proceedings of the 2005 ACM symposium onApplied computing*, pages 1589–1593, New York, NY, USA, 2005. ACM Press.

[6] Albert-Laszlo Barabási. *Linked: How Everything Is Connected to Everything Else and What It Means for Business, Science, and Everyday Life.* Plume Books, April 2003.

[7] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, 284(5):34–43, 2001.

[8] Daniel Billsus and Michael J. Pazzani. Learning collaborative information filters. In *Proceedings of the 15th International Conference on Machine Learning*, pages 46–54. Morgan Kaufmann, San Francisco, CA, 1998.

[9] Anders Bordum. Trust as a critical concept. In *Proceedings of the 2006 Conference on Philosophy and the Social Sciences*, Prague.

[10] Christian Borgelt and Rudolf Kruse. Induction of association rules: Apriori implementation. In *Proceedings of the 15th International Conference on Computational Statistics (Compstat 2002, Berlin, Germany)*, Heidelberg, Germany, 2002. Physika Verlag.

[11] Terry Bossomaier. Linked: The new science of networks by albert-laszlo barabási. *Artif. Life*, 11(3):401–402, 2005.

[12] John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In Gregory F. Cooper and Serafín Moral, editors, *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 43–52, San Francisco, July 1998. Morgan Kaufmann.

[13] Brian Burke. Ebay town-hall meeting. Transcript, Senior Manager, eBay Marketplace Rules and Policy, eBay Inc, San Jose, CA. Feb 28th. 2006. available at http://pics.ebay.com/aw/pics/commdev/TownHallTranscript_022806.pdf.

[14] Robin Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.

[15] Robin Burke, Bamshad Mobasher, Roman Zabicki, and Runa Bhaumik. Identifying attack models for secure recommendation. In *Beyond Personalisation Workshop at the International Conferece on Intelligent User Interfaces*, pages 347–361, San Deigo, USA., 2005. ACM Press.

[16] Robin D. Burke, Kristian J. Hammond, and Benjamin C. Young. The findme approach to assisted browsing. *IEEE Expert*, 12(4):32–40, 1997.

[17] Jill Burstein, Karen Kukich, Susanne Wolff, Chi Lu, and Martin Chodorow. Enriching automated essay scoring using discourse marking. In Manfred

Stede, Leo Wanner, and Eduard Hovy, editors, *Discourse Relations and Discourse Markers: Proceedings of the Conference*, pages 15–21. Association for Computational Linguistics, Somerset, New Jersey, 1998.

[18] Sonny Han Seng Chee, Jiawei Han, and Ke Wang. Rectree: An efficient collaborative filtering method. In *DaWaK '01: Proceedings of the Third International Conference on Data Warehousing and Knowledge Discovery*, pages 141–151. Springer-Verlag, 2001.

[19] M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes, and M. Sartin. Combining content-based and collaborative filters in an online newspaper, proceedings of acm sigir workshop on recommender systems., 1999.

[20] Dan Cosley, Shyong K. Lam, Istvan Albert, Joseph A. Konstan, and John Riedl. Is seeing believing?: How recommender system interfaces affect users' opinions. In *Proceedings of the Conference on Human Factors in Computing Systems*, pages 585–592. ACM Press, 2003.

[21] Paul Cotter and Barry Smyth. PTV: Intelligent personalised TV guides. In *Proceedings of the 7th Conference on Artificial Intelligence (AAAI-00) and of the 12th Conference on Innovative Applications of Artificial Intelligence (IAAI-00)*, pages 957–964, Menlo Park, CA, 2000. AAAI Press.

[22] Norton Deutsch. *The resolution of conflict: Constructive and destructive processes*. Sage Publications, 1973.

[23] Gaia diLuzio. A sociological concept of client trust. *Current Sociology*, 54(4), 2006.

[24] Li Ding, Lina Zhou, Tim Finin, and Anupam Joshi. How the semantic web is being used: An analysis of foaf documents. In *HICSS '05: Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05) - Track 4*, page 113.3, Washington, DC, USA, 2005. IEEE Computer Society.

