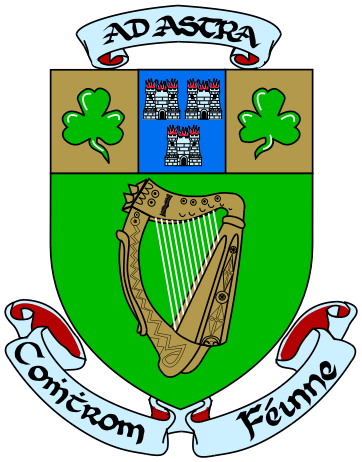
Towards Robust and Efficient Automated Collaborative Filtering

by

Michael P. O'Mahony



A Thesis submitted for the Degree of

Doctor of Philosophy of the
University College Dublin

Department of Computer Science
Information Hiding Laboratory
University College Dublin, Ireland

December 2004

Declaration

I declare that this dissertation is entirely my own work, carried out at University College Dublin, and has not been submitted for a degree to this or any other University and that the contents are original unless otherwise stated.

Signed,

Michael P. O'Mahony

22nd of December, 2004

Abstract

Recent years have seen an explosive growth in the quantity of information that is available. The need for automated techniques to deal with the information overload problem is clear. The ability of users to quickly locate items that meet their own specific needs is crucial. Recommender systems have now been widely and successfully implemented, particularly in e–commerce applications, as a solution to this problem.

One of the most popular information filtering techniques that is used to drive recommender systems is known as collaborative filtering. This collection of algorithms essentially automates the word of mouth process by using the preferences of like–minded people to make predictions and recommendations for individuals.

The performance of recommender systems has been extensively evaluated from the efficiency, scalability, coverage and accuracy perspectives. However, one important performance measure has not been considered – namely, the robustness of a system. Robustness against malicious attack is a key feature of many domains and in this work, we propose a framework within which to analyse the robustness of recommender systems. New performance measures are also introduced in order to evaluate system robustness.

We describe several attack types to which recommender systems can be subjected. The attacks that are considered in this thesis are all implemented by inserting bogus data through the normal system interface – no other access to a system's database is assumed. We show that one particular class of collaborative filtering algorithms proves very vulnerable to such attacks. In addition, we demonstrate that, even when financial costs are imposed on the insertion of ratings data, the gains for attackers far exceed the costs.

Finally, we propose techniques to defend against attack. These techniques aim to identify and exclude from neighbourhoods any biased data that is present in a system. In particular, we introduce profile utility, which defines novel approaches to neighbourhood formation, and similarity weight transformations, which are based on certain characteristics of the items that are contained in neighbours' profiles. We show that the profile utility algorithm, in addition to providing robustness against attack, results in accurate and efficient collaborative filtering. The algorithm is also suitable for large–scaled systems, since the on–line complexity of the approach is independent of system size.

To Max Eamon Kissane

Acknowledgments

There are a number of people that I would like to thank for all the help that I received in completing this thesis. First of all, I wish to express my gratitude to both my supervisors, Neil Hurley and Guénolé Silvestre, for their expertise, insight, assistance and patience during the course of this work. I am indeed grateful for everything.

I would also like to thank Nicholas Kushmerick for our collaborative work, and also for providing welcome and useful feedback. Likewise, I wish to acknowledge all others who assisted and supported this work.

I must also mention my fellow students for all the friendship, fun and distractions that were provided. To name just some, a special thanks to Barry O'Donovan, Keith Cullen and John Sheppard.

Finally, I wish to thank my sister, brother and parents for all the support and encouragement that I received throughout my studies and before.

Contents

Declaration i

Abstract ii

Dedication iii

Acknowledgments iv

Contents v

List of Figures ix

List of Tables xii

Acronyms & Abbreviations xiii

Notation xiv

Introduction 1

1 Recommender Systems 6

1.1 Introduction 6

1.2 Recommender Systems 6

1.3 Approaches to Information Filtering 9

1.3.1 Collaborative Filtering 9

1.3.2 Advantages & Limitations of the Collaborative Filtering Approach . 12

1.3.3 Alternative Approaches 14

1.4 Summary 16

2 Collaborative Recommender Algorithms 17

2.1 Introduction . . . 17

2.2 Memory–Based Systems . . . 17

2.2.1 Data Representation . . . 18

2.2.2 Similarity Metrics . . . 19

2.2.3 Neighbourhood Formation . . . 23

2.2.4 Prediction and Recommendation Generation . . . 26

2.2.5 Extensions to Memory–Based Algorithms . . . 29

2.3 Summary . . . 32

3 Performance of Recommender Systems 33

3.1 Introduction . . . 33

3.2 Performance Criteria . . . 33

3.2.1 Coverage . . . 34

3.2.2 Efficiency and Scalability . . . 35

3.2.3 Accuracy . . . 36

3.3 Summary . . . 42

4 Robustness of Recommender Systems 43

4.1 Introduction . . . 43

4.2 Robustness . . . 43

4.2.1 Malicious Attacks . . . 45

4.2.2 Related Work . . . 47

4.2.3 A Formal Definition of Robustness . . . 48

4.2.4 Robustness Metrics . . . 49

4.3 Summary . . . 51

5 Attack Strategies **52**

5.1 Introduction . . . 52

5.2 Experimental Datasets . . . 52

5.3 Attacks . . . 53

5.4 Developing an Attack Strategy . . . 55

5.4.1 Random Push Attack . . . 56

5.4.2 Focused Push Attack . . . 57

5.4.3 Large Attack Profiles . . . 60

5.4.4 Effect of Item Characteristics on Robustness . . . 67

5.4.5 Non–Optimal Attacks . . . 71

5.5 Effect of Algorithm Extensions . . . 77

5.5.1 Significance Weighting . . . 77

5.5.2 Case Amplification . . . 78

5.5.3 Inverse User Frequency . . . 79

5.5.4 Entropy . . . 80

5.6 Product Nuke Attacks . . . 81

5.7 Alternative Attack Strategies . . . 83

5.8 Neighbourhood Formation Schemes . . . 85

5.8.1 Coverage . . . 85

5.8.2 Accuracy . . . 86

5.8.3 Robustness . . . 87

5.9 Cost–Benefit Analysis . . . 89

5.10 Summary . . . 97

6 Preventing Attack **98**

6.1 Introduction . . . 98

6.2 Neighbourhood Filtering . . . 99

6.2.1 Robust Collaborative Filtering . . . 100

6.2.2 Robustness Against Informed Attacks . . . 104

6.3 Intelligent Neighbourhood Formation 108
6.3.1 Profile Utility 109
6.3.2 Similarity Weight Transformation 112
6.4 Summary 120

Conclusion **121**

Bibliography **128**

A Derivation of Inverse User Frequency **137**

B k–Nearest Neighbour Neighbourhood Size Experiments **140**

C Combined Nearest Neighbour and Thresholding Neighbourhood Size Experiments **143**

D Macnaughton–Smith *et al.* Clustering Algorithm **146**

List of Figures

2.1 The main components of memory–based systems 18

2.2 The effect of rating scales on Cosine similarity 20

4.1 An example of a product nuke attack 46

5.1 MovieLens: histogram of the deviations from user means 63

5.2 MovieLens: histogram of pre– and post–attack predictions 64

5.3 Pearson correlation: normalised MAPE vs. attack strength 65

5.4 Pearson correlation: percentage of good predictions vs. attack strength . . 65

5.5 Cosine similarity: normalised MAPE vs. attack strength 66

5.6 Cosine similarity: percentage of good predictions vs. attack strength 66

5.7 MovieLens: MAPE vs. item popularity 67

5.8 EachMovie: MAPE vs. item popularity 68

5.9 Smart Radio: MAPE vs. item popularity 68

5.10 EachMovie: MAPE vs. item likability 70

5.11 Smart Radio: MAPE vs. item entropy 70

5.12 EachMovie: normalised MAPE vs. item popularity knowledge 72

5.13 EachMovie: percentage of good predictions vs. item popularity knowledge 73

5.14 EachMovie: predictive accuracy vs. item popularity knowledge 74

5.15 EachMovie: normalised MAPE vs. item likability knowledge 75

5.16 EachMovie: percentage of good predictions vs. item likability knowledge . 76

5.17 EachMovie: percentage of positive prediction shifts vs. item likability knowledge 76

5.18 EachMovie: predictive accuracy vs. item likability knowledge 77

5.19 EachMovie: coverage vs. neighbourhood formation schemes 86

5.20 EachMovie: accuracy vs. neighbourhood formation schemes 87

5.21 EachMovie: robustness vs. neighbourhood formation schemes 88

5.22 EachMovie: maximum ROI vs. attack cost 92

5.23 EachMovie: attack profit vs. dataset size 94

5.24 EachMovie: minimum system size required to provide a zero profit margin 95

5.25 EachMovie: projected profit vs. system size 95

5.26 EachMovie: percentage of good predictions vs. attack cost 96

6.1 Macnaughton–Smith clustering 102

6.2 MovieLens: performance of neighbourhood filtering algorithm 103

6.3 MovieLens: genuine and attack cluster sizes 104

6.4 Ratings distribution for informed product nuke attack 106

6.5 MovieLens: performance of neighbourhood filtering algorithm (informed attack) 107

6.6 MovieLens: genuine and attack cluster sizes (informed attack) 107

6.7 Different approaches to neighbourhood formation 109

6.8 The definition of inverse item popularity as a function of the popularity threshold 110

6.9 EachMovie: accuracy of intelligent neighbour selection algorithm 112

6.10 EachMovie: accuracy of intelligent neighbour selection algorithm with similarity weight transformation scheme 114

6.11 MovieLens: accuracy of intelligent neighbour selection algorithm with similarity weight transformation scheme 115

6.12 Smart Radio: accuracy of intelligent neighbour selection algorithm with similarity weight transformation scheme 115

6.13 EachMovie: coverage provided by the intelligent neighbour selection algorithm with similarity weight transformation scheme 116

6.14 MovieLens: coverage provided by the intelligent neighbour selection algorithm with similarity weight transformation scheme 116

6.15 Smart Radio: coverage provided by the intelligent neighbour selection algorithm with similarity weight transformation scheme 117

B.1 k–Nearest Neighbour: accuracy vs. neighbourhood size for the MovieLens dataset 141

B.2 k–Nearest Neighbour: accuracy vs. neighbourhood size for the EachMovie dataset 141

B.3 k–Nearest Neighbour: accuracy vs. neighbourhood size for the Smart Radio dataset 142

C.1 Combined Nearest Neighbour and Thresholding: accuracy vs. neighbourhood size for the MovieLens dataset 144

C.2 Combined Nearest Neighbour and Thresholding: accuracy vs. neighbourhood size for the EachMovie dataset 144

C.3 Combined Nearest Neighbour and Thresholding: accuracy vs. neighbourhood size for the Smart Radio dataset 145

List of Tables

1.1 Example of a simplified recommender system database 10

5.1 Details of experimental datasets 53

5.2 MovieLens: random push attack results 57

5.3 MovieLens: focused push attack results 59

5.4 Results for focused push attacks, using large attack profiles, carried out on all datasets 62

5.5 The effects of algorithm extensions on system robustness 79

5.6 Results for focused nuke attacks, using large attack profiles, carried out on all datasets 82

5.7 Results for an AverageBot push attack carried out on all datasets 84

5.8 Mean and standard deviation of dataset ratings 85

5.9 EachMovie: cost–benefit analysis for a product push attack 93

6.1 Comparison between the reputation system algorithm of Dellarocas (2000) and the derived collaborative filtering algorithm 101

6.2 MovieLens: results for focused nuke attacks, using large attack profiles . . 105

Acronyms & Abbreviations

ACF	Automated Collaborative Filtering
APS	Attack Profile Size
CBR	Case–Based Reasoning
CD	Compact Disk
CF	Collaborative Filtering
k–NN	k–Nearest Neighbour
KBS	Knowledge–Based Systems
LSI	Latent Semantic Indexing
MAE	Mean Absolute Error
MAPE	Mean Absolute Prediction Error
MSD	Mean Squared Difference
NMAE	Normalised Mean Absolute Error
NMAPE	Normalised Mean Absolute Prediction Error
PTV	Personalised Television
PU	Profile Utility
ROC	Receiver Operating Characteristic
ROI	Return on Investment
SVD	Singular Value Decomposition
TRC	Total Rating Comparisons

Notation

κ_i	Browser–to–buyer conversion rate for item i
ρ	Case amplification parameter
σ_a	Standard deviation of ratings assigned by user a
A_j	Set of users over which an attack on item j is evaluated
b	Buyer b
c_i	Purchase cost of item i
$D(a, i)$	Distance between users a and i
E	E metric
F_1	F_1 metric
g_i	Pre–attack probability that a good prediction is made for item i
g'_i	Post–attack probability that a good prediction is made for item i
$H(j)$	Entropy of item j
$H(j)_{max}$	Maximum entropy of item j
I_a	Set of items rated by user a
$\text{InvPop}(j)$	Inverse popularity of item j
k	Neighbourhood size
l	Number of items contained in a user profile
L	Similarity weight threshold
m	Item popularity threshold
$\text{MAE}_{a,j}$	Mean absolute error for a prediction made for user a on item j
MAPE_j	Mean absolute prediction error for all predictions made on item j
n	Number of common items between two users
n_d	Total number of users in a system
n_j	Number of ratings received by item j
$n_{i,a}$	Number of bogus ratings that are inserted into a system in the course of an attack on item i
N	Significance weighting parameter
$\text{NMAE}_{a,j}$	Normalised mean absolute error for a prediction made for user a on item j
NMAPE_j	Normalised mean absolute prediction error for all predictions made on item j
$p_{a,j}$	Pre–attack predicted rating for user a, item j
$p'_{a,j}$	Post–attack predicted rating for user a, item j

$r_{a,j}$	Rating assigned by user a to item j
r_{max}	Maximum rating
r_{min}	Minimum rating
r_n	Neutral or middle rating
$\bar{r}_a$	Mean rating of user a
r	Pearson's correlation coefficient
R	Recommended set of items
R_a	Expected utility of a ranked list of items for user a
R_a^{max}	Maximum achievable utility of a ranked list of items for user a
s	Seller s
T	Test set of items
$\mathbf{u}$	A user vector in 2–dimensional item space
$\mathcal{U}$	Universe of all users
$U(a)$	Profile utility of user a
$\mathbf{v}$	A user vector in 2–dimensional item space
$w(a,i)$	Similarity weight between users a and i
$w'(a,i)$	Transformed similarity weight between users a and i

Introduction

Rationale

In recent years, an explosive increase in the quantity of information that is available on all domains has occurred. In particular, the World Wide Web has seen exponential growth, and is used by many as a primary source of information. While ready access to vast stores of information is to be welcomed, problems have arisen for people seeking to locate relevant and high quality sources of information from the sheer volume that exists.

The need for automated techniques to deal with the information overload problem has become very clear. Many businesses have now moved on–line in an effort to decrease costs, increase market share and improve profit. For the business community, the ability of customers to quickly find products that meet their own specific requirements is crucial. In recent years, recommender systems (Resnick and Varian, 1997; Schafer *et al.*, 1999; Sarwar *et al.*, 2002) have emerged and have been implemented successfully in many e–commerce applications.

One of the most popular information filtering techniques that is used to drive recommender systems is known as *collaborative* or *social filtering* (Resnick *et al.*, 1994; Shardanand and Maes, 1995; Hill *et al.*, 1995). Frequently, people are required to make choices without having sufficient personal knowledge of the alternatives. For example, we rely on the recommendations from others concerning such matters as which movie to see, which book to read or which car to buy. Collaborative filtering algorithms automate this process by using the opinions of a community of users to assist members of the community to more effectively identify useful goods and services. These algorithms have the advantage of being able to make use of criteria such as human taste and judgment when making recommendations, which cannot be easily leveraged by other, keyword–based, information filtering approaches.

The performance of recommender systems is, of course, a key consideration. Performance has been extensively evaluated from the point–of–view of *efficiency*, *scalability*, *coverage* and *accuracy* (Breese *et al.*, 1998; Herlocker *et al.*, 1999; Sarwar *et al.*, 2000b). Efficiency is concerned with the time taken for a system to return a prediction to a user. Issues with efficiency can occur as the system scales in size. The addition of new users, items and transactions to a system results in the recommender system having to perform more calculations in order to produce predictions. Coverage refers to the percentage of predictions

sought that a system is able to deliver. Accuracy relates to the quality of the predictions and recommendations that are made. Obviously, system accuracy is of major importance and has been a prime consideration in research to date.

On a different note, the privacy risks that apply to recommender system users have also been researched (Ramakrishnan *et al.*, 2001; Canny, 2002; Oard *et al.*, 2003). For example, Ramakrishnan *et al.* (2001) identify *straddlers* as a potential risk, in that they may enable users to uncover the identities and personal details of others. A straddler is a user who has eclectic tastes and rates items from different sub–domains of a system. A positive benefit of straddlers is that they enable serendipitous recommendations to be made – i.e. recommendations that are unexpected but nevertheless desirable. However, information about their existence, used in conjunction with, for example, statistical database queries carried out on a system, can be exploited to uncover particular information about other users that are contained in a system.

Notwithstanding all of the work carried out to date, one important performance measure has, however, been omitted – that is the *robustness* of recommender systems. Robustness is concerned with a system's ability to provide consistent and quality predictions when some degree of noise has been introduced into the system data. In the normal course of events, inaccuracies in the data should be expected. For example, in systems where users are required to explicitly express a preference on items, it is natural that some degree of unbiased noise will occur through human error.

Of greater concern and importance is that malicious users may be motivated to deliberately *attack* recommender systems by inserting biased data into the systems. Potentially, there are significant advantages for those inclined to carry out an attack. Consider, for example, an author who wishes to promote the sales of his or her latest book. By forcing a recommender system to consistently give good predicted ratings for the book, there may well follow a significant increase in sales' returns. Since most recommender systems operate in a web environment, it is impossible to check the honesty of those who access a system, and therefore it is feasible that such attacks could occur. Thus, it is necessary to understand how much tolerance a system has to any biased noise that is present in the data.

There are several examples of malicious behaviour that are related to recommender systems. One such example relates to Amazon.com's recommender system, which linked a spiritual guide authored by a well–known televangelist with certain literature of an adult nature. An investigation conducted by Amazon.com concluded that the link was *not* the result of legitimate customers' shopping habits. A security consultant, who commented on the matter, stated "[this] prank makes me also wonder how much the Amazon recommendation system is being hacked by authors and publishers as a new marketing tool"[1].

[1]Taken from `http://news.com.com/2100-1023-976435.html`.

A second example concerns the on–line auction house eBay.com, which uses a recommender system as a reputation reporting system. Reputation reporting systems collect, distribute and aggregate feedback about participants' past behaviour (Resnick *et al.*, 2000). The system used by eBay.com, called the Feedback Forum, allows buyers and sellers to rate each other with scores of 1, 0 or -1. A text–based feedback facility also exists. In a study conducted by Resnick and Zeckhauser (2002), it was found that the relative frequency of negative feedback was highly suspicious, where only 0.6% of those that provided feedback gave negative ratings to counterparts. In addition, there was very strong evidence of retaliation and reciprocation in the provision of feedback.