[25] eBay Incorporated. Facebook growth statistics for 2006 and 2007. Press release, San Jose, CA. Jul 2007. available at http://blog.facebook.com/blog.php?post=2245132130.

[26] eBay Incorporated. Fourth quarter and full year financial results for 2005. Press release, San Jose, CA. Jan 18 2006. available at http://investor.ebay.com/news/Q405/EBAY0118-123321.pdf.

[27] eBay Incorporated. Fourth quarter and full year financial results for 2006. Press release, San Jose, CA. Jan 18 2006. available at http://pages.ebay.nl/perskamer/persberichten/070124.html.

[28] D. Fisk. An application of social filtering to movie recommendation. In *Software Agents and Soft Computing, Lecture Notes in Computer Science*, pages 116–131, 1997.

[29] Robert H Frank. Melding sociology and economics: James coleman's foundations of social theory. *Journal of Economic Literature*, 30(1):147–70, March 1992. available at http://ideas.repec.org/a/aea/jeclit/v30y1992i1p147-70.html.

[30] Jill Freyne, Rosta Farzan, Peter Brusilovsky, Barry Smyth, and Maurice Coyle. Collecting community wisdom: integrating social search & social navigation. In *IUI '07: Proceedings of the 12th international conference on Intelligent user interfaces*, pages 52–61, New York, NY, USA, 2007. ACM Press.

[31] Thomas M. J. Fruchterman and Edward M. Reingold. Graph drawing by force-directed placement. *Software - Practice and Experience*, 21(11):1129–1164, 1991.

[32] Brian Gaines. Adapting to a highly automated world, the collective stance in modeling expertise in individuals and organizations. Technical report, University of Calgary, Canada., 1987.

[33] Diego Gambetta. *Can We Trust Trust?* Published Online, 2000.

[34] Michael Gamon, Anthony Aue, Simon Corston-Oliver, and Eric K. Ringger. Pulse: Mining customer opinions from free text. In *Advances in Intelligent Data Analysis (IDA)*, pages 121–132, 2005.

[35] Malcolm Gladwell. *The Tipping Point : How Little Things Can Make a Big Difference*. Time Warner Books Uk, January 2002.

[36] Jennifer Golbeck. Generating predictive movie recommendations from trust in social networks. In *Proceedings of the fourth international conference on trust management*, pages 27–32, 2006.

[37] Jennifer Golbeck and James Hendler. Accuracy of metrics for inferring trust and reputation in semantic web-based social networks. In *Proceedings of EKAW'04*, pages LNAI 2416, p. 278 ff., 2004.

[38] Jennifer Golbeck and James Hendler. Filmtrust: movie recommendations using trust in web-based social networks. *Consumer Communications and Networking Conference*, pages 282–286, Jan 2006.

[39] Jennifer Golbeck and James A. Hendler. Inferring binary trust relationships in web-based social networks. *ACM Trans. Internet Techn.*, 6(4):497–529, 2006.

[40] Jennifer Golbeck and Michael Wasser. Socialbrowsing: Integrating social networks and web browsing. *Conference on Human Facturs in Computing Systems (CHI 2007)*, 2007.

[41] Jennifer Ann Golbeck. *Computing and applying trust in web-based social networks*. PhD thesis, University of Maryland at College Park, College Park, MD, USA, 2005. Chair-James Hendler.

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

[43] Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, 2001.

[44] Andrew R. Golding and Dan Roth. A winnow-based approach to context-sensitive spelling correction. *Machine Learning*, 34(1-3):107–130, 1999.

[45] Nathaniel Good, J. Ben Schafer, Joseph A. Konstan, Al Borchers, Badrul Sarwar, Jon Herlocker, and John Riedl. Combining collaborative filtering with personal agents for better recommendations. In *Proceedings of the 11th Conference on Innovative Applications of Artificial Intelligence*, pages 439–446, Menlo Park, Cal., July 18th 1999. AAAI/MIT Press.

[46] Jonathan L. Herlocker, Joseph A. Konstan, and John Riedl. Explaining collaborative filtering recommendations. In *Proceedings of ACM CSCW'00 Conference on Computer-Supported Cooperative Work*, pages 241–250, 2000.

[47] Yoshinori Hijikata, Hanako Ohno, Yukitaka Kusumura, and Shogo Nishida. Social summarization of text feedback for online auctions and interactive presentation of the summary. In *IUI '06: Proceedings of the 11th international conference on Intelligent user interfaces*, pages 242–249, New York, NY, USA, 2006. ACM Press.

[48] Peter Hoschka. Cscw research at gmd-fit: from basic groupware to the social web. *SIGGROUP Bull.*, 19(2):5–9, 1998.

[49] Don House, Victoria Interrante, David H. Laidlaw, Russell Taylor, and Colin Ware. Panel: Design and evaluation in visualization research. In *Proceedings of IEEE Visualization Conference*, Minneapolis, MN, October 2005.

[50] Yukari Iseyama, Satoru Takahashi, and Kazuhiko Tsuda. A study of knowledge extraction from free text data in customer satisfaction survey. In *KES*, pages 509–515, 2004.

[51] Intelligence Technology Innovation Center (ITIC). Blackbook prototype framework for the knowledge discovery and dissemination (kdd) program. McLean, VA, USA, October 3–4 2006.

[52] A. Jameson, J. Konstan, and J. Riedl. AItechniques for personalised recommendation. In *Anthony Jameson, Joseph A. Konstan and John Riedl. Tutorial SA1, Eighteenth International Conference on Artificial Intelligence, August 10th, 2003*, 2003.

[53] Ken Jordan, Jan Hauser, and Steven Foster. Augmented social network. *http://en.wikipedia.org/wiki/*, July 2004.

[54] Andun Josang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision.

[55] Audun Josang. Prospectives for online trust management. *IEEE Transactions on Knowledge and Data Engineering*, 2007.

[56] Audun Jøsang, Elizabeth Gray, and Michael Kinateder. Analysing Topologies of Transitive Trust. In Theo Dimitrakos and Fabio Martinelli, editors,

Proceedings of the First International Workshop on Formal Aspects in Security and Trust (FAST2003), pages 9–22, Pisa, Italy, September 2003.

[57] Audun Jøsang, Elizabeth Gray, and Michael Kinateder. *Simplification and Analysis of Transitive Trust Networks. Web Intelligence and Agent Systems: An International Journal*, pages 1–1, September 2005. ISBN ISSN: 1570-1263.

[58] George Karypis. Evaluation of item-based top-n recommendation algorithms. In *Proceedings of the 2001 ACM CIKM International Conference on Information and Knowledge Management, Atlanta, Georgia, USA*, pages 247–254, 2001.

[59] Liadh Kelly. An intelligent navigational aid for the world wide web. Master's thesis, University College Dublin, Belfield, Dublin 4. Ireland, February 2002.

[60] Sherry Koshman. Web-based visualization interface testing: Similarity judgments. *J. Web Eng.*, 3(3-4):281–296, 2004.

[61] B. Krulwich. Lifestyle finder. *AI magazine*, 18:37–46, 1997.

[62] N. Kushmerick. Robustness analyses of instance-based collaborative recommendation. In H. Toivonen T. Elomaa, H. Mannila, editor, *Proceedings of the European Conference on Machine Learning, Helsinki, Finland.*, volume 2430, pages 232–244. Lecture Notes in Computer Science Springer-Verlag Heidelberg, 2002.

[63] Shyong K. Lam and John Riedl. Shilling recommender systems for fun and profit. In *Proceedings of the 13th international conference on World Wide Web*, pages 393–402. ACM Press, 2004.

[64] Cliff A.C. Lampe, Nicole Ellison, and Charles Steinfield. A familiar face(book): profile elements as signals in an online social network. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 435–444, New York, NY, USA, 2007. ACM Press.

[65] Raph Levien. *Attack Resistant Trust Metrics.* PhD thesis, University of California, Berkeley, California, 2003. ISBN 9729961506.