The above examples underline the importance of security in recommender systems. In this work, we propose a definition of robustness for recommender systems. Using this definition, the effects of deliberate attacks on system performance are investigated. Two attack types are considered – these are *product push* and *product nuke* attacks. The objectives of these attacks are to promote and demote, respectively, the predictions that are made for selected items in a system. If the attacks are successful, items that are pushed will receive uniformly high predicted ratings, while nuked items will receive low predicted ratings.

The strategies that are developed for the above attacks prove to be successful in substantially altering the predictions that are made by collaborative recommender systems. Thus, the need for robust collaborative filtering algorithms is clear and, in this work, two such algorithms are proposed. The first algorithm is based on related work from the field of reputation reporting systems. It operates by identifying and excluding from the prediction process any biased data that may be present in a system. The second algorithm employs novel approaches to evaluating the usefulness of potential neighbours and also makes use of new similarity weight transformation schemes. This algorithm substantially increases the cost of mounting successful attacks, and thereby removes the motivation for those who may be inclined to undertake such practice.

Contributions

The following list provides a summary of the main contributions that are contained in this work:

- We introduce and define the concept of robustness for recommender systems. We show that the existing metrics are not suitable for evaluating system robustness. We propose new metrics which are independent of other performance criteria and thus, these metrics facilitate a direct comparison of the robustness of different recommender systems.

- We describe different attack types and strategies that can be mounted against recommender systems. All attacks are implemented by inserting bogus item ratings through the normal user interface. We show that these attacks are capable of significantly biasing the predictions that are made by a widely used class of collaborative filtering algorithms. We demonstrate that, even when financial costs are imposed on rating entry, the gains for attackers far exceed the costs.

- To protect recommender systems against attack, techniques are developed to identify and exclude from neighbourhoods any biased data that is present in a system. In particular, we introduce the concept of *profile utility*, which defines novel approaches to neighbourhood formation based on certain characteristics of the items that are contained in neighbours' profiles. In addition, new similarity weight transformations are proposed that serve to enhance the robustness of the algorithm.

- We show that the profile utility algorithm, in addition to providing robustness against attack, results in accurate and efficient collaborative filtering. The algorithm is also suitable for large–scaled systems, because the on–line complexity of the approach is independent of system size.

- The robustness of recommender systems against malicious attack has thus far received little attention from the research community. In other domains, malicious attacks are anticipated and systems are designed accordingly. However, this view has not prevailed with respect to recommender systems, which seems surprising and naïve given the clear potential that exists for attacks. In this work, we demonstrate that robustness is indeed an issue for recommender systems, and we propose that robustness becomes an inherent evaluation metric for the design of recommender systems and, indeed, for information retrieval systems in general.

Thesis Outline

The remainder of the thesis is organised in the following manner:

- **Chapter 1** presents an overview of recommender systems and the information filtering algorithms that drive these systems. In particular, the collaborative filtering approach is described, which has been successfully implemented in many real–world systems. Some of the alternative information filtering techniques, such as content– and demographic–based methods, etc. are also described.

- **Chapter 2** provides a detailed description of memory–based collaborative filtering algorithms that have been implemented and studied in many settings. In particular,

a review of several neighbourhood formation schemes, similarity metrics and prediction and recommendation generation techniques is presented. These algorithms are used in later chapters to evaluate the robustness of collaborative recommender systems which are subjected to attack.

- **Chapter 3** reviews the existing performance measures that are used to evaluate recommender systems. Generally, recommender are evaluated along the dimensions of prediction and recommendation accuracy, efficiency, scalability and coverage. Each of these criteria are discussed in detail along with the metrics that are used to quantify system performance.

- **Chapter 4** introduces the concept of system robustness as an additional performance measure for recommender systems (O'Mahony *et al.*, 2004a). The need for such a measure is established, and following a review of related work in the field of knowledge–based systems, a definition of robustness for recommender systems is proposed. New metrics are also developed in order to evaluate system robustness.

- **Chapter 5** examines several attack types and strategies and the effect of each on system robustness is evaluated (O'Mahony *et al.*, 2002a, b). The relative strengths and weaknesses of the algorithmic framework as outlined in Chapter 2 are examined. The chapter concludes with a cost–benefit analysis that is performed for a particular attack type.

- **Chapter 6** develops schemes to defend against the attack strategies that are introduced in Chapter 5. Two different approaches are proposed, which seek to filter attack profiles from the prediction process and thereby provide robustness against attack (O'Mahony *et al.*, 2003a, 2004d).

- Finally, in the **Conclusion**, the contributions and the findings of this thesis are summarised and an outline for future work is presented.

Chapter 1

Recommender Systems

1.1 Introduction

In this chapter, an overview of recommender systems is presented. These systems are designed with the objective of overcoming the information overload problem that is prevalent in many domains. The information filtering algorithms that drive recommender systems are also discussed. In particular, the collaborative filtering approach is described, which has been implemented in many real–world systems and which has proved very successful. In addition, some of the alternative information filtering techniques, such as content– and demographic–based methods, etc. are described.

1.2 Recommender Systems

The enormous potential of the Internet has been well recognised by the business community and the growth and popularity of e–commerce applications is set to continue. Today, all manner of goods and services are available on the world–wide–web, ranging from on–line banking, virtual travel agencies, music finders etc. Business leaders and Internet developers have also been aware of the new challenges that Internet technology brings with it, and indeed, of the solutions offered to old problems. Given the vast and ever–increasing size of the web, where competitors are just a mouse–click away, customer loyalty is essential, and new concepts have emerged to attract and retain customers to companies' web sites. Personalisation techniques, which enable web sites to adapt to customers needs, and recommender systems, that automate such personalisation of the web, have been developed and are now common place in e–commerce applications.

The great benefit of recommender systems is that they enable customers to identify relevant products from the potentially huge range of goods and services available. From a business perspective, this increases the likelihood of sales. Recommender systems can assist customers in various ways to help them find products that best meet their need. Products can be recommended based on the demographics of the customer, based on the

current 'best sellers' on a particular site, or by using the previous purchase transactions of a customer as an indication of future needs. The result is a win–win situation for both customers and on–line retailers: customers are happy that they have found, with relatively little difficulty, relevant products and retailers are, of course, benefiting from increased sales.

According to Schafer *et al.* (1999), recommender systems provide the following advantages for e–commerce applications:

Turning Web Browsers into Buyers. Due to the sheer volume of information on the web, people frequently spend only a small amount of time browsing individual web sites, and often leave without making any purchases. By ensuring good web site design and by helping customers to find suitable products, businesses can attract the attention of customers and can thereby increase sales and profits.

Cross–Selling. Recommender systems can promote sales by suggesting further products to customers that are likely to appeal to them. Provided that high quality recommendations are made, additional purchases by customers may well follow. For example, during the check–out process, items that are similar to those already selected by the customer may be recommended.

Customer Loyalty. Attracting and retaining customers is an essential business strategy. As more and more companies are moving their businesses on–line, competition increases and consumers are presented with a greater variety of choices. Recommender systems can help in securing customer loyalty if customers are satisfied with the recommendations that are being made to them. In addition, as a recommender system learns more about a customer's preferences, the ability of the system to make better recommendations improves. Thus, a relationship builds up over time between customers and specific sites, where the customer has invested time and effort in informing a recommender system about his or her own preferences, and in turn, the system reciprocates by continuously improving the quality of recommendations. Following such investments, individual customers become less likely to switch allegiances to competitors, even if other companies were to provide a similar service.

Recommender systems are now acknowledged as a key tool in promoting sales and therefore these systems are in widespread use today. For example, Amazon.com, one of the best–known on–line retailers, uses a recommender system to help customers choose titles that they may be interested in. One of the recommendation strategies used is *customers who bought this book, also liked these books*... The system also recommends authors whose books were liked by the other customers of a selected author. Recommendation systems have been commonly used in music and movie finder services. The MovieLens system[1], for example, allows users to rate movies that they have seen on a scale of 1 to 5. The system

[1] `http://movielens.umn.edu/`

can then be used to generate personalised recommendations, based on the preferences previously expressed by users.

Recommender systems have also revolutionised unsolicited advertising campaigns. Most people are familiar with large quantities of unsolicited advertisements that arrive by post or email. Frequently, poor sales returns result from such campaigns, because potential customers have to trawl through long lists of, in many cases, irrelevant items. Therefore, many people simply discard such advertisements without ever reading them. Now, recommender systems are being used to tailor advertising campaigns to suit individual peoples' needs, and thereby help to ensure a greater return on investment. For example, when CD-Now.com, an on–line vendor of compact disks, used a recommender system to automate an email marketing system, sales per solicitation increased by a factor of 100% (Ungar and Foster, 1998).

Recommender systems typically collect user preferences in either one of two ways. Preferences may either be explicitly entered by users into a system (explicit rating) or inferred from user activity (implicit rating). In the case of explicit rating, users are requested to express a preference on items, typically in the form of a numerical value on a discrete rating scale. For example, the GroupLens system (Resnick *et al.*, 1994) uses a 5–point rating scale of 1 to 5 to rate Netnews articles (where low ratings indicate a negative preference), and the MovieLens system uses a similar rating scale for users to rate motion pictures. Implicit rating refers to deriving user preference from user behaviour. For example, web browsing patterns, purchase records and various other kinds of user activity can be utilised to infer user preference (Herlocker *et al.*, 1999). Typically, implicit rating takes the form of a binary vote, where, for example, a rating of 1 means a particular product has been purchased or a web page has been visited, and a rating of 0 indicates an unvisited web page, or an item that has not been purchased.

Generally, recommender systems assist users in two basic ways. Firstly, users can request a prediction for a specific product, or group of products (prediction generation). Consider, for example, the case of a movie recommendation system, in which a user is interested in seeing a particular movie. The user can interrogate the system to obtain a prediction for the movie in question, and can then make an informed decision on whether or not to go and see it. Secondly, recommender systems can be used to output ranked lists of items that are likely to be of interest to users (recommendation generation). Continuing with the movie–based example, consider a scenario where a user wants to see a movie, but is unsure about which one to see. In this case, the system can generate a ranked list of, say, the top–10 movies that might interest the user, based on the previous movies that the user has seen and liked.

While recommender systems are undoubtedly useful, they are also costly to maintain, and consideration needs to be given to strategies that might be used to recuperate those

costs (Resnick and Varian, 1997). One such strategy is to impose a charge on recommendations, either through a pay–per–use scheme or a subscription policy. Another strategy is to charge the owners of the items being evaluated, i.e. charging record companies for 'official' reviews of their records. A third strategy is to recoup costs through advertising. Since recommender systems can also be used to target advertisements at specific consumer groups, this represents a further attraction to advertisers, along with, of course, marketing access to the user database itself.

Given the success of recommender systems in e–commerce applications, much research has been conducted into understanding and improving the technology involved. Recommender systems have been implemented using a variety of techniques, with new ideas continuously being explored. In the next section, a review of some of the most widely used implementation techniques is provided.

1.3 Approaches to Information Filtering

In this section, some of the information filtering approaches that have used to drive recommender systems are described. To begin, the approach that is the object of study in this work, that is the *collaborative filtering* approach, is described. Alternative approaches to information filtering, such as *content–based filtering*, *demographic filtering* and *case–based reasoning*, are described in subsequent sections.

1.3.1 Collaborative Filtering

This particular approach is one of the most popular and successful filtering techniques that has been used to date. Collaborative filtering (Goldberg *et al.*, 1992; Resnick *et al.*, 1994; Shardanand and Maes, 1995; Terveen *et al.*, 1997; Herlocker *et al.*, 1999; Goldberg *et al.*, 2001) helps users to make choices based on the opinions of other users in a system. The basic heuristic employed is that users who agreed or disagreed on items in the past are likely to agree or disagree on future items. To make predictions for a particular user, collaborative filtering algorithms typically find the most similarly–minded users in a system, and weight and combine the preferences of those users. This technique of information processing is commonly used by people in every–day life. People frequently seek advice from others, and, of course, place most faith in the advice of those they know and trust. For example, if a person is unsure about purchasing a particular music CD, he or she might ask some friends for their opinions. The opinions of close friends will be treated highly, while those of mere acquaintances less so, and weighted accordingly. Based on all available input, the person can then decide, with a measure of confidence, whether or not to purchase the CD. Thus, the collaborative filtering approach can be said to automate the word–of–mouth process.

Table 1.1: Example of a simplified database for a movie domain recommender system. Ratings are based on a scale of 1 to 5 with a high rating indicating a strong preference for a movie. The symbol ⊥ indicates unobserved ratings.

	The Quiet Man	Casino	Star Wars	Top Gun	Dallas: The Movie
Eamon	3	⊥	3	4	2
Sharon	4	1	4	2	4
John	5	2	⊥	2	5
Trisha	2	5	3	⊥	1
Mike	**?**	2	4	1	5

Collaborative filtering algorithms are now widely used in Internet applications, and have achieved considerable success. For example, Amazon.com and CDNow.com, the largest on–line book and music stores respectively on the web, use collaborative filtering to provide personalised information filtering for users. Many other recommender systems have been developed using this technology, such as MovieFinder.com, Belcore Video Recommender (Hill *et al.*, 1995) (movie recommendation sites), Levis Style Finder (www.levis.com, a clothing recommender system), etc.

1.3.1.1 Memory–Based & Model–Based Systems

The collaborative filtering algorithms that drive many recommender systems can be divided into two general classes, which are commonly referred to as *memory–based* and *model–based* algorithms (Breese *et al.*, 1998). Memory–based algorithms are the more prevalent of the two classes, and utilise all available data from a system database to compute predictions and recommendations. The system database contains sets of user preferences, recording the transactions that are made by all users of the system. Table 1.1 shows a simplified system database, where the preferences of users on several movies are recorded. The system database is often referred to as the *user–item matrix*. Let us assume that the objective is to predict a rating for user Mike for the movie 'The Quiet Man'. The algorithm seeks to discover other users with similar tastes to Mike in the system, and these *neighbours* are subsequently used to calculate a prediction for Mike. Various metrics have been proposed to compute the similarities between users – for example, Pearson correlation, Spearman rank correlation and Cosine similarity. In this example, it can be seen that users Sharon and John are in general agreement with Mike when the ratings assigned for co–rated movies are examined, whereas users Eamon and Trisha do not appear to share similar interests. Since both Sharon and John liked the movie 'The Quiet Man', the algorithm predicts a high rating for this particular movie for Mike. This approach is an example of a *user–user* collaborative filtering algorithm, and such algorithms have been successfully implemented in many applications.

Model–based collaborative filtering algorithms operate in a different fashion to those described above. In the first instance, a model is derived from the system data, and this model is subsequently used in the recommendation process. There are many kinds of model–based algorithms described in the literature, which include cluster models, item–item algorithms and probabilistic models, etc. (Breese *et al.*, 1998; O'Connor and Herlocker, 1999; Karypis, 2001; Sarwar *et al.*, 2002; Miller, 2003; Deshpande and Karypis, 2004). The clustering approach, as described by Sarwar *et al.* (2002), for example, first attempts to partition the dataset into groups of users. The clustering technique that is employed is the bisecting K–means algorithm (Steinbach *et al.*, 2000), a variant of the K–means clustering algorithm. Each group contains users who are identified to have similar preferences for the sets of items that each has rated. A prediction is computed for a particular user by taking a weighted average of the preferences of all other users contained in the same cluster. While the predictive accuracy provided by this approach is slightly worse when compared to considering all users as potential neighbours, the clustering approach offers significantly improved performance in terms of the number of predictions that are generated per unit time.

Item–item collaborative filtering algorithms have been proposed as a technique to deal with efficiency and scalability issues (Sarwar *et al.*, 2001; Karypis, 2001; Miller, 2003; Deshpande and Karypis, 2004). In these algorithms, analysis of transaction records is performed to find relationships between items. The goal is to identify sets of items that may imply the presence of other sets of items. Predictions and recommendations can thus be efficiently made by applying these pre–defined relationships to items that have already been rated by users. As an example of such techniques, consider the algorithm that is proposed by Miller (2003). The similarities between all pairs of items are first calculated using vector (Cosine) similarity (see Section 2.2.2), where the vector for an item j is given by the j^{th} column of the user–item matrix. A prediction can then be made for an item by taking a weighted average of the ratings for the user in question, where the pre–computed item similarities form the weights. The analysis presented by Deshpande and Karypis (2004) indicates that item–item algorithms can be up to two orders of magnitude faster than traditional user–user algorithms while offering comparable or better accuracy in terms of system accuracy.

The different strategies employed in memory–based and model–based collaborative filtering algorithms give rise to certain performance issues relating to each class. For example, memory–based approaches can suffer from serious scalability problems. The computational complexity associated with these approaches increases as additional users are registered on a system. Since it is not unknown for many systems to have millions of registered users, the ability of these algorithms to operate efficiently on an ever–increasing volume of data can be poor. In contrast, model–based algorithms are generally small, efficient and can achieve comparable accuracy to memory–based methods (Breese *et al.*, 1998).

Model–based systems need, of course, time to build a model, but this process can be done off–line at suitable times. Therefore model–based systems offer the advantage of fast and efficient recommendation generation, and are suitable for applications where any updates made to the system database, relative to the time taken to re–build the model, are small. However, when applications call for the rapid and frequent assimilation of new data, memory–based approaches are more suitable, as such approaches operate on all available data present in a system, at any given point in time, when generating predictions and recommendations.

1.3.2 Advantages & Limitations of the Collaborative Filtering Approach

In this section, the primary advantages and limitations that apply to the collaborative filtering approach are discussed. To begin, the main benefits of the approach that have been identified by Breese *et al.* (1998), Herlocker *et al.* (1999) and Rafter *et al.* (2000) are the following:

- **Quality & taste**. Human beings are have the ability to analyse information based on such diverse criteria as quality or taste. Since collaborative filtering operates on the preferences that users have assigned to items, it also has the capability to provide high quality recommendations and predictions that meet the specific needs of individual users. Other filtering techniques are less able to quantify the quality that is inherent in certain items. Consider, for example, the set of documents that are returned by a content–based search. While all the documents returned may be relevant, some may not be actually liked by users in the system for reasons of insufficient references or poor organisation, etc. Since collaborative filters consider item ratings when making recommendations, such problems can be avoided.

- **Serendipitous recommendations**. Collaborative filtering systems have the ability to provide serendipitous recommendations – i.e. where recommended items include content that is not anticipated by a user but which are, nonetheless, of interest to the user. It has been noted by Herlocker *et al.* (1999), for example, that such recommendations frequently occur in the movie domain, where collaborative filters recommend movies to users that are enjoyed, but which would never have been considered had the system not made the recommendations.

- **Item features**. Collaborative filtering makes predictions and recommendations to users based only on what similar users in a system have previously liked. Thus, in contrast to other filtering approaches, collaborative filtering does not rely on any item features that need to be pre–determined before filtering can take place. As

a result, collaborative filtering algorithms can be readily implemented in complex domains which contain items that are difficult to analyse by automated processes, such as people, ideas, emotions, movies, etc.