[66] Greg Linden, Steve Hanks, and Neal Lesh. Interactive assessment of user preference models: The automated travel assistant. In *Proceedings, User Modeling '97, Chia Laguna, Sardinia, Italy, June 2-5*, 1997.

[67] Hugo Liu, Pattie Maes, and Glorianna Davenport. Unraveling the taste fabric of social networks. *International Journal on Semantic Web and Information Systems*, 2(1):42–71, 2006.

[68] Niklas Luhmann, Howard Davis, John Raffan, and Kathryn Rooney. *Trust and Power: Two works by Niklas Luhmann.* Chichester Wiley, 1980.

[69] S. Marsh. Formalising trust as a computational concept. *Ph.D. Thesis. Department of Mathematics and Computer Science, University of Stirling, 1994.*

[70] Paolo Massa and Paolo Avesani. Trust-aware collaborative filtering for recommender systems. *Proceedings of International Conference on Cooperative Information Systems, Agia Napa, Cyprus, 25 Oct - 29 Oct 2004.*

[71] Paolo Massa and Bobby Bhattacharjee. Using trust in recommender systems: an experimental analysis. *2nd International Conference on Trust Managment, Oxford, England*, 2004.

[72] Kevin McCarthy, James Reilly, Lorraine McGinty, and Barry Smyth. Experiments in dynamic critiquing. In *IUI '05: Proceedings of the 10th international conference on Intelligent user interfaces*, pages 175–182, New York, NY, USA, 2005. ACM Press.

[73] Kevin McCarthy, James Reilly, Barry Smyth, and Lorraine McGinty. Generating diverse compound critiques. *Artif. Intell. Rev.*, 24(3-4):339–357, 2005.

[74] Sean M. Mcnee, Nishikant Kapoor, and Joseph A. Konstan. Don't look stupid: avoiding pitfalls when recommending research papers. In *CSCW '06: Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, pages 171–180, New York, NY, USA, 2006. ACM Press.

[75] P. Melville, R. Mooney, and R. Nagarajan. Content-boosted collaborative filtering. In *In Proceedings of the Eighteenth National Conference on Artificial Intelligence*, 2002.

[76] Stuart E. Middleton. Exploiting synergy between ontologies and recommender systems, semantic web workshop 2002 hawaii, usa, 2002.

[77] S. Milgram. The small world problem. *Psychology Today*, 1:61–67, May 1967.

[78] Bradley N. Miller, Istvan Albert, Shyong K. Lam, Joseph A. Konstan, and John Riedl. Movielens unplugged: experiences with an occasionally connected recommender system. In *IUI '03: Proceedings of the 8th international conference on Intelligent user interfaces*, pages 263–266, New York, NY, USA, 2003. ACM Press.

[79] Bamshad Mobasher, Robin Burke, Runa Bhaumik, and Chad Williams. Effictive attack models for shilling item-based collaborative filtering systems. In *WebKDD,*, Chicago, Illinois, USA., 2005. ACM Press.

[80] Bamshad Mobasher, Robin Burke, Runa Bhaumik, and Chad Williams. Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness. *ACM Trans. Inter. Tech.*, 7(4):23, 2007.

[81] Audris Mockus, Roy T Fielding, and James D Herbsleb. Two case studies of open source software development: Apache and mozilla. *ACM Trans. Software Engineering and Methodology*, 11(3), 2002.

[82] Miquel Montaner, Beatriz Lopez, and Josep Lluis de la Rosa. Developing trust in recommender agents. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 304–305. ACM Press, 2002.

[83] Lik Mui, Mojdeh Mohtashemi, and Ari Halberstadt. Notions of reputation in multi-agents systems: a review. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 280–287. 280–287, 2002.

[84] Paul Mutton. Inferring and visualizing social networks on Internet Relay Chat. In *Eighth International Conference on Information Visualization (IV04)*, page 9 total. IEEE, July 2004.