While the advantages of collaborative filtering algorithms as outlined above are considerable and facilitate a significant improvement in performance for recommender systems, a number of limitations also apply. These limitations, as outlined by Condliff *et al.* (1999), Claypool *et al.* (1999) and Melville *et al.* (2002) are as follows:

- **Cold–start problem**. Collaborative recommender algorithms rely on relationships between users and items to generate predictions and recommendations. In recommender systems that have been recently commissioned, however, a sufficient number of such relationships may not yet exist or may be based only on very limited quantities of data. Thus, in these circumstances, pure collaborative filtering algorithms are generally unlikely to be able to provide high–quality predictions, and predictions may be unavailable for certain users and items until more ratings data is inserted into the system database.

- **Early rater problem**. This issue is related to the above problem, except that it applies to predictions that are sought for items that are new or only recently added to a system. For such items, predictions are problematic, since there are no (or few) ratings contained in the system with which to calculate predictions. Similarly, newly registered users are problematic for even established systems, because they are unable to form any relationships with other users until at least some items have been rated. Some systems, such as the joke recommender system described by Gupta *et al.* (1999), require all users to rate a defined subset of items upon registration in an effort to solve both the early–rater and cold–start problems.

- **Sparsity problem**. Typically, recommender system datasets are very sparse, because users will have assigned ratings to only a relatively small number of items. This is particularly true for e–commerce systems, where very large numbers of items are frequently offered for sale and where most users will never purchase more than a mere fraction of the available products (consider, for example, the on–line store Amazon.com, which contains literally millions of items for sale). Thus, it can be difficult to find items that have received a sufficient number of ratings on which to base collaborative filtering predictions.

- **Grey sheep**. In all systems, there is likely to exist at least some users whose preferences do not consistently agree or disagree with any subgroup of people. In consequence, the predictions and recommendations that are made by collaborative filtering algorithms for such users cannot be expected to have a high degree of accuracy.

1.3.3 Alternative Approaches

Various other approaches to information filtering have also been proposed in the literature. In many cases, one of more of these approaches have been used in conjunction with collaborative filters to form hybrid systems (Pazzani, 1999; Claypool *et al.*, 1999; Burke, 1999; Clerkin *et al.*, 2003), which can overcome the limitations that are associated with pure collaborative recommenders. Some alternative approaches to information filtering are described in the following sections.

1.3.3.1 Content–Based Filtering

In content–based systems, items are recommended to users that are similar to items that have been liked in the past (Balabanovic, 1997; Condliff *et al.*, 1999; Popescul *et al.*, 2001). Comparisons between items are calculated over the features that are associated with each item. For example, in a document retrieval domain, features might be comprised by each distinct word in a document and the corresponding frequency of occurrence data. Based on the range of features particular to a user's past choices, further documents with similar features are recommended. Generally, content–based systems are implemented using traditional information retrieval or machine learning methodology.

One example of this class of systems is *case–based reasoning* (CBR) (Watson, 1997; O'Sullivan *et al.*, 2002). CBR systems operate by matching a user query to similar cases that are present in the system. While feature identification and matching functions are key components of all content–based filters, CBR systems typically employ a more refined and structured approach to this process. Briefly, the four main processes of CBR are: *retrieval*, *reuse*, *revision* and *retention*. In the retrieval process, the current problem is matched with cases from the case library. Retrieved cases are then reused to solve the problem in question. The proposed solution is subsequently revised and adapted if required, and finally, the new solution is retained as part of a new case.

An advantage of content–based systems is that early predictions for users and items can be facilitated. A potential drawback, however, is that feature identification can be problematic in some domains. In addition, unlike humans, content–based filters have difficulty in distinguishing between high quality and low quality information that is related to the same topic, especially as the volume of information that is available on a particular topic increases in size (Claypool *et al.*, 1999).

1.3.3.2 Demographic Filtering

Demographic information about users has also been employed to establish preferences for certain types of items (Krulwich, 1997; Pazzani, 1999). For example, information on

age, gender, address, education, employment, etc. can be used along with preference data to learn what type of person likes a particular product. Based on such information, users can be classified into groups of similar people from which future predictions and recommendations can be derived.

In some domains, however, acquiring demographic information can be difficult. In this regard, various data acquisition methodologies can be used. LifeStyle Finder (Krulwich, 1997) engages in a dialog with users in order to elicit the required information. The goal is to identify the demographic cluster that best describes each user. If more than one demographic cluster matches the user–supplied data, a series of questions are then posed which are designed to optimally reduce the set of matching clusters and further refine the classification process. Machine learning techniques have also been employed to classify users – for example, such techniques can be used to automatically obtain information from web pages, etc.

1.3.3.3 Hybrid Systems

To combine the strengths and overcome the limitations that are associated with each of the various filtering techniques, hybrid recommender systems have been developed (Burke, 2000; Tran and Cohen, 2000; Melville *et al.*, 2002). Typically, these systems employ a collaborative filtering component together with one or more of the other filtering techniques. Burke (2002) provides a description of some of the hybrid systems that have been developed, and uses the following classification. *Weighted* systems produce a single recommendation by combining the output of several filtering techniques. *Switching* systems choose between filtering techniques based on certain criteria that apply at a particular point in time. In *mixed* recommenders, the output from the various filtering components are presented at the same time. *Feature combination* recommenders employ features that are taken from different recommendation data sources, which are then used by a single recommendation algorithm. In *cascade* systems, recommendations obtained using one technique are refined by another. *Feature augmentation* recommenders use the output of one filter as an input feature to another. Finally, *meta–level* hybrid systems learn a model using one filter, which is then used as input to another filter. Typically, real–world recommender systems employ hybrid techniques in order to optimise performance. Some examples of such systems are PTV, a personalised TV listings service (Smyth and Cotter, 2000); GroupLens, a Usenet news articles filtering system (Sarwar *et al.*, 1998); Libra, a book recommender system (Mooney and Roy, 2000) and Fab, a web page recommendation service (Balabanovic and Shoham, 1997). In this thesis, we focus on the robustness of collaborative filters, and do not consider other filtering techniques. Nevertheless, our work has implications for all recommender systems that employ a collaborative filtering component – if collaborative filters are vulnerable to malicious attack, then a system's combined output may also be affected.

1.4 Summary

In this chapter, a overview of recommender systems has been presented and the benefits that these systems have brought to e–commerce applications have been outlined. In addition, the information filtering algorithms that are used to implement these systems were reviewed. The review focused primarily on the collaborative filtering technique, since it is one of the most popular techniques and it has been used to considerable success in many commercial recommender systems. In the next chapter, a detailed description of the memory–based class of collaborative filtering algorithms is provided, which form the basis of the robustness analysis that is presented in later chapters.

Chapter 2

Collaborative Recommender Algorithms

2.1 Introduction

This chapter provides a detailed description of some of the collaborative recommender algorithms that are most frequently used in real–world and academic applications. In particular, a review of several neighbourhood formation schemes, similarity metrics and prediction and recommendation generation techniques is presented. In addition, some of the extensions to these algorithms that have been proposed in the literature are examined. A selection of these algorithms will then be used in subsequent chapters to evaluate the effect of the various attack types on the performance of collaborative recommender systems.

2.2 Memory–Based Systems

The goal of collaborative recommender systems is to recommend items to users in a personalised manner. Based on the transactions and preferences of similar users in a system, specific–item predictions or ranked–list recommendations are delivered that can satisfy each user's individual needs. In this thesis, system robustness is analysed in the context of memory–based algorithms which have been widely implemented and studied in many settings. The main features of these algorithms can be divided into the following steps, as shown in Figure 2.1. These steps consist of *data representation*, *neighbourhood formation* and *prediction and recommendation generation* (Herlocker *et al.*, 1999; Sarwar *et al.*, 2000b). In addition, the *similarity metrics* that are employed in the neighbourhood formation process are also of key importance. In the following sections, a review of each of these steps is presented. In addition, several enhancements to memory–based algorithms that have been proposed in the literature, such as significance weighting, case amplification, etc., are also described.

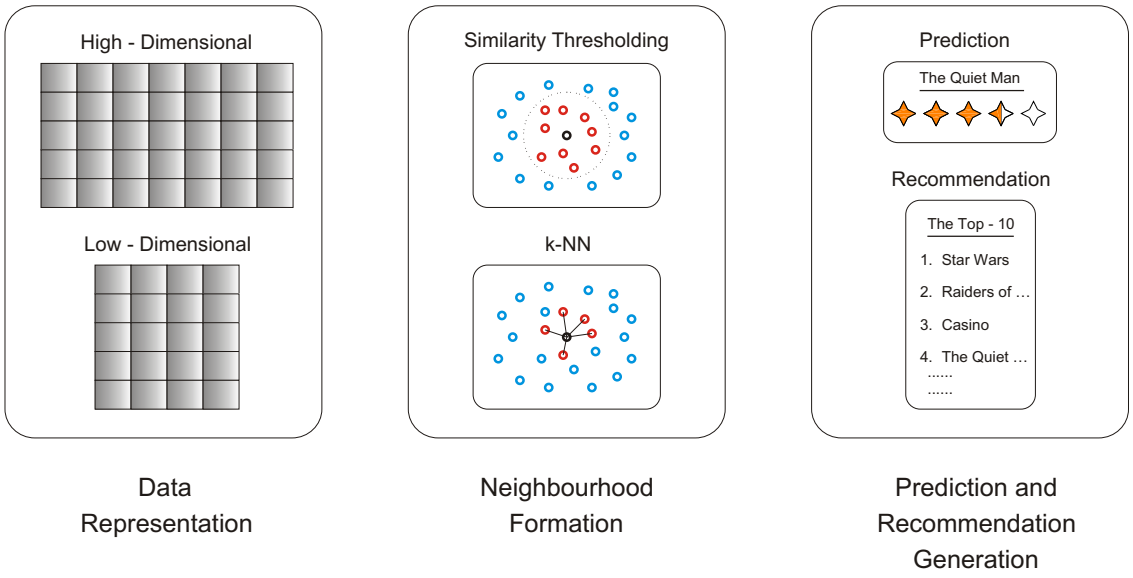


Figure 2.1: The Main Components of Memory–Based Collaborative Recommender Algorithms

2.2.1 Data Representation

The task of representation deals with the data structure used to model the system database, which contains the past transactions of all users in the system. Typically, this takes the form of a user–item matrix, where each user is represented by a unique row in the matrix and each item is represented by a unique column. Thus the $(i, j)^{th}$ entry represents the rating or vote of user i for item j. Following the notation of Sarwar *et al.* (2000b), this representation of input data is termed the *original representation*.

A common feature of many recommender system databases is the sparse nature of data contained therein, where, for the most part, the user–item matrix is empty of ratings. With potentially millions of users registered on a system, and with many items being offered for sale or review, it is reasonable to expect that the majority of users have rated only a fraction of the available items. Consider, for example, Amazon.com, the on–line book store, where many different book titles and other products are on offer. In this particular scenario, it is most unlikely to find users who have purchased or reviewed even a modest proportion of the products on offer.

As a result of the usually sparse nature of the user–item matrix, collaborative filtering may be unable to find many (or any) other users in a system to formulate a recommendation for the active user. In addition, where users are found, the number of items over which the similarity between users is calculated may be small, resulting in possibly unreliable calculations, and therefore in poor quality recommendations. Stated differently, sparseness is a key factor influencing the performance of collaborative filtering algorithms.

To solve the sparseness problem, several strategies have been presented. For example, a lower dimensional representation of the original matrix can be computed using latent semantic indexing (LSI) (Foltz, 1990; Sarwar *et al.*, 2000a). In essence, LSI obtains a rank–k approximation of an $n \times m$ user–item matrix by using singular value decomposition (SVD), a well known matrix factorisation technique. This representation of input data is referred to as *reduced dimensional representation* by Sarwar *et al.* (2000b). The new representation helps to solve the sparseness problem as the $n \times k$ matrix contains all non–zero entries, with users now having a preference for each of the k *meta–items*. A further benefit of this representation is that, since k is much less than n, scalability problems are also reduced. Finally, the LSI approach improves the likelihood of capturing the latent associations that are often present between users. Latent associations refer to the cases where users have expressed a preference for different, but related, items. These associations are missed when using many of the typical similarity metrics on non–reduced input data.

Another solution lies in the implementation of hybrid recommender systems (Basu *et al.*, 1998; Tran and Cohen, 2000). Such systems, employing content–based or other filtering techniques along with a collaborative component, can use the former technique(s) to drive the recommendation process when insufficient data is present to enable the collaborative filtering component to function effectively. When, over the course of time, more data is made available to a system, the collaborative filtering component begins to take a more active part in the recommendation process.

2.2.2 Similarity Metrics

Similarity metrics are a critical component of memory–based algorithms, and much work has been carried out to compare the performance of various metrics with respect to recommendation accuracy (Breese *et al.*, 1998; Herlocker *et al.*, 1999; Sarwar *et al.*, 2000b). The principle similarity metrics that have been used in the literature are *Cosine similarity*, *Pearson correlation*, *constrained Pearson correlation*, *Spearman rank correlation* and *mean squared difference*.

2.2.2.1 Cosine Similarity

Cosine (or vector) similarity is a widely used technique in the field of information retrieval, where the similarity between documents is computed as the cosine of the angle between the word vectors that are associated with each document (Salton and McGill, 1983). In collaborative filtering systems, users can be represented by vectors in the m–dimensional item space (Breese *et al.*, 1998). Thus, the similarity between any two users a and i can

be calculated as the cosine of the angle between the vectors as follows:

$$w(a,i) = \sum_{j \in I_a \cap I_i} \frac{r_{a,j}}{\sqrt{\sum_{k \in I_a} r_{a,k}^2}} \frac{r_{i,j}}{\sqrt{\sum_{k \in I_i} r_{i,k}^2}} \tag{2.1}$$

where $r_{a,j}$ is the rating assigned to item j by user a and I_a and I_i are the set of items that are rated by users a and i, respectively. This weight gives a value between -1 and 1 and is 0 for any given pair of users if they have not rated any common items, i.e. $w(a,i) = 0$ if $I_a \cap I_i = \phi$. The normalisation terms in the denominator are important to ensure that users who have rated many items are not *a priori* more similar to other users. Note that, because of normalisation, the weight tends to be higher if the items that users have in common are strongly *liked* or *disliked* – i.e. receive high (absolute) ratings – by both users.

Many recommender systems operate over *positive* rating scales – for example, the MovieLens system[1] uses a 5–point integer scale of 1 to 5. For these systems, the angle, θ, between any pair of user vectors is constrained to lie in the range $0 \leq \theta < \frac{\pi}{2}$. This effect is shown in Figure 2.2 for a 2–dimensional system, where the ratings that are assigned for two items, a and b, are considered. In the figure, System 1 uses an integer rating scale of 1 to 5 and the maximum angle, $\theta_{\max}$, that can exist between any two user vectors, $\mathbf{u}$ and $\mathbf{v}$, is shown. In this case, the maximum angle is achieved when $\mathbf{u} = (1,5)$ and $\mathbf{v} = (5,1)$ (or *visa–versa*), and is equal to 1.18 radians.

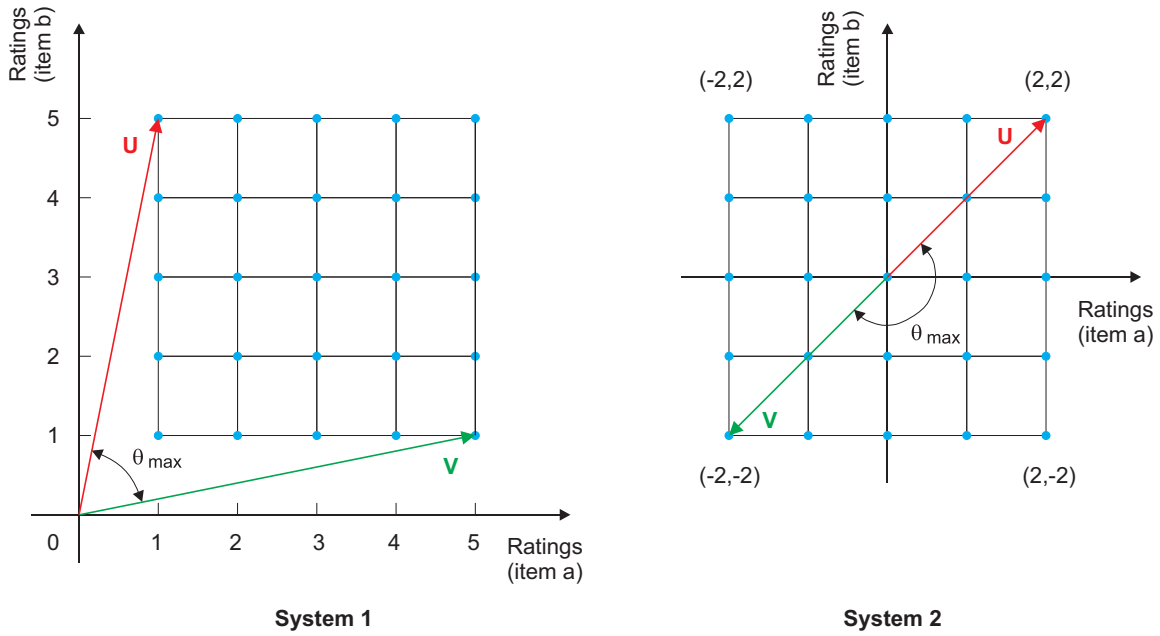


Figure 2.2: The effect of the type of rating scale that is used on Cosine similarity. System 1 uses a 5–point integer rating scale of 1 to 5. System 2 also uses a 5–point rating scale, but ratings range from -2 to $+2$.

[1] `http://movielens.umn.edu/`

This situation changes when rating scales are centered around 0. In System 2, where an integer rating scale ranging from -2 to $+2$ is used, the angle between vectors is no longer constrained and lies between $0 \leq \theta \leq \pi$ (Anton, 1994). Thus, the maximum angle between users is now π radians. In consequence, any *negative* similarities that exist between users can now be captured, whereas only positive similarities between users were possible in System 1. Thus, rating scales which are centered around 0 are the appropriate choice when Cosine similarity is used.

For both Systems 1 and 2, the minimum angle that is achievable between users is 0 radians. It is interesting to note that when the performance of Cosine similarity was evaluated by Breese *et al.* (1998), the limitations of using positive ratings scales were not considered and none of the experimental datasets used negative ratings. In addition, positive rating scales are typically used in systems that infer preferences from user behaviour, where, for example, ratings of 1 and 0 are assigned to web pages that are visited or not visited, and there is no place for negative ratings. In practice, many recommender systems do not, in fact, utilise negative ratings, and indeed, the collaborative filtering datasets that are used for evaluation purposes in this thesis all operate over positive rating scales. It will be shown in Chapter 5 that, for Cosine similarity, attacks are more straightforward to implement when negative ratings are *not* used by a system.