[85] John O'Donovan. Adrec: an adaptive recommender system. Master's thesis, National University of Ireland, 2005.

[86] John O'Donovan and John Dunnion. A framework for evaluation of collaborative recommendation algorithms in an adaptive recommender system. In *Proceedings of the International Conference on Computational Linguistics (CICLing-04), Seoul, Korea*, pages 199–203. Springer-Verlag, February 2004.

[87] John O'Donovan and Barry Smyth. Trust in recommender systems. In *Proceedings of the 10th International Conference on Intelligent User Interfaces*, pages 167–174. ACM Press, 2005.

[88] John O'Donovan and Barry Smyth. Is trust robust? an analysis of trust-based recommendation. In *IUI '06: Proceedings of the 11th International Conference on Intelligent User Interfaces*, pages 101–108. ACM Press, 2006.

[89] Michael O'Mahony, Neil Hurley, Nicholas Kushmerick, and Guenole Silvestre. Collaborative recommendation: A robustness analysis. *ACM Trans. Inter. Tech.*, 4(4):344–377, 2004.

[90] Michael P. O'Mahony, Neil Hurley, and Guenole C. M. Silvestre. An attack on collaborativeattack on collaborative filtering. In *Proceedings of the 13th International Conference on Database and Expert Systems Applications*, pages 494–503. Springer-Verlag, 2002.

[91] Michael P. O'Mahony, Neil J. Hurley, and Guenole C. M. Silvestre. Utility-based neighbourhood formation for efficient and robust collaborative filtering. In *EC '04: Proceedings of the 5th ACM conference on Electronic commerce*, pages 260–261, New York, NY, USA, 2004. ACM Press.

[92] Derry O'Sullivan, David C. Wilson, and Barry Smyth. Improving case-based recommendation: A collaborative filtering approach. In *Proceedings of the Sixth European Conference on Case Based Reasoning.*, pages LNAI 2416, p. 278 ff., 2002.

[93] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[94] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in NaturalLanguage Processing (EMNLP)*, pages 79–86, 2002.

[95] Fulvia Ferrazzi Paolo Ciccarese, Stefano Mazzocchi and Lucia Sacchi. Genius: a new tool for gene networks visualization. *Intelligent Data Analysis in Medicine and Pharmacology (IDAMAP)*, pages 107–111, 2004.

[96] DeSola Pool and Manfred Kochen. Contacts and influence. *Social Networks*, 1(1):5–51, 1978.

[97] M. F. Porter. An algorithm for suffix stripping. *Readings in information retrieval*, pages 313–316, 1997.

[98] R. Rafter, K. Bradley, and B. Smyth. Passive profiling and collaborative recommendation. In *Proceedings of the 10th Irish Conference on Artificial Intelligence and Cognitive Science, Cork, Ireland*. Artificial Intelligence Association of Ireland (AAAI Press), 1999.

[99] Adrian E. Raftery, David Madigan, and Jennifer A. Hoeting. Bayesian model averaging for linear regression models. *Journal of the American Statistical Association*, 92(437):179–191, 1997.

[100] S. Ramchurn, T. Huynh, and N. Jennings. Trust in multi-agent systems. *The Knowledge Engineering Review. Vol. 19:1, 1V25. Cambridge University Press.*, 2004.

[101] Drummond Reed, Marc Le Maitre, Bill Barnhill, Owen Davis, and Fen Labalme. The social web: Creating an open social network with xdi. *Published Online*, July 2004.

[102] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of ACM CSCW'94 Conference on Computer-Supported Cooperative Work*, pages 175–186, 1994.

[103] Paul Resnick and Richard Zeckhauser. Trust among strangers in internet transactions: Empirical analysis of ebay's reputation system. *The Economics of the Internet and E-Commerce. Volume 11 of Advances in Applied Microeconomics.*, December 2002.

[104] Paul Resnick, Richard Zeckhauser, Eric Friedman, and Ko Kuwabara. Reputation systems. *Communications of the ACM*, December 2000. Vol. 43, No. 12.