2.2.2.2 Pearson Correlation

The Pearson correlation coefficient was originally proposed in the GroupLens system (Resnick *et al.*, 1994). The correlation between users a and i is given by

$$w(a,i) = \frac{\sum_{j \in I_a \cap I_i}(r_{a,j} - \bar{r}_a)(r_{i,j} - \bar{r}_i)}{\sqrt{\sum_{j \in I_a \cap I_i}(r_{a,j} - \bar{r}_a)^2 \sum_{j \in I_a \cap I_i}(r_{i,j} - \bar{r}_i)^2}} \tag{2.2}$$

where the summations are over only those items that have been rated by both users a and i. Pearson correlation results in a value between -1 and 1, irrespective of the type of rating scale that is used. A correlation of $+1$ indicates perfect agreement on all common items rated by each user, and a correlation of -1 indicates total disagreement on co–rated items.

A potential problem with Pearson correlation is that it is possible to achieve a high correlation (or anti–correlation) between users that have co–rated only a very small number of items. Correlations calculated thus are unlikely to be reliable – in general, the greater the number of co–rated items over which correlations are calculated, the more accurate the result is likely to be. A technique to overcome this problem is presented in Section 2.2.5, where some extensions to memory–based algorithms are described.

As discussed above, Cosine similarity is affected by whether or not negative ratings are used by a system. This is not the case with Pearson correlation, where it is possible to

achieve perfect correlation between users provided that both users have *agreed* on the ratings given to items, regardless of whether or not they were actually liked or disliked.

Pearson correlation was found by Breese *et al.* (1998) to outperform Cosine similarity with respect to predictive accuracy, and it is the most widely used metric in memory–based collaborative recommender algorithms.

2.2.2.3 Constrained Pearson Correlation

This weight is a special case of the Pearson correlation weight, wherein average ratings are replaced by the midpoint of the particular rating scale that is used in a system. It was proposed by Shardanand (1994) for the Ringo music recommender system and is defined as follows:

$$w(a,i) = \frac{\sum_{j \in I_a \cap I_i}(r_{a,j} - 4)(r_{i,j} - 4)}{\sqrt{\sum_{j \in I_a \cap I_i}(r_{a,j} - 4)^2 \sum_{j \in I_a \cap I_i}(r_{i,j} - 4)^2}} \tag{2.3}$$

where 4 was the midpoint of the 7–point rating scale that was used in the Ringo system. This weight, like Pearson, results in a value of between -1 and 1. The motivation behind this approach is that any data whose trend is positively (or negatively) sloped will have a positive (or negative) correlation when the standard Pearson weight is used – irrespective of whether or not items are actually *liked* by users. For example, suppose that a particular user a gave ratings of 2 and 3 to items x and y, respectively, and that another user, b, gave ratings of 5 and 7 to the same items. In this case, standard Pearson correlation results in a value of $+1$, even though only one of the users has liked the items in question. When the constrained Pearson correlation weight is used, however, the result becomes -0.85, which, in the opinion of Shardanand (1994), is a better reflection of the true situation. However, this result is only valid if the rating distributions of all users are centered around identical points, which may not always be the case. This issue is explored further in Section 2.2.4, where some of the techniques that are used to aggregate the preferences of like–minded users are discussed.

2.2.2.4 Spearman Rank Correlation

Pearson correlation is derived from a linear regression model, and depends on several assumptions made with respect to the data being correlated. For example, the distributions of the variables being correlated are assumed to be normal and the relationship between the variables is assumed to be linear. If any of these assumptions are violated, Pearson correlation can become a much less accurate indicator of similarity. Given that it is not unusual to find such violations in recommender system datasets, the Spearman rank correlation coefficient, which is similar to Pearson, provides an alternative. Spearman rank

correlation makes no assumptions with respect to the data, and operates on ranks rather than ratings. It is defined as:

$$w(a,i) = \frac{\sum_{j \in I_a \cap I_i}(\text{rank}_{a,j} - \overline{\text{rank}}_a)(\text{rank}_{i,j} - \overline{\text{rank}}_i)}{\sqrt{\sum_{j \in I_a \cap I_i}(\text{rank}_{a,j} - \overline{\text{rank}}_a)^2 (\sum_{j \in I_a \cap I_i}(\text{rank}_{i,j} - \overline{\text{rank}}_i)^2}} \tag{2.4}$$

and results, as with Pearson correlation, in a value of between -1 and $+1$. The performance of Spearman rank and Pearson correlations was compared by Herlocker *et al.* (1999), and both were found to provide similar results. The afore–mentioned authors expressed some surprise at this finding – however, this result should probably be expected, given that ratings are used to rank items, where such ratings are, in essence, ranks.

2.2.2.5 Mean Squared Difference

This approach, introduced by Shardanand and Maes (1995), provides a measure of the similarity between users by computing the mean squared difference of the ratings for those items which are co–rated by users. This weight disproportionately punishes larger differences between item ratings over smaller differences. The mean squared difference (MSD) weight is defined as follows:

$$w(a,i) = \frac{1}{|I_a \cap I_i|} \sum_{j \in I_a \cap I_i} (r_{a,j} - r_{i,j})^2 \tag{2.5}$$

and results in a value of ≥ 0, where weights that are close to 0 indicate strong similarities between users. It was found by Herlocker *et al.* (1999), however, that MSD did not perform well when compared to Pearson correlation, and thus, this particular metric will not be considered further.

2.2.3 Neighbourhood Formation

Neighbourhood formation concerns the method by which those users that contribute to the recommendation and prediction process are selected. Once the similarities between the active user and all other users have been computed, a proximity–based neighbourhood of the most similar users (or neighbours) is formed. The preferences of these neighbours are then aggregated and delivered to the active user. Research indicates that it is useful, from both the system accuracy and efficiency perspectives, to select a limited number of neighbours when calculating recommendations and predictions rather than using the entire user database (Shardanand and Maes, 1995). A further consideration when selecting neighbours, as suggested by Breese *et al.* (1998), is that neighbours with relatively high similarities to the active user (i.e. those with similarities exceeding 0.5) can be exceptionally more valuable as predictors than those with lower similarities.

Several neighbourhood formation schemes have been proposed in the literature, and descriptions of some of the most widely used schemes are provided in the following sections. With all of these techniques, consideration needs to be given to the size of the neighbourhood to ensure that adequate coverage is provided by a system. Naturally, it is desirable for any system to provide as close to full coverage as possible. However, as stated above, the contributions from the typically few, exceptionally valuable (i.e. most similar) users may get lost in the noise from the many less similar users if neighbourhood sizes are configured too large. Therefore, a compromise often exists between the coverage and accuracy performance that a system can deliver, and this needs to be taken into account when setting the parameters of the neighbourhood formation techniques that are used in collaborative recommender algorithms.

2.2.3.1 k–Nearest Neighbour

The k–nearest neighbour (k–NN) scheme (Herlocker *et al.*, 1999; Sarwar *et al.*, 2000b) forms a neighbourhood of size k by simply selecting the k users with the highest (absolute) similarity to the active user. The neighbourhood size that is selected is application dependent, and the optimal size, from the point–of–view of system accuracy, is typically determined by experiment. Picking a large neighbourhood size can result in good coverage, but may dilute the influence of the most similar users in the neighbourhood. If the neighbourhood size selected is too small, poor quality recommendations may result for those users who do not have many (or any) close neighbours. The general trend is that predictive accuracy improves until a certain neighbourhood size is reached, and beyond this particular size system accuracy begins to deteriorate.

2.2.3.2 Similarity Weight Thresholding

In this scheme, a similarity threshold is set, and those users whose (absolute) similarities with the active user exceed the threshold value are selected as neighbours (Shardanand and Maes, 1995). The motivation behind this scheme is to limit the neighbourhood to the highest value neighbours. Setting a high threshold restricts the neighbourhood to very similar users, with corresponding benefits for recommendation accuracy, but for many users close neighbours are not available. If potential neighbours are unable to satisfy the threshold criterion, a neighbourhood cannot be formed and thus, recommendations and predictions cannot be made. Setting a low threshold leads to large sized neighbourhoods, and therefore nullifies the purpose of thresholding. This scheme was compared to the k–NN scheme in experiments conducted by Herlocker *et al.* (1999), and provided better performance in terms of predictive accuracy at higher threshold values. Neighbourhood thresholding did, however, result in a significant loss in coverage, while in comparison,

the k–NN scheme provided almost complete coverage and also delivered predictions to a high standard.

2.2.3.3 Combined Nearest Neighbour and Thresholding Approach

This approach combines the features of the k–NN scheme and the similarity weight thresholding scheme, where neighbourhoods of fixed size are constructed with users whose similarity to the active user exceeds a particular threshold value. For example, Herlocker *et al.* (1999) found that good performance in terms of predictive accuracy and coverage was achieved when a minimum absolute correlation threshold of 0.1 was set and when maximum neighbourhood sizes were fixed at 20. In other work (Lam and Riedl, 2004; Shardanand, 1994), the threshold criterion that is applied differs from the above in that all users with correlations that are less than the threshold value are excluded, including those that are *negatively* correlated with the active user. From analysis that is presented by Shardanand (1994), the relative frequency of negatively correlated users was found to be low, and the conclusion that was drawn was that such users can therefore be safely excluded from neighbourhoods.

2.2.3.4 Aggregate Neighbourhoods

This scheme forms a neighbourhood by comparing each user's similarity with the neighbourhood centroid, where the neighbourhood centroid is computed from the active user and those neighbours already selected at any given point in time. Thus, this scheme allows the nearest neighbours to influence the formation of the neighbourhood, and therefore may be beneficial to applications that operate using very sparse datasets. From experiments that were conducted on such a dataset by Sarwar *et al.* (2000b), however, the aggregate neighbourhood formation approach was outperformed by k–NN neighbourhood formation in terms of recommendation accuracy, and thus, it is not considered further in this thesis.

2.2.3.5 Trust–Based Neighbourhoods

In the above neighbourhood formation schemes, neighbours are selected on the basis of similarity to the active user. O'Donovan and Smyth (2005) argue that this approach can be augmented by also considering the trustworthiness of potential neighbours. In this work, computational models of trust are developed and are integrated into standard collaborative filtering algorithms. Trust is expressed as a function of a user's previous recommendation performance – i.e. the more accurate a user's past recommendations have been, the more likely it is that the user is trustworthy and suitable as a neighbour. In addition, it is shown that the inclusion of trust in the algorithm leads to an improvement

in predictive accuracy. In the scheme proposed by Massa and Avesani (2004), a social network of trust is constructed from trust data that is provided by users. This trust data overcomes the sparseness and coverage problems that are associated with many recommender system domains, since trust values between unrelated users can be inferred from indirect relationships. Thus, by using a relevance measure based on trust in addition to, or as an alternative to, user similarity measurements, a system is able to make predictions for users that traditional collaborative filters may be unable to provide. In addition, the trust network can improve the scalability problems that arise as a system grows in size by pre–filtering neighbours on the basis of predicted trust values. Other related work in this field has been proposed by Massa and Bhattacharjee (2004), Avesani *et al.* (2004) and Montaner *et al.* (2002).

In this thesis, we apply a robustness analysis to the similarity–based neighbourhood formation schemes as described previously. We do not consider the trust–based approaches that are outlined in this section, although one could reasonably expect that robustness may well be an issue for these systems also.

2.2.4 Prediction and Recommendation Generation

Recommender systems typically assist users to make choices in two ways. Users who wish to make decisions about specific items can request a system to provide predictions for the items in question. Alternatively, users can interrogate a system to recommend a ranked–list of items that are likely to be of interest to them.

In the former case, there are a variety of techniques that can be used to aggregate neighbours' ratings and provide a prediction. Here, three such techniques are described, which are referred to as the *simple weighted average* approach, the *deviation from mean* approach the *z–scores* approach.

2.2.4.1 The Simple Weighted Average Approach

The basic approach that can be used to combine neighbours' preferences is to take a simple weighted average of the neighbours' ratings. For example, Shardanand (1994) uses neighbourhood thresholding and the mean squared distance similarity metric to calculate a prediction, $p_{a,j}$, for user a, item j as follows:

$$p_{a,j} = \frac{\sum_{i=1}^{n} w(a,i) r_{i,j}}{\sum_{i=1}^{n} w(a,i)} \ . \tag{2.6}$$

In the above equation, n is the neighbourhood size and $r_{i,j}$ is the rating given by neighbour i for item j. The weights, $w(a,i)$, are calculated as $w(a,i) = \frac{L - D(a,i)}{L}$, where $D(a,i)$ is the

similarity between users a and i, obtained using the mean squared distance metric, and L is the similarity weight threshold. The assumption that is inherent in this approach is that all users rate items on approximately the same distribution. This is not always, however, a safe assumption to make and this observation has lead to the following approach being introduced.

2.2.4.2 The Deviation from Mean Approach

In this approach, which appeared in the context of the GroupLens project (Resnick *et al.*, 1994), it is assumed that the rating distributions of users may be centered around different points. For example, on an integer rating scale of 1 to 5, some users may tend to give low ratings for most items, with even liked items only receiving a rating of, say, 3 or 4. For these users, maximum predictions should only be delivered by systems on a relatively infrequent basis. In contrast, other users may generally give higher ratings to items, and would expect to receive high or maximum predictions for the majority of liked items.

The formulation of the following algorithm, often called *Resnick's* algorithm, removes any such rating inconsistencies between users from the prediction process. Predictions are calculated by aggregating the average deviation of the neighbours' ratings from each of the neighbours' mean ratings. Using this approach, the predicted rating that is made for a user a on an item j is given by:

$$p_{a,j} = \bar{r}_a + \frac{\sum_{i=1}^{n} w(a,i)(r_{i,j} - \bar{r}_i)}{\sum_{i=1}^{n} |w(a,i)|} \tag{2.7}$$

where n is the number of users in the neighbourhood, $w(a,i)$ is the similarity weight between users a and i and $\bar{r}_i$ is the mean rating for user i, which is calculated over all the items that user i has rated.

It has been established by Herlocker *et al.* (1999) that Resnick's algorithm outperforms the simple weighted average approach described above in terms of predictive accuracy. This algorithm, in conjunction with some extensions that are described in Section 2.2.5, has now been widely implemented in collaborative recommender systems and frequently serves as a benchmark against which to evaluate the performance of new approaches (Bridge and Kelleher, 2002; Canny, 2002; Rafter and Smyth, 2003).

2.2.4.3 Z–Scores

In (2.7), no consideration is given to the differences in spread between the neighbours' rating distributions, and in what way such differences might impact on recommendation quality (Herlocker *et al.*, 1999). By converting all of the neighbours' ratings to z–scores,

these differences can be factored into Resnick's algorithm by taking a weighted average of the z–scores as follows:

$$p_{a,j} = \bar{r}_a + \sigma_a \frac{\sum_{i=1}^{n} w(a,i) \, \frac{(r_{i,j} - \bar{r}_i)}{\sigma_i}}{\sum_{i=1}^{n} |w(a,i)|} \tag{2.8}$$

where σ_a is the spread of the rating distribution of user a. From experiments conducted by Herlocker *et al.* (1999), however, comparable prediction quality was achieved when using the above approach and Resnick's original algorithm. This suggests that differences in spread between neighbours' rating distributions do not have a significant impact on predictive accuracy.

When recommender systems are used to generate a ranked–list of recommendations, several approaches have been outlined in the literature. In the following sections, a description of three such schemes is provided.

2.2.4.4 Most Frequent Item Recommendation

In this approach, as described by Sarwar *et al.* (2000b), a candidate set of items for recommendation is first selected by taking the union of all the items that have been rated by the active user's neighbours. From this set, those items that have already been rated by the active user are excluded, and the remainder are ranked in terms of the number of ratings that each item has received. The N most frequently occurring items are then returned as recommendations to the active user.

2.2.4.5 Most Liked Item Recommendation

This approach is similar to the most frequent item approach, wherein a candidate set of items for recommendation is first obtained as before. Predictions for the active user are then made for all candidate items, and items are ranked in descending order according to predicted ratings. The N most liked items are returned as recommendations. This recommendation strategy is used by Lam and Riedl (2004).

2.2.4.6 Association Rule–Based Recommendation

Sarwar *et al.* (2000b) use the data mining technique of association rules to identity items that are suitable for recommendation. Association rules are concerned with discovering association between two sets of items such that the presence of some items in a transaction implies that items from another set are also present in the same transaction. The quality of the rules that are discovered is typically evaluated in terms of *support* and *confidence*. Support is a measure of the frequency of occurrence of a rule in a system and confidence

provides a measure of the strength of implication, i.e. the conditional probability that both sets of items are present in a transaction given that one of the sets is present. This technique generates recommendations based on the association rules relating to those items that have already been rated by a user, with new items, that have the best support and confidence, being returned as recommendations.

Experimental evaluation indicates that there is no significant difference in performance between this approach and the most frequent item recommendation approach. Given the speed and simplicity of the latter approach, it is recommended by Sarwar *et al.* (2000b) that it should be used in preference to association rule–based recommendation.

2.2.5 Extensions to Memory–Based Algorithms

Several extensions to memory–based algorithms have been proposed in the literature from a predictive accuracy point–of–view. In the following sections, a review of some of these extensions is presented.

2.2.5.1 Significance Weighting

Similarity weights calculated on the basis of a small number of common items provide an unreliable measure of 'true' similarity between users. Indeed, if the number of common items between users is exactly 2, Pearson and Spearman rank similarity measures always give a result of $+1$, -1 or 0. Therefore, the probability of such users forming a substantial part of a neighbourhood is high, even though they may be, in reality, poor predictors for the active user. Since sparseness is inherent in many collaborative filtering datasets, it is reasonable to expect that many neighbourhoods are at least partially formed by poor quality neighbours, and thus the quality of recommendations is likely to be poor.

Resnick's algorithm as expressed above does not take these consideration into account. To solve this problem, significance weighting has been proposed by Herlocker *et al.* (1999). This technique modifies estimated similarity weights based on the number of common items between users as follows

$$w'(a,i) = \begin{cases} w(a,i) \times \frac{n}{N} & \text{if } n < N \\ w(a,i) & \text{if } n \geq N \end{cases} \tag{2.9}$$

where N is an arbitrary constant. With this technique, similarity weights calculated on the basis of smaller numbers of common items are appropriately devalued, while weights calculated over large numbers of common items ($> N$) are unaffected. Thus, significance weighting captures the intuition that the greater the degree of overlap between users, the more accurate the result is likely to be.

2.2.5.2 Case Amplification

Case amplification, proposed by Breese *et al.* (1998), is designed to emphasise weights that are close to 1 and to reduce the influence of lower weights. Weights are transformed by the following equation

$$w'(a,i) = \begin{cases} w^{\,\rho}(a,i) & \text{if } w(a,i) \geq 0 \\ -\Big(- w(a,i)\Big)^{\rho} & \text{if } w(a,i) < 0 \end{cases} \tag{2.10}$$

where ρ is an arbitrary constant. As is the case with significance weighting, this technique can lead to the formation of more accurately defined neighbourhoods. For example, neighbourhoods formed under similarity weight thresholding can see less valuable users removed from the neighbourhood altogether, as lower weights may be transformed below the threshold value.

2.2.5.3 Inverse User Frequency

Inverse user frequency (Breese *et al.*, 1998) is analogous to the concept of *inverse document frequency* from the information retrieval domain. In this domain, similarity measurements can be based on the frequency of occurrence of words that appear in documents. However, it has been found that frequently occurring words are less indicative of the topics contained in a document, while words that occur less frequently are more so. Inverse document frequency takes these factors into account and transforms the weights associated with each word accordingly.

In the context of collaborative filtering, ratings take the place of words, and the goal is now to minimise the influence of very popular items in the calculation of similarity weights. The motivation is, as before, that less popular items are better able to discriminate between users than more popular items. Inverse user frequency is implemented as follows. The term f_j is defined as $\log \frac{n_d}{n_j}$, where n_j is the popularity of item j in the database and n_d is the total number of users in the database. The Pearson correlation weight is modified to incorporate this term as follows[2]:

$$w(a,i) = \frac{\Big(\sum_j f_j\Big)\Big(\sum_j f_j r_{a,j} r_{i,j}\Big) - \Big(\sum_j f_j r_{a,j}\Big)\Big(\sum_j f_j r_{i,j}\Big)}{\sqrt{UV}} \tag{2.11}$$

where $j \in I_a \cap I_i$ and

$$U = \Big(\sum_j f_j\Big)\Big(\sum_j f_j r_{a,j}^2\Big) - \Big(\sum_j f_j r_{a,j}\Big)^2$$

[2]Note that the definition of inverse user frequency given above differs from that as presented by Breese *et al.* (1998). See Appendix A for details.

$$V = \Big(\sum_j f_j\Big)\Big(\sum_j f_j r_{i,j}^2\Big) - \Big(\sum_j f_j r_{i,j}\Big)^2$$

The Spearman rank correlation coefficient is adapted in a similar fashion. For Cosine similarity, the weight with inverse user frequency applied becomes:

$$w(a,i) = \sum_{j \in I_a \cap I_i} \frac{f_j r_{a,j}}{\sqrt{\sum_{k \in I_a}(f_k r_{a,k})^2}} \frac{f_j r_{i,j}}{\sqrt{\sum_{k \in I_i}(f_k r_{i,k})^2}} \; . \tag{2.12}$$

2.2.5.4 Item Entropy

The concept of item entropy (Yu *et al.*, 2001) is related to the above, in that it attempts to minimise the influence of certain items in the calculation of similarity weights. In this case, the intuition is that items which are rated consistently by many users are less able to discriminate between users than those that receive a more uniform distribution of ratings. Let $p_{r,j}$ be the probability of item j receiving a rating of r, and define $H(j)$, the entropy of item j, as:

$$H(j) = -\sum_i p_{r,j} \ln(p_{r,j}) \; . \tag{2.13}$$

To incorporate entropy measurements into the various similarity measures, equations 2.11 and 2.12 are used with the term f_j re–defined as $\frac{H(j)}{H(j)_{max}}$, where $H(j)_{max}$ represents the maximum entropy of item j with all ratings assumed to be uniformly distributed over the particular rating scale in question. This term avoids the issue of having different discrete rating scales for different items. With this formulation, items that are rated consistently in a database receive low values for f_j and thus have a reduced influence on similarity weight calculations, while items that are rated in a more diverse fashion are correspondingly more influential.

2.2.5.5 Item Selection Strategies for Efficient Similarity Computation

This approach, proposed by Rafter and Smyth (2003), seeks to improve the efficiency of the collaborative filtering process by computing the similarity between users based on only a limited number of shared ratings. In particular, the authors base their analysis on Resnick's algorithm and the Pearson correlation similarity metric. Three different item selection strategies are proposed. The first strategy involves choosing k items which have the highest inverse popularity. Inverse popularity is calculated for an item j as $\text{InvPop}(j) = 1 - \frac{n_j}{n_d}$, where n_j is the number of users who have expressed a preference for item j and n_d is the total number of users in the system. In the second strategy, the k items which have received the most diverse ratings are selected. Item rating diversity is

defined as the standard deviation of all the ratings submitted by users for each particular item. The motivation behind these strategies is that user profiles containing items which have received relatively few or diverse ratings are more likely to serve as better predictors. The final strategy adopted is to choose k items at random. Experimental results indicated that the random strategy performed relatively poorly (as expected), whereas the former strategies provided good and consistent performance for all values of k between 20 and 100. These results suggest that efficiency gains can indeed be achieved in the manner outlined, without incurring a significant loss in predictive accuracy. The efficiency of this approach is $O(r)$ compared to $O(r.i)$ for Resnick's algorithm, where r is the number of users who have rated the item for which a prediction is being sought and i is the average number of items that have been rated by each of these users.

2.3 Summary

In this chapter, a review of memory–based algorithms has been presented. In particular, Resnick's deviation from mean algorithm, which is widely used in the prediction generation process, has been described. A review of the similarity metrics and neighbourhood formations schemes that have been used in conjunction with this algorithm has also been presented. In addition, some of the extensions to the algorithm that have been proposed from a predictive accuracy perspective have been described. This algorithmic framework will be used to evaluate the concept of the robustness of collaborative filtering systems as developed in subsequent chapters of this thesis.

Chapter 3

Performance of Recommender Systems

3.1 Introduction

In this chapter, a review of the existing performance measures that are used to evaluate recommender systems is presented. Recommender systems have been extensively evaluated in the literature along the dimensions of prediction and recommendation accuracy, efficiency, scalability and coverage, and much work continues to be done in order to improve system performance according to these criteria. In the following sections, each of the afore–mentioned performance criteria is explored in detail, and a review of the associated metrics is also provided.

3.2 Performance Criteria

The performance of any system can only be measured with respect to some utility function which models the usefulness of a system's output. Clearly, the usefulness or utility of a system's output depends on the perspective of the parties involved. In recommender systems, three distinct parties, each with specific needs (though sometimes related), can be identified. These are:

- **End–users**. This group are the primary clients of the system, to whom recommendations and predictions are delivered. From this perspective, utility primarily depends on recommendation quality, i.e. how well the system follows the end–users' personal tastes. In addition, factors such as coverage – the ability of a system to provide the requested recommendations – and efficiency – the time taken to deliver recommendations – are also important.
- **System owners**. The primary interest of the system owner is in the throughput of the system as measured by the total number of transactions. From this perspective,

the recommender system need not actually be accurate, so long as it attracts and retains customers to the service. The coverage, efficiency and scalability – delivering timely recommendations irrespective of database size – provided by the algorithm used to drive the recommendation process are further concerns of the system owner.

- **External interested parties**. These parties have a direct interest in the transactions that are made by the end–users of the system. Consider, for example, the case of a book recommendation system, where such a party might be the author or publisher of some of the books that are being recommended. Obviously, these third parties have a vested interest in the recommendations that are generated for their products, but they do not have direct access to the recommender system, other than through the normal user–interface.

In the analysis to date, the performance of recommender systems has focused on the perspectives of end–users and system owners. To this end, the principle criteria introduced to evaluate system performance are *efficiency*, *scalability*, *coverage*, *predictive accuracy* and *recommendation accuracy*. Leaving aside issues such as interface design, user interaction and output presentation, any system that satisfies these criteria is likely to meet with a reasonable level of success.

The above performance measures do not, however, necessarily reflect the objectives of all external parties. In general, external parties can be expected to increase the probability of receiving good recommendations for their products by attending to issues such as purchase cost and quality. While this may well be the prevailing view among these parties, one must assume that some may be motivated to compromise a system in order to produce more favourable product recommendations. Thus, it is necessary to devise solutions to protect systems against such activity, should it occur. These matters are considered in subsequent chapters. In the following sections, a discussion on each of the performance criteria that were mentioned above is provided.

3.2.1 Coverage

Coverage is defined as the percentage of items for which a system can provide predictions. Collaborative recommender systems typically form a recommendation for the active user by combining the preferences of a number of similar users contained in the system. Thus, the popularity of individual items in a system has a significant impact on coverage. For example, given the usually sparse nature of recommender system datasets, some items may be un–rated in the system, or may be rated by only a small number of users who have no other items in common with the active user. In either case, no prediction for such items can be made by a system. This issue is of particular concern in recently launched applications (i.e. the 'cold start' problem), and indeed, since sparseness is likely

to remain a feature of many, if not most, recommender databases, adequate coverage is an ever–present problem. Thus, many recommender systems employ a combination of information filtering techniques to overcome this issue – for example, systems may employ content–based and collaborative filtering techniques to improve the coverage provided by the system at all stages of the system's deployment.

The choice of neighbourhood size – the number of users selected to generate recommendations – is an important factor in, among others, ensuring wide system coverage. Small neighbourhood sizes can result in a reduction in coverage, whereas larger neighbourhoods are more likely to contain a greater variety of items. In addition, the neighbourhood formation technique used has an impact on coverage for both prediction and ranked list generation. For example, sampling of users to find neighbours and prior clustering of a database into groups of users (from which all neighbours are subsequently selected), tend to limit the coverage that a system is able to provide.

3.2.2 Efficiency and Scalability

Recommender systems that employ memory–based collaborative filtering algorithms are known to suffer from serious scalability problems – i.e. as a database grows in size, the number of computations required to generate a prediction increases accordingly (Sarwar *et al.*, 2000b). Efficiency is therefore a key concern for recommender systems and many solutions have been proposed to improve system performance. For example, the LSI technique, which forms a reduced dimensional representation of the user–product space, is known to have beneficial impacts from the sparseness and scalability perspectives (Sarwar *et al.*, 2000b; Berry *et al.*, 1995). Perhaps the most efficient approach to collaborative recommendation would be to randomly select neighbours from the available set of users, thereby allowing recommendations to be generated with only minimal computation. There is, however, no guarantee that neighbourhoods so–formed are optimal or near–optimal, and thus, in addition to possible coverage issues, the quality of recommendations using this method is likely to be reduced.

The clustering approach, wherein a database is first partitioned into distinct groups of like–minded users from which all neighbours are subsequently drawn, would appear to offer a viable solution to the scalability problem. Since well–defined clusters should contain the majority of the neighbours for each user in a given cluster, recommendation quality should not be significantly compromised. Such an approach is adopted by Bridge and Kelleher (2002), where separate experiments were conducted to evaluate the coverage, efficiency and accuracy of clustering both users *and* items. In addition, Bridge and Kelleher (2002) propose an implementation–independent measure of efficiency, termed *total rating comparisons* (TRC). TRC can be used to compare the efficiency of different systems and is defined as the number of rating comparisons carried out by a system in

making a prediction. Experimental results indicated that, while the clustering of users provided a significant increase in efficiency (compared to considering all users when forming neighbourhoods), a corresponding decrease in both system coverage and accuracy was observed. Efficiency was improved because relatively few rating comparisons were required to generate predictions. Coverage and accuracy suffered because, in the former case, the clustering technique that was used confined each user's ratings to only one cluster (and thereby denied other clusters of this data), and in the latter case, the nearest neighbours for users were not always contained within the respective clusters. When clustering was performed on items, efficiency was found to be poor, although coverage and accuracy remained high. This reduction in efficiency can be explained by the fact that when several items were clustered into *super–items*, many more rating comparisons were subsequently required to make predictions for these super–items than for any of the individual items that formed such clusters.

Although the experimental results outlined above suggest that clustering users does not provide a meaningful solution to efficiency problems (i.e. without suffering significant losses in coverage and accuracy), it is worth considering some observations that are made by Mirza *et al.* (2003). In this work, it is noted that users buy and rate different categories of products in qualitatively different ways. For example, movie databases tend to follow a *hits–buffs* distribution, where many users have seen many of the hit movies and where movie buffs, although comparatively few in nature, have seen many of the movies. Thus, movie databases tend to be well connected and hence it is difficult to derive good quality clusters. Other product domains, however, are known to be more clustered. For example, in music domains, the hits–buffs structure is also present, but typically only within specific genres. Connections within particular genres are therefore quite strong, whereas connections between different genres, while present, are generally weaker. Thus, clustering techniques are likely to work better for these kinds of domains. (It should be noted that the experiments conducted by Bridge and Kelleher (2002) were performed using the MovieLens movie database[1], and the results obtained can therefore be rationalised with respect to the above observations).

3.2.3 Accuracy

System accuracy is, of course, a vital consideration for recommender systems. Users can readily assess the accuracy of any system by seeking and examining recommendations for items for which they already have a preference. Any system that fails to meet users' expectations by producing poor quality recommendations is unlikely to retain a significant client–base, with all the implications for sales and advertising revenue that follow.

[1] `http://movielens.umn.edu/`

Thus, much research has been conducted into evaluating and improving the accuracy of recommender systems.

Many of the specific algorithmic factors that influence accuracy – such as the neighbourhood formation technique employed, choice of similarity weight, etc. – were discussed in detail in Chapter 2. In this section, the focus is on the various accuracy evaluation techniques that have been proposed in the literature. Many different accuracy metrics have been used, ranging from machine learning and information retrieval approaches through to statistical formulations.

It is useful to examine these metrics in the context of developing a robustness performance measure, which is introduced in Chapter 4. To begin, several metrics that have been frequently used to evaluate the accuracy of specific–item predictions are described. Such measures include *accuracy*, *mean absolute error*, *mean squared error*, *correlation* and *receiver operating characteristic*.

3.2.3.1 Accuracy Metric

Accuracy is a measure of the fraction of correct predictions that are made by a system to the total number of predictions that are sought (Geyer-Schulz and Hahsler, 2002). It is widely used in the machine learning community and is defined as follows:

$$\text{Accuracy} = \frac{\text{number of correct predictions (within } \pm \epsilon)}{\text{total number of predictions}} \tag{3.1}$$

where ϵ is an error limit. There are two methods by which accuracy may be computed for systems that operate over explicit ratings (i.e. where users are required to express a preference for items using some form of rating scale). In the first method, predicted ratings are compared directly to the true ratings that are expressed for items. A correct prediction occurs when a predicted rating is equal to the true rating. In the second method, user ratings are first 'interpreted' by a system in the following manner. Consider, for example, a system that operates on a 5–point rating scale, where ratings of 4 and 5 may be considered to indicate a positive preference, and all other ratings are assumed to indicate a negative preference. A correct prediction is now said to occur if the predicted and actual ratings for an item are both indicative of either a positive or negative preference. This approach mirrors the situation where recommender systems operate over implicit ratings, wherein a rating of 1 is typically used to indicate the purchase of an item or a visit to a web page (i.e. a positive preference), and a rating of 0 is used to indicate the non–purchase of an item, etc. (i.e. a negative preference).

3.2.3.2 Mean Absolute Error

Mean absolute error or MAE is a measure of the absolute difference between predicted and actual ratings, and has been widely used in the literature to evaluate the predictive accuracy of recommender systems (Breese *et al.*, 1998; Geyer-Schulz and Hahsler, 2002). The mean absolute error for a prediction made for a user a on an item j is defined as:

$$\text{MAE}_{a,j} = |p_{a,j} - r_{a,j}| \tag{3.2}$$

where $p_{a,j}$ and $r_{a,j}$ are the predicted and actual ratings of user a for item i, respectively. This quantity can then be averaged over all users in a test set to calculate the overall mean absolute error of a system.

3.2.3.3 Normalised Mean Absolute Error

Normalised mean absolute error (NMAE) is a variant of the above metric, wherein the result obtained using mean absolute error is normalised as follows:

$$\text{NMAE}_{a,j} = \frac{\text{MAE}_{a,j}}{r_{max} - r_{min}} \tag{3.3}$$

where r_{min} and r_{max} are the minimum and maximum ratings that are defined on a particular rating scale, respectively. The advantage of this metric is that it facilitates a direct comparison between the accuracy of systems that operate on different rating scales (Canny, 2002).

3.2.3.4 Mean Squared Error

The mean absolute error metric described above gives equal emphasis to all errors, in that it does not significantly punish any large errors that may occur. The mean squared error metric, in contrast, has the property of disproportionately punishing larger errors over smaller errors (Miller *et al.*, 1997). The motivation behind this approach is that large errors are much more likely to be of concern to users. For example, users would almost certainly notice the discrepancy between a predicted rating of 5 and a true rating of 2, whereas a rating of 2 that is predicted for an item whose true rating is 1 is unlikely to cause a real concern for users. The mean squared error or MSE is defined as:

$$\text{MSE}_{a,j} = (p_{a,j} - r_{a,j})^2 \tag{3.4}$$

As before, the mean squared error of the entire system can be calculated by taking the average of the errors across the test set.

3.2.3.5 Correlation

The correlation between predicted and actual ratings has been used by Hill *et al.* (1995) to evaluate predictive accuracy. A strong positive correlation indicates that a system has succeeded in recovering the trends observed in the actual ratings, and thus has the ability to provide high quality predictions. Low or negative correlations indicate poor performance with respect to predictive accuracy.

3.2.3.6 Receiver Operating Characteristic

The function of recommender systems is to provide assistance to users in some decision making process, i.e. whether or not to go and see a particular movie or purchase a particular product. Thus, decision making can be viewed as a binary process, and even when recommender systems operate over rating scales, users often employ subjective *thresholds* in order to determine what constitutes a 'good' or a 'bad' prediction. For example, on a 5–point rating scale, a user may interpret a predicted rating of 4 and above as an endorsement for an item, while anything lower indicates a negative endorsement. In this case, a prediction of 1 is equivalent to a prediction of 2, in that the user would, in both cases, choose to reject the item in question. To measure the accuracy of recommender systems in this regard, the receiver operating characteristic (ROC) approach from signal detection theory can be used.

Signal detection theory is concerned with the probability of distinguishing between signals that contain noise alone from those that contain both signal and noise. In many cases, it is impossible to make such distinctions with absolute accuracy, and therefore the probability of *good detection* and *false alarm* are measured with respect to some threshold value. The rate of good detection refers to the probability of correctly identifying those instances where both signal and noise are present, and the rate of false alarm refers to the probability of incorrectly detecting a signal when it is not actually present. By varying the threshold above and below which a signal is assumed to be present or absent, a complete representation of the diagnostic power of a system can be obtained by plotting the corresponding good detection and false alarm rates. Such plots are referred to as ROC curves, where the area under the curve increases as the good detection rate improves and the false alarm rate diminishes.

ROC sensitivity has been applied by Herlocker *et al.* (1999) to measure the diagnostic power of collaborative recommender systems. In this context, *sensitivity* refers to the probability of a randomly selected good item being accepted by the filter and *specificity* refers to the probability of a randomly selected bad item being rejected by the filter. The ROC curve is thus obtained by plotting sensitivity against $1 -$ specificity for various values of the threshold. Each of the users' own ratings are used to determine good and

bad items, i.e. for a threshold value of 3, signal is indicated by those items with ratings of 3, 4 and 5 and noise is indicated by ratings of 1 and 2. (Note the similarities between this approach and the accuracy metric approach, as described in Section 3.2.3.1).

It has been noted by Herlocker *et al.* (1999) that mean absolute error, mean squared error and correlation all provide similar results. In addition, results obtained using receiver operating characteristic, the decision support metric, mirrored those of the above. Since the results obtained using mean absolute error can be readily interpreted, this metric has gained widespread use as the basis with which to evaluate the predictive accuracy of recommender systems. In the remainder of this section, some of the metrics that have been used to evaluate the utility (accuracy) of ranked–list recommendations are discussed.