[105] Paul Resnick, Richard Zeckhauser, Eric Friedman, and Ko Kuwabara. A context-aware music recommendation system using fuzzy bayesian networks with utility theory. *Lecture Notes in Computer Science*, September 2006. Vol. 4223,2006.

[106] Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz. A bayesian approach to filtering junk E-mail. In *Learning for Text Categorization: Papers from the 1998 Workshop on Learning for Text Categorization (AAAI-98)*, Madison, Wisconsin, 1998. AAAI Technical Report WS-98-05.

[107] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John Reidl. Item-based collaborative filtering recommendation algorithms. In *World Wide Web*, pages 285–295, 2001.

[108] Badrul M. Sarwar, Joseph A. Konstan, Al Borchers, Jon Herlocker, Brad Miller, and John Riedl. Using filtering agents to improve prediction quality in the grouplens research collaborative filtering system. In *Proceedings of ACM CSCW'98 Conference on Computer-Supported Cooperative Work*, Social Filtering, Social Influences, pages 345–354. ACM Press, 1998.

[109] J. Ben Schafer, Joseph Konstan, and John Riedl. Recommender systems in E-Commerce. In *Proceedings of the ACM Conference on Electronic Commerce (EC-99)*, pages 158–166, New York, November 1999. ACM Press.

[110] Shilad Sen, Shyong K. Lam, Al M. Rashid, Dan Cosley, Dan Frankowski, Jeremy Osterhouse, Maxwell F. Harper, and John Riedl. Tagging, communities, vocabulary, evolution. In *CSCW '06: Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, pages 181–190, New York, NY, USA, 2006. ACM Press.

[111] Upendra Shardanand and Patti Maes. Social information filtering: Algorithms for automating "word of mouth". In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, volume 1 of *Papers: Using the Information of Others*, pages 210–217, 1995.

[112] Rashmi Sinha and Kirsten Swearingen. The role of transparency in recommender systems. In *CHI '02 extended abstracts on Human factors in computing systems*, pages 830–831. ACM Press, 2002.

[113] B. Smyth, D. Wilson, and D. O'Sullivan. Improving the quality of the personalised electronic programme guide. In *In Proceedings of the TV'02 the 2nd Workshop on Personalisation in Future TV, May 2002.*, pages 42–55, 2002.

[114] Barry Smyth. Adaptive information access and the quest for the personalization-privacy sweetspot. In *IUI '05: Proceedings of the 10th international conference on Intelligent user interfaces*, pages 2–2, New York, NY, USA, 2005. ACM Press.

[115] Barry Smyth and Paul McClave. Similarity vs. diversity. In *ICCBR '01: Proceedings of the 4th International Conference on Case-Based Reasoning*, pages 347–361, London, UK, 2001. Springer-Verlag.

[116] Alvin Toffler. *Future Shock*. Random House, New York, Februray 1970.

[117] LLC TouchGraph. Touchgraph available at www.touchgraph.com.

[118] Davor Ubrani and Gail C. Murphy. Hipikat: Recommending pertinent software development artifacts. In *Proceedings of the 25th International Conference on Software Engineering*, pages 408–418. IEEE Computer Society, 2003.

[119] M.G. Vosalis and K.G. Margaritis. Using svd and demographic data for the enhancement of generalized collaborative filtering. *Information Sciences*, August 2007. Vol. 177/15/3017.

[120] Li Xiong and Ling Liu. Building trust in decentralized peer-to-peer electronic communities. In *Fifth International Conference on Electronic Commerce Research (ICECR-5)*, 2002.

[121] Cai-Nicolas Ziegler and Georg Lausen. Propagation models for trust and distrust in social networks. *Information Systems Frontiers*, 7(4-5):337–358, 2005.

[122] Cai-Nicolas Ziegler, Sean M. Mcnee, Joseph A. Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, New York, NY, USA.

[123] Cai-Nicolas Ziegler and Michal Skubacz. Towards automated reputation and brand monitoring on the web. In *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intleligence*, pages 1066–1070, Hong Kong, December 2006. IEEE Computer Society Press.