3.2.3.7 Information Retrieval Measures

The *recall* and *precision* metrics are widely used in the field of information retrieval to evaluate system accuracy. These metrics have been adapted by Sarwar *et al.* (2000b) to evaluate ranked–list generation in recommender systems, and are defined, in the context of recommendation generation, as follows:

$$\text{Recall} = \frac{|T \cap R|}{|T|} \tag{3.5}$$

$$\text{Precision} = \frac{|T \cap R|}{|R|} \tag{3.6}$$

where T is the test set and R is the recommended set of items for each user, respectively. The recall and precision metrics are, however, often conflicting properties. For example, increasing the total number of predictions is likely to improve recall, but reduce precision. To resolve this conflict, the F_1 metric, which combines the recall and precision metrics, is used (Geyer-Schulz and Hahsler, 2002; Sarwar *et al.*, 2000b). It is defined as:

$$\begin{aligned} F_1 &= \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \\ &= \frac{2}{1/\text{Precision} + 1/\text{Recall}} \end{aligned} \tag{3.7}$$

The F_1 metric is a special case of the more generic E metric (Geyer-Schulz and Hahsler, 2002). It is defined as follows:

$$E = \frac{1}{\alpha(1/\text{Precision}) + (1-\alpha)(1/\text{Recall})} \tag{3.8}$$

Whereas the F_1 metric gives equal weight to both recall and precision, the weighting can be controlled in the E metric by selecting values in range of $(0, 1)$ for the parameter α.

3.2.3.8 Ranked Scoring

While the recall and precision metrics described above are suitable for recommender systems that operate over binary ratings of like or dislike, they do not take the explicit preferences (i.e. ratings) that are given to items into consideration. In order to incorporate such preferences into the evaluation of a ranked list, a further scoring metric is proposed by Breese *et al.* (1998). In this metric, the expected utility of a ranked list is defined as the probability of viewing a recommended item times its utility, where the utility of an item is defined as the difference between the actual rating given to an item by a user and the default or neutral rating in a domain. In addition, the likelihood of visiting an item on a ranked list is estimated. For this purpose, the heuristic that is used is that each successive item on the list is less likely to be viewed by a user with an exponential decay. Thus, the expected utility of a ranked list of items (sorted by index j in order of decreasing ratings, $r_{a,j}$) for a user a is given by:

$$R_a = \sum_j \frac{\max(r_{a,j} - r_n, 0)}{2^{(j-1)(\alpha-1)}} \tag{3.9}$$

where r_n is the neutral rating and α is the viewing half–life. The half–life is defined as the number of the item on a list such that there is a 50/50 chance of a user viewing that particular item. In experiments conducted by Breese *et al.* (1998), a half–life of 5 items was used (experiments were also carried out with a half–life of 10 items and no significant differences in results were observed).

The overall score for an experiment over a set of users contained in a test set is determined as follows:

$$R_t = 100 \frac{\sum_a R_a}{\sum_a R_a^{max}} \tag{3.10}$$

where R_a^{max} is the maximum achievable utility if all observed items had been at the top of a ranked list, ordered by rating value. This formulation allows consideration of results independent of the size of a test set and the number of items predicted in a given experiment.

The above metrics have been widely used in the literature to evaluate the accuracy of the recommendation process, and are relevant to the development of a robustness metric. While system accuracy and robustness are clearly related concepts, accuracy metrics are not, however, necessarily suited to a strong evaluation of system robustness. This matter is discussed in detail in Chapter 4, where the limitations of the existing metrics are outlined, and where new metrics are derived in order to evaluate the robustness of recommender systems.

3.3 Summary

In this chapter, a review of the coverage, efficiency, scalability and accuracy performance criteria that are typically used in the literature to evaluate recommender systems has been presented. In addition, the principle metrics that have been developed to quantify system performance have been outlined. In the next chapter, it is argued that a further performance measure is also required for recommender systems – that is, the ability of a system to provide robust predictions when varying degrees of biased or unbiased noise are present in the system data.

Chapter 4

Robustness of Recommender Systems

4.1 Introduction

In this chapter, the concept of system robustness is introduced as an additional performance measure for recommender systems. The motivations behind such a measure, particularly in the context of malicious attacks being carried out on systems, are explained. Related work from the field of knowledge–based systems is reviewed, and a definition of robustness for recommender systems is then proposed. To conclude, some new metrics that are designed to evaluate system robustness are introduced.

4.2 Robustness

One of the primary goals of this thesis is to complement the existing work, as described above, by considering the performance of recommender systems along a further dimension – that is, the *robustness* of the recommendation process. In essence, robustness considers how system recommendations vary with changes in the system database. In order to have confidence in system recommendations, the system's output should be stable. That is, small changes in the database should not lead to large changes in the output.

Recommender systems update their databases when users input new ratings. There is, however, no guarantee that these ratings reflect the users' true preferences. Typically, no quality assurance is performed by recommender systems when users input preferences and thus, in the normal course of events, some degree of noise in the data should be anticipated. In general, noise in recommender systems can be categorised into two classes. These classes are:

- **Unbiased Noise**. This class of noise relates to the methods by which recommender systems collect or infer the user preferences expressed for items. In the first instance,

many recommender systems operate over explicit rating schemes, wherein users are required to express a preference for items that they have reviewed or purchased. Rating input is typically facilitated by discrete or continuous rating scales. Since all human activity is prone to error, and indeed, given that rating entry is often viewed by users as an onerous process, some errors in the data will naturally occur. In addition, users may feel that particular rating scales do not offer sufficient options with which to rate items – for example, half–point ratings of, say, 3.5, may not be permitted when using some discrete rating scales.

In systems where implicit rating schemes are utilised (for example, by inferring user preference from web site visits), the potential for noisy input is perhaps greater, given the very method of rating collection. While the noise introduced in the manner described may well be relatively minor in significance and possibly infrequent in nature, nevertheless it is important to recognise the presence of such unbiased noise in a system.

- **Biased Noise**. Of more serious concern is the possibility of biased noise being deliberately inserted into a system. The insertion of such noise is termed an *attack*. Security is a major concern for all Internet systems and applications, and yet, thus far, it seems little consideration has been given to the security of recommender systems. It is certainly conceivable that malicious agents may be motivated to attack recommender systems, given the significant advantages that may accrue. Since many recommender systems operate in a commercial environment, strong motives exist for those minded to behave in an unscrupulous manner. Consider, for example, authors wishing to promote their work by forcing a recommender system to output artificially high ratings for their publications, while reducing the recommendations made for other works in the same genre. A further example is that of a recommender system owner, who in an effort to increase the market share of his own particular system, may attempt to tamper with the recommendation quality of rival systems.

It is with such malicious attacks that the robustness analysis in this thesis pertains. Even if one assumes that a system's database and recommendation algorithms are secure against attack, recommender systems remain vulnerable given the very manner in which they operate. Since it is practically impossible to assess the integrity of those who use a system (especially true for on–line systems), there is often little to prevent users from inputting bogus data through the normal user interface and mounting attacks in this fashion. In the next section, the attack types considered in this thesis and the basic methodology used to implement these attacks are described.

4.2.1 Malicious Attacks

In all the attacks considered in the thesis, it is assumed that an attacker has no direct access to the system database. The only interaction the attacker has with the system is, as with genuine users, through the normal user interface. Attacks are implemented in the following manner. An attacker, through multiple registrations with the system, assumes several identities within the application database. It is in this manner that the attacker is able to insert biased noise into the system. Each of the separate identities assumed by the attacker are referred to as *attack profiles*.

While it is true that systems owners have become aware of this issue concerning multiple registrations, it remains, in many cases, an unsolved problem. For example, consider the on–line auction house eBay[1], which has attempted to reduce the instances of such behaviour in recent times. One of the issues in question for eBay is that some sellers may attempt to inflate auction bids by also masquerading as buyers in the same auction. Users may also consider unfairly inflating their own reputation estimates by providing bogus feedback for themselves. Such action by unscrupulous users clearly represents a significant threat to the integrity of the auction process. Thus, the registration procedure adopted by eBay, which is designed to deal with these issues, includes the following steps:

- Users are requested to provide their name, street address and some additional contact information – in particular, a valid e–mail address. E–mail addresses are a key component of eBay's strategy to prevent multiple registrations.
- Since e–mail accounts are now widely available from many service providers (for example, Yahoo, HotMail etc.), it is a simple matter for people to acquire multiple accounts. Thus, if the e–mail address supplied during registration is one obtained from such a source, then users need to supply either of the following additional pieces of information:
  - details of a current credit card, or
  - a second e–mail address, one which cannot be obtained from a service provider.
- Note that if the e–mail address originally supplied is not from a service provider, i.e. if it is obtained from a company or institution, etc., then no further information is required.

While these steps represent, to some degree, a barrier to users engaging in multiple registrations, it certainly does not represent a robust solution to this problem. Further, many applications, which do not involve financial transactions between users, are not so well

[1] `http://www.eBay.com/`

protected against this activity. Thus, an attack strategy, which relies on the ability to repeatedly register with a system, is indeed feasible in many cases.

Two attack types are considered in this thesis, each of which are designed to target specific items or sets of items. The objectives of *product push* and *product nuke* attacks are to promote or demote the recommendations made for items, respectively. For example, the goal of an attacker might be to force a system to output poor recommendations for his competitors' products (nuke) while attempting to secure positive recommendations for his own (push).

To illustrate the basic attack strategy that is employed, consider the simplified recommender system database that is presented in Figure 4.1. In this example, the objective is to demote the recommendations that are made for item 7 (i.e. a product nuke attack), and a number of attack profiles (users I through M) have been inserted into the system to target this item.

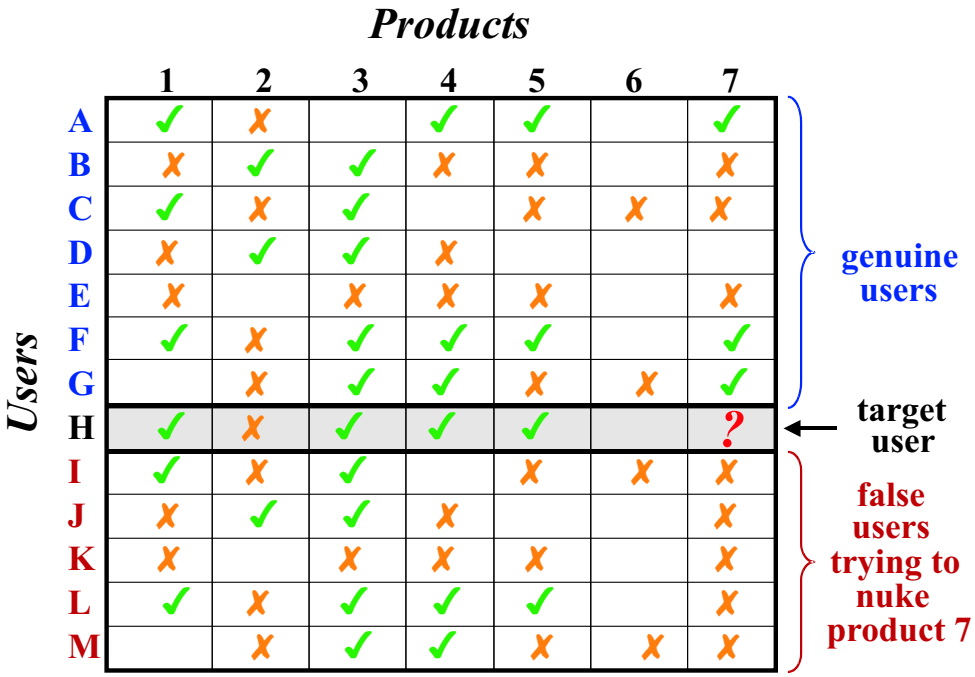


Figure 4.1: Simplified system database showing genuine user profiles and a number of attack profiles inserted. In this example, user H is seeking a prediction for item 7, which is the subject of a product nuke attack.

In particular, consider the binary recommendation problem in which the task is to predict whether or not user H likes item 7. In the first instance, let the attack profiles be ignored and consider only the genuine profiles (users A through G) as possible neighbours for the active user, H. Regardless of the specific recommendation algorithm used, presumably the algorithm would determine that users A and F are similar to the active user, and since both of these users like item 7, a positive recommendation for the item follows.

When the attack profiles are also considered as possible neighbours, the situation is, however, significantly altered. Several of these attack profiles are also similar to user H, and, since all of these profiles rate item 7 poorly, the system is now likely to recommend a negative rating for the item. Thus, the objective of the attack is realised. Clearly, if such an attack is to be successful, an attacker needs to ensure that any inserted profiles make significant contributions to the recommendation process. This matter is discussed in detail in Chapter 5, where a description of a range of attack strategies for all of the attacks outlined above is presented.

4.2.2 Related Work

In this section, a review of some of the related work that has been carried out in the field of knowledge–based systems (KBS) is presented. One of the underlying claims of KBS is that they are able to deal with incomplete, inaccurate or uncertain data. Few researchers in the field have, however, analysed the extent to which this claim is true. To measure the performance of a system, the database upon which predictions are based is typically sub–divided into test and training sets and the predictive accuracy of the test sets are measured. While this process gives an indication of how well a system performs given the training data, it does not measure how the the performance of a system evolves as a database changes. Thus, in addition to a measure of system accuracy, a measure of system robustness is also required. In this regard, the previous work of Hsu and Knoblock (1995, 1996) and Groot *et al.* (2000) is of interest, in which techniques are proposed for examining the robustness of knowledge discovery and knowledge based systems.

Groot *et al.* (2000) use as a starting point the informal definition of robustness found in (IEEE, 1990):

The degree to which a system or component can function correctly in the presence of invalid inputs or stressful environmental conditions.

The authors proceed by analysing robustness through a degradation study, in which the quality of the KBS output is measured as the quality of the input is gradually decreased. Robustness is then evaluated with respect to the precision and recall accuracy metrics. A single criterion is not proposed, however, for making a comparison between the robustness of different systems; instead, a number of robustness definitions are presented. For example, one system is considered to be more robust than another if the output decreases at a slower rate with decreasing input quality.

Hsu and Knoblock (1995) give a more precise definition of robustness, wherein the focus is on the consistency of a rule set with a database from which it is derived. An intuitive argument is advanced that the robustness of a rule, r, in the KBS can be measured as the

probability that the database is in a state that is consistent with r. As a more tractable definition of robustness, the following is proposed:

Given a database state d and a rule r that is consistent with d, let t denote the transactions on d that result in new database states inconsistent with r. The robustness of r in the database state d is Robust$(r|d) = 1 - Pr(t|d)$.

Thus, the robustness if a rule, given a particular database state, depends on the probability that a transaction that moves it into an inconsistent state will occur. In the analysis presented by Hsu and Knoblock (1995), however, only 'normal' transactions are considered and therefore only the system's robustness to natural noise in the data is examined. In this thesis, it is argued that, for open systems (in which a database is updated through system use), it is necessary to consider also how much effort is required on the behalf of external users to deliberately change the output of a system. Thus, it is not appropriate to measure the probability that the transaction will occur, rather the *cost* of such a transaction should be measured. If a transaction that renders a system inconsistent is inexpensive to perform, then it represents a major threat to the integrity of a system.

4.2.3 A Formal Definition of Robustness

In general, there are two aspects to the robustness of recommendation generation. The first is concerned with recommendation *accuracy*: are the products recommended after an attack actually liked? The second issue relates to recommendation *stability*: does a system recommend different products after an attack, regardless of whether customers actually like them?

While stability and accuracy are distinct, they are not independent. For example, if a recommender has perfect accuracy for a given task both with and without noise, then it must be perfectly stable. On other the hand, consider a product that no–one likes. The recommendation policies *recommend to no–one* and *recommend to everyone* are both perfectly stable, yet the first is always correct and the second is always wrong.

The fact that an attack could force a recommender system to change its recommendations is clearly problematic, irrespective of whether or not the biased (or unbiased) recommendations are correct. Thus, in the following definition of robustness for recommender systems, the focus is on the consistency of the recommendation process rather than the accuracy of the process.

Robustness is the ability of a system to provide stable predictions and recommendations given some degree of noise present in the data.

Note that this definition has some parallels to that given in (IEEE, 1990), which is quoted in Section 4.2.2. In the next section, this definition of robustness is translated into metrics which provide the means with which to evaluate the robustness of recommender systems.

4.2.4 Robustness Metrics

To begin, a simple example that illustrates the deficiencies of the accuracy metrics, in the context of the evaluation of system robustness, is presented. In particular, consider the mean absolute error (MAE) metric, which is representative of the metrics used to evaluate predictive accuracy as described in Section 3.2.3. Suppose that the true rating given by a user for a particular item on a 5–point rating sale was 3, and that the pre– and post–attack predictions that were made for this item were 2 and 4 respectively. In this scenario, MAE, which measures the absolute difference between true and predicted ratings, remains unchanged after the attack, even though the attack has succeeded in causing a significant prediction shift to occur. As this example demonstrates, accuracy metrics are not adequate to effectively evaluate system robustness. In addition, since the robustness metrics that are introduced below abstract the true item ratings from the calculations, a direct comparison of robustness across different recommender systems is facilitated. Such comparisons would not be possible with metrics that depend on system accuracy considerations, given that different systems are unlikely to provide identical performance in terms of recommendation accuracy.

4.2.4.1 Percentage of Attack Profiles in Neighbourhoods

If an attack is to be successful, attack profiles need to be present in the neighbourhoods of targeted users. Thus, as a first order measure of attack success, the percentage of attack profiles that are contained in neighbourhoods is reported. While the presence of attack profiles in neighbourhoods is a prerequisite for attack success, it is not, however, a guarantee of attack success. Unless the attack profiles are sufficiently well constructed to achieve the desired attack intent (i.e. product push, product nuke, etc.), and cause significant prediction shifts, then the mere presence of such profiles in neighbourhoods may be of little consequence.

4.2.4.2 Mean Absolute Prediction Error

In order to measure the prediction shift that has been achieved by an attack, the mean absolute prediction error (MAPE) metric is introduced. MAPE is defined as the absolute difference between the pre– and post–attack predictions. Let A_j be the set of users over which an attack on item j is evaluated. The MAPE for item j is calculated as:

$$\text{MAPE}_j = \frac{1}{|A_j|} \sum_{a \in A_j} |p'_{a,j} - p_{a,j}| \tag{4.1}$$

where $p_{a,j}$ and $p'_{a,j}$ are the pre– and post–attack predicted ratings, respectively. The overall robustness of a system can then be calculated by taking the average MAPE over

all targeted items. A low value of MAPE indicates that a system is robust against a particular attack, while high values indicate successful attacks. Note that, with this metric, the prediction shift evident in the example presented above is captured, giving the desired result of 2.

In addition, the *direction* of prediction shifts resulting from attacks are reported, i.e. whether the shifts achieved are positive or negative. For example, the goal of product push attacks is to increase the predictions that are made for targeted items, and thus, for successful attacks, positive shifts are required. Likewise, for product nuke attacks, the goal is to achieve negative shifts.

4.2.4.3 Normalised Mean Absolute Prediction Error

To facilitate a comparison between the robustness of systems that operate on different rating scales, normalised mean absolute prediction error (NMAPE) is defined, in a manner that is analogous to normalised mean absolute error, as follows:

$$\text{NMAPE}_j = \frac{\text{MAPE}_j}{r_{max} - r_{min}} \tag{4.2}$$

where r_{min} and r_{max} are the minimum and maximum ratings that are defined on a particular rating scale, respectively. This metric results in a value of 0.5 when applied to the above example, indicating that the prediction has been shifted half–way along the rating scale.

4.2.4.4 Percentage of *Good* and *Bad* Predictions

The MAPE metric described above can be viewed as a general measure of system robustness, in that prediction shifts of equal amount are treated as being of equal importance. While it is important to capture all the prediction shifts that are achieved by an attack, some prediction shifts are intuitively of greater significance than others when it comes to influencing users' behaviour. Consider, for example, a system that operates on a five–point rating scale, wherein a shift from 3 to 4 is likely to be more meaningful to users than, say, a shift from 1 to 2. The probability of users *acting* on a recommendation with a low predicted rating, be it a 1 or a 2, is presumably small. However, at the upper end of the scale, any positive shifts achieved are likely to be of much greater significance.

Motivated by the above discussion, and by the accuracy and ROC metrics described in Section 3.2.3, two further metrics are now proposed. In the first metric, the increase in the number of *good* predictions that are made for targeted items after an attack are calculated. A good prediction is defined as either the best or second–best rating on a particular scale. This is a suitable metric for evaluating the effects of a product push

attack, in which the goal is increase in the number of good predictions that are made. In a similar manner, the increase in the number of *bad* predictions can be used to evaluate product nuke attacks, where a bad prediction is defined as any rating below the midpoint of the rating scale in question.[2]

These metrics represent a useful measure of attack success because they allow a direct interpretation of user behaviour. If it is assumed that some proportion of users act on good and bad predictions in a corresponding manner (i.e. purchase/do not purchase an item), these metrics can be used as the basis with which to construct a cost–benefit analysis for the various attacks that are considered in this thesis. This matter is explored further in Chapter 5.

4.3 Summary

In this chapter, the concept of robustness as an additional performance measure for recommender systems has been introduced. Given the security implications that are typical for many recommender systems, the importance and relevance of such a measure has been underlined. Following a review of related work in the field of knowledge based systems, a definition of robustness has been proposed and new metrics have been developed to evaluate the robustness of recommender systems. In the next chapter, attack strategies that are designed to influence the predictions that are made by recommender systems are described.

[2]While definitions of what constitute good and bad predictions are a subjective matter, those presented above are, in the author's opinion, reasonable and suitable for general analysis.

Chapter 5

Attack Strategies

5.1 Introduction

In this chapter, several attack strategies are developed and the effect of each on system robustness is evaluated. We show that our attack strategies outperform those that are described in related work and succeed in substantially biasing the predictions that are made for targeted items.

The effect of item characteristics on robustness is also examined, along with the various extensions to memory–based collaborative filtering algorithms that are described in Section 2.2.5. Further, the relative strengths and weaknesses of the similarity metrics and neighbourhood formation schemes as outlined in Sections 2.2.2 and 2.2.3 are evaluated, and we show that no combination of these achieves an acceptable degree of robustness against attack when systems are configured to provide good performance across other dimensions.

To conclude, a cost–benefit analysis is presented in order to determine what financial rewards are to be gained from mounting attacks. We show that very significant financial benefits can indeed be realised by attacks, thereby underlining the importance of a robustness measure for recommender systems.

5.2 Experimental Datasets

Three real–world datasets have been used to evaluate the attacks as developed in subsequent sections. The first is the EachMovie system dataset that is provided by Digital Equipment Corporation's Systems Research Center[1]. This movie recommender system operated between 1995 and 1997. The original dataset has some $72,916$ users who entered a total of $2,811,983$ numeric ratings for $1,628$ different movies. From this, a random sample of $1,000$ users, containing $63,507$ transactions on a 6–point rating scale of 1 to 6 inclusive (modified from the original 6–point scale of 0 to 1) is selected.

[1] `http://www.research.digital.com/SRC/eachmovie/`

Table 5.1: Details of the experimental datasets that are used to evaluate the robustness of collaborative recommender systems.

Dataset	#Users	#Items	#Ratings	Rating Scale	Sparseness
Smart Radio	63	1,055	4,075	1–5	93.9%
MovieLens	943	1,682	100,000	1–5	93.7%
EachMovie	1,000	1,628	63,507	1–6	96.1%

The second dataset used is that provided by the MovieLens project[2], a web–based research recommender system that debuted in 1997. It consists of 943 users, 1,682 movies and contains 100,000 transactions in total. The final dataset that is used is obtained from the Smart Radio system, which is a personalised web–based music recommender service (Hayes *et al.*, 2002). This dataset is provided by the Department of Computer Science, Trinity College Dublin, and contains 4,075 ratings by 63 users on 1,055 songs. The rating scale is from 1 to 5. Table 5.1 describes the datasets that are used in our evaluation.

The experimental procedure that is adopted in all cases is as follows. The performance of a particular item that is subjected to attack is evaluated over all the genuine users in a system who have rated the item in question. For each of these users, an *all but one* protocol is adopted in which the test set is formed by removing the user–item pair in question from the database, and a prediction for this pair is made using all remaining data. The average result across all users is then computed.

In all the results that are shown in this thesis, unless stated otherwise, the average performance that is achieved across all the items that are contained in a system is presented. Average performance is calculated by repeating the above procedure for all items that are present in a system, and taking the mean of the results obtained.

For the k–NN neighbourhood formation scheme, the optimal neighbourhood size was chosen by experiment in the standard manner, with k set to 35 for MovieLens, 45 for EachMovie and 14 for Smart Radio. See Appendix B for details on the neighbourhood size selection process. Where used, the significance weighting and case amplification parameters N and ρ are set to 50 and 2.5, respectively, as per Breese *et al.* (1998) and Herlocker *et al.* (1999).

5.3 Attacks

Two different kinds of attack are considered – namely *product push* and *product nuke* attacks. The goal of the these attacks is to force the predicted rating of an item or group

[2] `http://movielens.umn.edu/`

of items to specific ratings. For example, in a product nuke attack, the goal is to force the predicted ratings of targeted items to the minimum rating. Attackers are assumed to have no direct access to a system database, and attacks which are implemented by inserting attack profiles through the normal user interface are considered only.

Clearly, any attack profiles added need to be sufficiently close or similar to the target users if they are to have an influence on predicted ratings. In this regard, the choice of similarity metric is important. For the Cosine metric, given the normalisation that is applied in (2.1), the similarity between attack and targeted users is, in general, likely to increase with increasing attack profile size, assuming that the number of common items between attack profiles and targeted users remains high. In addition, Cosine similarity is influenced by whether or not common items between users were 'liked' – the higher the ratings given for these items the higher the similarity will be. For the Pearson and Spearman rank correlation metrics, similarity is calculated only over those items that both users have in common, and depends on how well users have 'agreed' on the items in question, rather than on the number of such items. Thus, to ensure a successful attack regardless of the similarity metric that is used by a system, the formulation of attack user profiles will need to take all of the above considerations into account.

From (2.7), the dependence of a prediction on a database D may be written as follows:

$$p_{a,j}(D) = \bar{v}_a(a) + \sigma_{a,j}(D)\,. \tag{5.1}$$

Since the first term depends entirely on the active user, the insertion of attack profiles into a system can only affect the deviation from user mean term, $\sigma_{a,j}(D)$. Therefore, for successful attacks, it is necessary to maximise the magnitude of the deviation term. The contribution to this term by any particular neighbour is a function of:

(a) the rating given to the item for which a prediction is being sought,

(b) the neighbour's mean rating, and

(c) the similarity between the neighbour and the targeted user.

Thus, careful selection of the items and ratings that are used to build attack profiles is required. An optimal ratings strategy for attack profile items is developed in subsequent sections. The item selection strategy that will be used in many of the attacks involves the use of *popular* items – i.e. items which are rated by many users in a system. There are a number of advantages in choosing popular items to build attack profiles:

(a) the likelihood of a high ratio of the number of co–rated items between genuine and attack profiles to attack profile size is increased, which is important to ensure large similarities between genuine and attack profiles, especially when the significance weighting algorithm extension is used,

(b) it is desirable that each attack profile have a high probability of being located in the neighbourhood of many targeted users in order to minimise the number of attack profiles that need to be created, and thereby reduce the cost of attack, and

(c) popular items tend to receive consistent and predictably high ratings from users.

From an attacker's perspective, it is not unreasonable to suggest that the formulation of such attack profiles is feasible, given that the strategy relies on global characteristics of a system which are generally known or easy to estimate.

For product push and nuke attacks, in order to successfully force predicted ratings to particular values, it is necessary to ensure that the magnitude of the deviation from mean term, $\sigma_{a,j}(D)$, is maximised and minimised, respectively. Once more, the choice of similarity measure, along with the rating scale that is used by a system, is important in this regard. From the analysis presented in Section 2.2.2, Cosine similarity always results in a value between 0 and 1 for systems that operate on positive rating scales. Thus, as it will be shown below, it is straightforward to ensure that all attack profiles contribute to maximising or minimising the deviation from mean term in (5.1). However, Pearson and Spearman rank correlations give values between -1 and 1 irrespective of the rating scale that is used and, therefore, care needs to be taken if all attack profiles are to correlate either positively or negatively with targeted users. In the following sections, strategies for product push and nuke attacks, which take all of the above considerations into account, are developed.

5.4 Developing an Attack Strategy: Product Push Attack

To begin, consider a product push attack mounted on a system using the k–NN neighbourhood formation scheme. The basic strategy for the attacker is to create an attack profile which correlates strongly and positively with the ratings of a set of genuine users in the system, and to add to this the maximum rating for the pushed item. Alternatively, the attacker might create a profile which correlates strongly and negatively with the target group and add to this the minimum rating for the target item. A uniform push across the entire database, however, seems problematic. Assuming that the database is a random sample of the user population, there seems no criterion for selecting the item values in the new profiles, other than to set them randomly.

5.4.1 Random Push Attack

In this first attack strategy, the attacker creates attack profiles by selecting a number $l < m$ randomly (where m is the total number of items in the system), selects l items at random and rates them randomly. Then the maximum rating for the pushed item is added to the newly created profile. However, this strategy is doomed to failure. With high probability the correlation of two independently chosen random profiles is close to zero. Hence the contribution of a random attack profile to the predicted rating of a random profile in the database is likely to be insignificant.

The following experiment was carried out using the MovieLens dataset, where 64 attack profiles of size 100, created using the above strategy, were inserted into the system. Results are presented in Table 5.2 for the three similarity metrics evaluated. For Pearson correlation ('Pe'), the results indicate that the attack was not successful. Only 59% of all shifts achieved were positive as required for a successful product push attack, which means that in 41% of cases, the opposite effect (or the nuking of predictions) was achieved. In terms of the robustness metrics described in Section 4.2.4, 70% of all profiles, on average, which were present in neighbourhoods were attack profiles, which is certainly high. This finding did not have a significant effect on predictions, however, with an MAPE of only 0.4377 achieved. (Recall that MAPE measures the absolute difference between pre– and post–attack predictions.) As stated above, randomly created attack profiles were as likely to correlate positively as they were negatively with genuine users, and thus, the contributions of multiple attack profiles in individual neighbourhoods were largely canceled out, causing only relatively small prediction shifts to occur.

The lack of attack success is further confirmed when the percentage of good predictions that were achieved pre– and post–attack are examined. Recall from Section 4.2.4 that a good prediction is defined as either the best or second–best rating on a scale. For the MovieLens system, which operates on a discrete 5–point rating scale, any prediction ≥ 4 is defined as a good prediction. When no attack profiles were present in the system, some 18% of predictions resulted in a rating of 4 or greater being returned. Post–attack, this figure actually dropped by 1%, which indicates that the negative shifts caused more predicted ratings to drop below the second–best rating than were pushed above.

It is also interesting to examine system accuracy pre– and post–attack. For Pearson correlation, accuracy, calculated according to MAE, deteriorated from 0.7948 to 0.8596 post–attack. In order to quantify this decrease, it is useful to consider a non–personalised benchmark algorithm which simply calculates a prediction for a particular item as the average of all ratings received by that item in a system. Clearly, if collaborative filtering is to be useful, it should outperform the benchmark in terms of predictive accuracy. For MovieLens, the non–personalised approach resulted in a MAE of 0.9363, which is worse than that provided by Pearson correlation, pre– or post–attack. Thus, we can conclude

Table 5.2: Random Push Attack carried out on the MovieLens dataset. Results shown are for an attack strength of 64 attack profiles inserted into the system. Accuracy (MAE) computed according to the non–personalised benchmark algorithm is 0.9363. Any MAE values that exceed this baseline are shown in boldface.

	% Pos. Shifts	% Attack Profiles in Neighbourhoods	MAPE	% Good Preds. (pre)	% Good Preds. (post)	MAE (pre)	MAE (post)
Pe	59	70	0.4377	18	17	0.7948	0.8596
Sp	59	70	0.4385	18	17	0.7965	0.8608
Co	100	35	0.5881	17	38	0.7975	**1.0533**

the Pearson correlation similarity metric was robust to this particular form of attack.

Regarding Spearman Rank ('Sp') correlation, it is clear from the results that this similarity metric provided very similar performance, across all dimensions evaluated, to Pearson correlation. As stated in Section 2.2.2, this finding was to be expected since the ratings that are used to rank items are, in essence, ranks. Thus, a similar analysis and conclusion to that above also applies to this particular similarity metric.

Since MovieLens operates on a positive rating scale and given the formulation of the attack profiles, Cosine similarity ('Co') can only result in positive prediction shifts (as required for successful product push attacks). Attack profiles accounted for on average 35% of all neighbours, significantly less than for the correlation metrics. Given that average neighbourhood sizes pre– and post–attack were 23 and 35 respectively, this represents only a marginal improvement on the minimum percentage expected given that neighbourhood sizes for the k–NN neighbourhood formation scheme were fixed at 35. Since all prediction shifts were positive, the MAPE achieved was higher than than for the correlation metrics, at 0.5881. In addition, the number of good predictions was significantly increased from 17% to 38% post–attack. Accuracy post–attack was also poor – in this case the accuracy provided by the system was worst than that of the benchmark non–personalised algorithm. Thus, we can conclude that Cosine similarity was not robust to this particular random attack, which is a cause for concern given the simplicity of the attack strategy employed.

5.4.2 Focused Push Attack

In this section, an attack strategy focusing on the correlation similarity metrics is developed. In this regard, an important weakness in the correlation formulae from the point–of–view of robustness is exploited to develop attack profiles. The particular weakness is that correlations are calculated only on the items which *both* users have rated. Hence, users can correlate strongly even though they have few items in common. While this

weakness has been noted from a predictive accuracy point–of–view and a suitable extension to collaborative filtering algorithms has accordingly been proposed (i.e. significance weighting), we highlight here its implications in the context of system robustness.

With the correlation formulae, the correlation between users who have *exactly* two items in common is always $+1$, -1 or 0. If the attacker adopts a strategy of building attack profiles consisting of the item to be pushed together with two other carefully selected items, then the probability of a significant number of attack profiles appearing in neighbourhoods is high, leading to a successful attack. Ideally, two items about which there is strong consensus in the user population are selected to generate the attack profiles.

Consider, for example, an active user a and an attack profile f, where both users have rated only two common items, j and k. Using Pearson correlation (a similar analysis applies for Spearman rank correlation), the weight between users a and f becomes

$$w(a,f) = \frac{(v_{a,j} - \bar{v}_a)(v_{f,j} - \bar{v}_f) + (v_{a,k} - \bar{v}_a)(v_{f,k} - \bar{v}_f)}{\sigma_a \sigma_f}\,, \tag{5.2}$$

where $\bar{v}_a = 0.5(v_{a,j} + v_{a,k})$ and $\bar{v}_f = 0.5(v_{f,j} + v_{f,k})$. Hence, if $(v_{a,j} - \bar{v}_a)$ is positive then $(v_{a,k} - \bar{v}_a)$ must be negative, and similarly for user f. Knowing the signs of the terms for the active user, it is possible to select the ratings for the attack user profile, so that both products are positive (similarly negative). Thus, by manipulating the ratings assigned to the attack profile, an attacker can achieve positive or negative correlations between the active and attack profiles and thereby implement a product nuke or a product push attack as desired.

The remaining difficulty is that the strategy depends on knowing how the genuine users have actually assigned ratings to items. Of course, this knowledge is unknown to an attacker unless complete access to a system's database is available. However, some simple heuristics can suffice in practice. One possible assumption is that items that have received many ratings in a system (i.e. popular items) generally receive high ratings while less popular items receive lower ratings.

Thus, in this attack, we adopt the strategy of 3 item attack profiles, consisting of the item to be pushed which is set to the maximum rating, r_{max}, together with the two most popular items in the system. These latter items are given ratings according to the following rule: the less popular of the two items is assigned the minimum allowed rating, r_{min}, and the other item is assigned a rating of $r_{min}+1$.[3] This particular ratings scheme is optimal in the sense that it ensures that the difference term $(r_{i,j} - \bar{r}_i)$ in (2.7) is maximised for the attack profiles, thereby maximising the prediction shifts that are achieved by the attack. The identical attack profile is added a number of times. Since this attack can

[3]Note that if these ratings are reversed for the genuine users that are contained in the system, the result will be a product nuke attack instead of the desired product push!

Table 5.3: Focused Push Attack carried out on the MovieLens dataset. Results shown are for an attack strength of 64 attack profiles inserted into the system. Accuracy (MAE) computed according to the non–personalised benchmark algorithm is 0.9363. Any MAE values that exceed this baseline are shown in boldface.

	% Pos. Shifts	% Attack Profiles in Neighbourhoods	MAPE	% Good Preds. (pre)	% Good Preds. (post)	MAE (pre)	MAE (post)
Pe	85	73	1.7982	15	64	0.7721	**2.0265**
Sp	85	73	1.7982	15	64	0.7730	**2.0279**
Co	100	33	0.4075	15	30	0.7738	0.9241
Pe+sig	91	27	0.4135	16	30	0.7675	**0.9568**
Sp+sig	91	27	0.4129	16	30	0.7683	**0.9577**
Co+sig	100	33	0.0584	15	16	0.7753	0.7754

only be successful against those users in the database that have themselves rated the two most popular items (for example, 35% of all users for the MovieLens dataset), average results are presented over only those users who have rated both of the items in question.

Table 5.3 shows the results for the MovieLens system for an attack strength of 64 attack profiles inserted into the system. As before, the results obtained using the Pearson and Spearman rank correlation metrics were closely matched. For these metrics, the attack strategy was successful in achieving a large percentage of positive shifts – 85% of all shifts being positive – as is necessary for push attacks. Because of this, the large number (73%) of attack profiles that were present in neighbourhoods post–attack caused significant shifts to occur, as evidenced by the high MAPE values of 1.7982. Since MovieLens operates on a rating scale of 1 to 5, such shifts are sufficient to change predictions from 'dislike' to 'like'. The number of good predictions post–attack also increased dramatically, with 64% of all predictions equalling (or exceeding) the second–best rating on the scale, compared to a baseline of 15% pre–attack. Regarding accuracy, the post–attack MAE was significantly higher (i.e. worse) than that of the non–personalised algorithm. Thus, in all respects, it can be concluded that the correlation metrics were not at all robust against this particular attack strategy.

Due to the small attack profile size, the attack is less successful when using Cosine similarity. Given that the minimum profile size of genuine users in the dataset is 20, attack profiles are less likely to dominate the prediction shift when Cosine is used. Nevertheless, an MAPE of 0.4075 was achieved and the percentage of good predictions increased from 15% to 30% as a result of the attack.

5.4.2.1 Significance Weighting

The results from the same attack are also shown when the significance weighting extension (labeled '+sig') was included in the algorithm. Although the importance of significance weighting from the point–of–view of predictive accuracy has already been established for collaborative filtering algorithms, these results also highlight its importance in the context of system robustness. As can be seen, this modification had a significant impact on the effectiveness of the attack. In general, the system was much more robust as the influence on predictions of the small–sized attack profiles was reduced. However, robustness remained an issue for the correlation metrics with the percentage of good predictions almost doubling as a result of the attack. Thus, it can be concluded that, while significance weighting was useful in reducing attack success and should be included in the collaborative filtering algorithm, it did not provide complete robustness against this form of attack. In all future experiments, significance weighting is included in algorithms, unless stated otherwise.

5.4.3 Large Attack Profiles

In this section, the attack strategy as outlined above is expanded. Given the small–sized attack profiles that were used, the above strategy was successful in targeting only a small number of genuine users in the system. It is clear that if attacks are to be successful against a greater number of users, then larger attack profiles are required. In addition, from an attacker's perspective, larger attack profiles will remove the degree of protection afforded the algorithm by the significance weighting extension. As before, an attacker must ensure that all attack profiles created correlate in the same direction with the genuine users in a system when the correlation similarity metrics are used in the algorithm.

Thus, along with the item to be pushed, one solution is to construct attack profiles from two 'groups' of items. The first group consists of items that are generally rated higher than average in the database (i.e. *liked* items), and the second group consists of items that are generally rated lower than average (i.e. *disliked* items). By assigning a high rating to the liked items and a lower rating to the disliked items, an attacker can be confident that these attack profiles will correlate positively with the majority of genuine users in the database.

The difficulty with this approach is, of course, in correctly identifying such groups of items in a particular domain. Although some degree of error must be expected on the part of an attacker in this regard, it remains, nevertheless, a feasible proposition to select such items in many real–world situations. Consider, for example, movie or music domains, where such knowledge is possible to estimate or to mine from various sources.

In this section, it is assumed that an attacker has both *item popularity* and *item likability* knowledge for all the items that are contained in the system being attacked. Item popularity refers to the number of ratings for each item that are present in a system and item likability refers to the mean rating received by each item. (In Section 5.4.5, an analysis is presented for situations where such knowledge is unavailable, and the consequences for attack success are evaluated.) An item is included in the liked group if the average rating received for that item exceeds the mean item likability, otherwise it is included in the disliked group. Attack profiles are then constructed from the item being pushed, along with the most popular items drawn in equal quantities from each of the groups.

The following ratings are assigned to attack profile items. The minimum rating, r_{min} is assigned to each of the disliked items, ratings of $r_{min} + 1$ are assigned to the liked items and the item to be pushed is assigned the maximum rating, r_{max}. As before, this ratings scheme is optimal since it maximises the difference term $(r_{i,j} - \bar{r}_i)$ in (2.7) is for the attack profiles, as well as ensuring a high number of positive correlations between genuine and attack profiles.

For Cosine similarity, the attack strategy is more straightforward since all datasets evaluated operate on positive rating scales. Thus, the attacker can simply choose the l most popular items (assuming this knowledge is available to the attacker) and, to allow for some variability, ratings of r_{min} or $r_{min} + 1$ are assigned to each. If the item being pushed is assigned the maximum rating, these attack profiles meet all the criteria for a maximal push attack.

5.4.3.1 Results

Results are shown in Table 5.4 for an attack strength of 64 attack profiles for the MovieLens, EachMovie and Smart Radio datasets. The results presented are average performance over all users and items in the respective datasets. To facilitate a direct comparison between the datasets, normalised values for MAPE and MAE are presented. For the correlation similarity metrics, the majority of the prediction shifts achieved across the datasets were positive (upward of 95%), as necessary for a successful push attack. In addition, all the robustness metrics indicated that a very successful attack took place, with significant prediction shifts occurring and large increases in the percentage of good predictions resulting from the attack. Consider, for example, the EachMovie dataset (Pearson correlation), where the attack resulted in a normalised MAPE of 0.7024 and where the number of good predictions increased greatly from 24% to 89%. In all cases, accuracy performance post–attack had deteriorated well beyond that provided by the non–personalised algorithm. As before, the performance offered by both Pearson and Spearman rank correlations were very similar with little to distinguish between the results.

For the MovieLens dataset, the attack was less successful when Cosine similarity with

significance weighting was used. For example, normalised MAPE was 0.4553 and the percentage of post–attack good predictions that was achieved was 73%, compared to values of 0.6716 and 87% for Pearson correlation. These results were, nonetheless, significant, and clearly show that Cosine similarity is not secure against the form of attack as outlined above. For the other datasets, the performance indicators were more closely matched to those of the correlation metrics. In addition, the attack caused the accuracy provided by all systems to be significantly worse than that of the non–personalised algorithm.

Table 5.4: Focused Push Attack (large attack profiles) carried out on the three datasets. Results shown are for an attack strength of 64 attack profiles inserted into datasets. Accuracy (normalised MAE) computed according to the non–personalised benchmark algorithm is 0.2341 (MovieLens), 0.1986 (EachMovie) and 0.2689 (Smart Radio). Any normalised MAE values that exceed this baseline are shown in boldface.

	% Pos. Shifts	% Attack Profiles in Neighbourhoods	NMAPE	% Good Preds. (pre)	% Good Preds. (post)	NMAE (pre)	NMAE (post)
MovieLens							
Pe+sig	98	75	0.6716	19	87	0.1969	**0.7149**
Sp+sig	98	75	0.6699	19	87	0.1974	**0.7145**
Co+sig	100	51	0.4553	17	73	0.1997	**0.5149**
EachMovie							
Pe+sig	96	81	0.7024	24	89	0.1798	**0.7239**
Sp+sig	96	82	0.7046	24	89	0.1788	**0.7271**
Co+sig	100	74	0.6127	23	92	0.1812	**0.6295**
Smart Radio							
Pe+sig	97	93	0.7567	41	96	0.2465	**0.7811**
Sp+sig	95	93	0.7552	41	94	0.2467	**0.7790**
Co+sig	100	87	0.7313	41	99	0.2566	**0.7400**

5.4.3.2 Discussion

In order to analyse the above results in greater detail, consider the MovieLens dataset when the Pearson correlation similarity metric with significance weighting was used. Recall from Section 5.3 that the only effect that the attack profiles can have is on the deviation from mean term, $\sigma_{a,j}(D)$, in (5.1). In Figure 5.1, a histogram showing the distributions of the deviation terms pre– and post–attack is presented. It is clear that the attack has had a significant impact, where, before the attack, the distribution mean was 0.06, with only a relatively small bias in the number of non–zero positive values (38% of values were > 0). Post–attack, however, the distribution mean was 1.8818, with 82% of

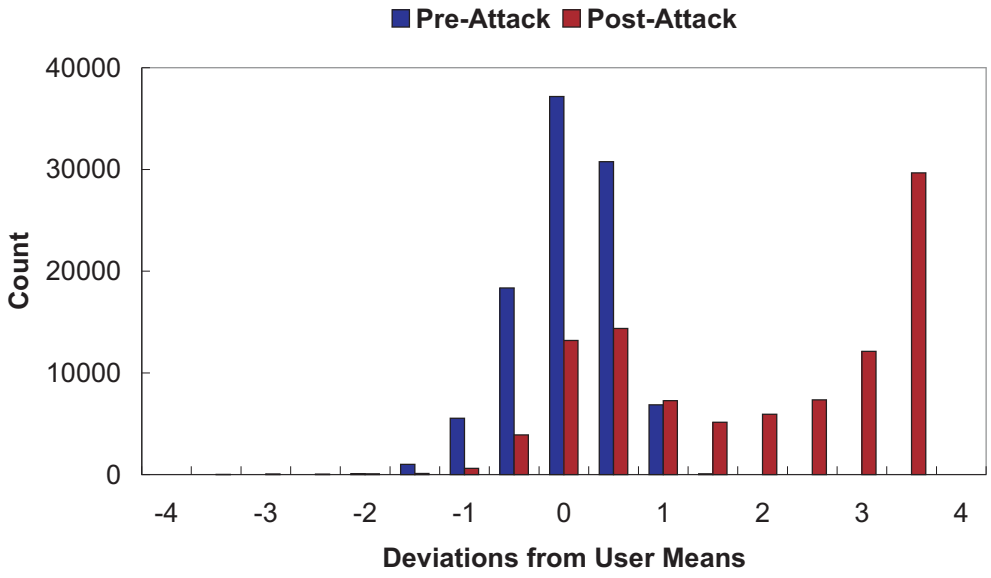


Figure 5.1: A histogram showing the distributions of the deviations from user means pre– and post–attack for Pearson correlation with significance weighting (MovieLens dataset)

the deviation terms > 0. (Note that positive values are required for a successful push attack.) Furthermore, a total of $13,134$ (or 13%) of all post–attack deviations were equal to 3.47, which was the maximum value that the attack could achieve[4].

The difference in the above distributions gave rise to many predictions being increased (pushed) by the attack. In Figure 5.2, a histogram of pre– and post–attack predictions is presented. It is apparent that a radical shift in predictions has occurred, where the mean predicted rating was shifted from 3.59 to the maximum rating of 5 by the attack. In addition, the number of predictions pre–attack equal to the maximum rating was $8,421$ (or 8% of all predictions), while post–attack, this number was $66,621$ (67%). These results certainly indicate the significant effect of the attack and highlight the lack of robustness exhibited by the system.

Figures 5.3 and 5.4 show normalised MAPE and the percentage of good predictions plotted against attack strength for the three datasets. These results are for the Pearson correlation similarity metric with significance weighting – since almost identical results were obtained for Spearman rank correlation with significance weighting, they are not presented. Attack strength is modeled by the number of attack profiles inserted into a system. Similar

[4]The maximum deviation is calculated for the MovieLens dataset as follows, where the maximum and minimum ratings are 5 and 1 respectively. Assume that, post–attack, the neighbourhood is comprised entirely of attack profiles. Each attack profile contains 100 items, with the item being pushed set to 5, and the remaining items set to ratings of 1 (50x) and 2 (49x), giving a mean rating of 1.53. Thus, the the difference term $(r_{i,j} - \bar{r}_i)$ in (2.7) is maximised to $5 - 1.53 = 3.47$. Since the neighbourhood is formed exclusively by attack profiles, the deviation from mean term is also equal to 3.47.

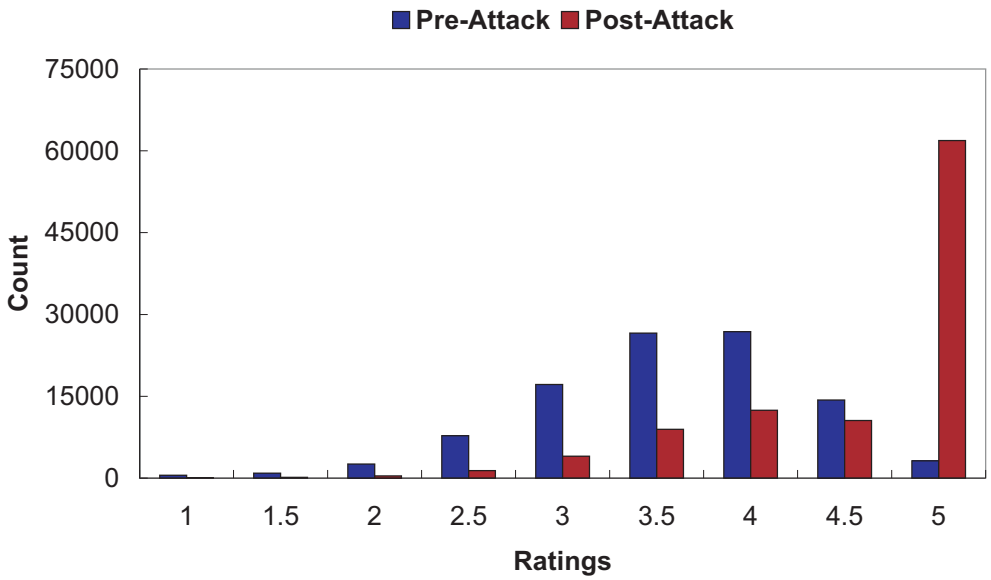


Figure 5.2: A histogram showing the distributions of predictions pre– and post–attack for Pearson correlation with significance weighting (MovieLens dataset)

trends were observed for the MovieLens and EachMovie datasets, while higher values were obtained for Smart Radio. These results were expected since Smart Radio contains significantly fewer users than the other datasets, and thus, for a given attack strength, the effect of the attack was increased. Similar trends are evident in both figures, with system robustness deteriorating as attack strength increased. From Figure 5.3, it is apparent that no further increases in normalised MAPE were observed for all datasets beyond certain attack strengths. These points correspond to the maximum neighbourhood sizes, k, set for the datasets (35, 45 and 14 for MovieLens, EachMovie and Smart Radio, respectively). In the attacks, the same attack profile was used repeatably, and thus, no additional effect could be achieved by inserting more than k attack profiles into the systems.

When the percentage of good predictions are examined, it can be seen that the most significant increases occurred at the lower attack strengths. For the MovieLens and EachMovie datasets, there was little additional gain in inserting more than 16 attack profiles into the systems, at which point the percentage of good predictions achieved for both datasets was approximately 86%. For Smart Radio, an attack strength of 4 was sufficient to push 95% of all predictions to the best or second–best rating, thereafter only a further 1% of predictions were pushed to the upper ratings. Perhaps the best indicator of attack success is to examine the percentage increase in the number of good predictions that was achieved when **only** 1 attack profile was inserted into the systems. For example, for the EachMovie dataset, the average number of good predictions achieved across all dataset items was 46%, compared to 24% pre–attack – equivalent to a percentage increase of 93%. Similar results pertained for the other datasets.

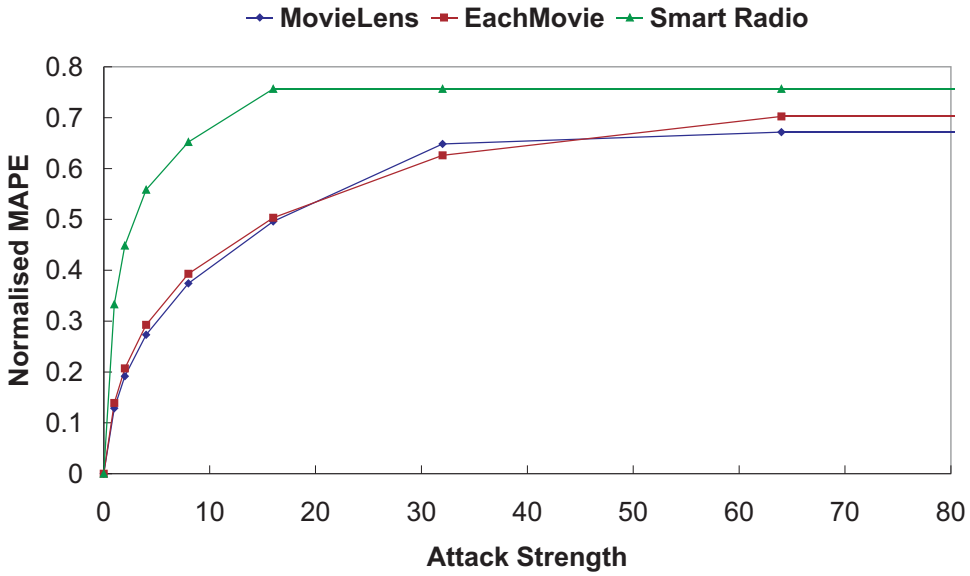


Figure 5.3: Normalised MAPE plotted against attack strength for Pearson correlation with significance weighting

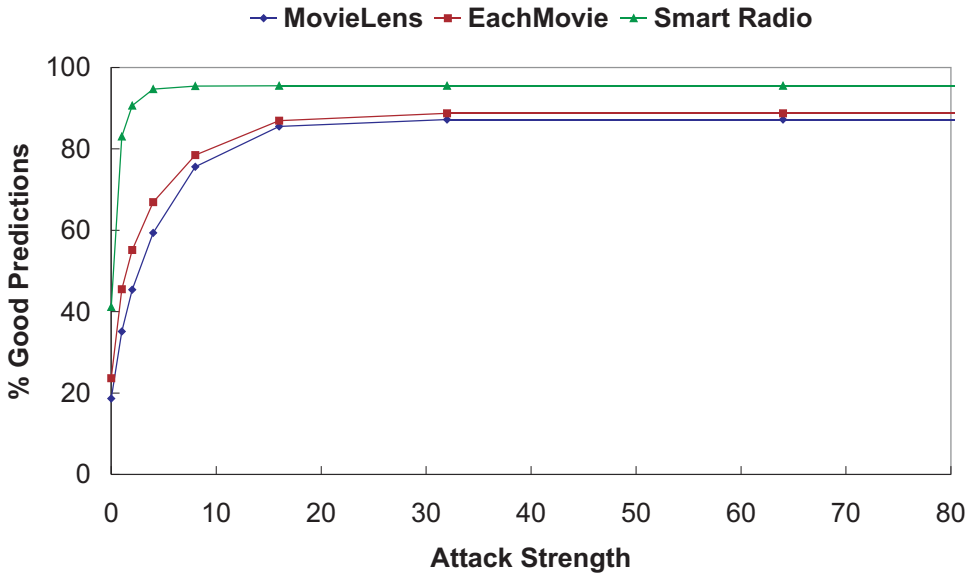


Figure 5.4: The percentage of good predictions plotted against attack strength for Pearson correlation with significance weighting

Figures 5.5 and 5.6 show the same results for Cosine similarity with significance weighting. As stated above, the attack was not as successful for the MovieLens dataset as when the correlation similarity metrics were used. Nevertheless, all the systems evaluated were significantly compromised by the attack. Consider, as before, the EachMovie dataset, where the number of good predictions increased from 23% to 40% when only 1 attack profile was inserted – a percentage increase of 74%. As before, the attack was most successful

against the Smart Radio dataset, given the relatively small number of users present in that particular system. In addition, the most significant deterioration of robustness occurred at the lowest attack strengths.

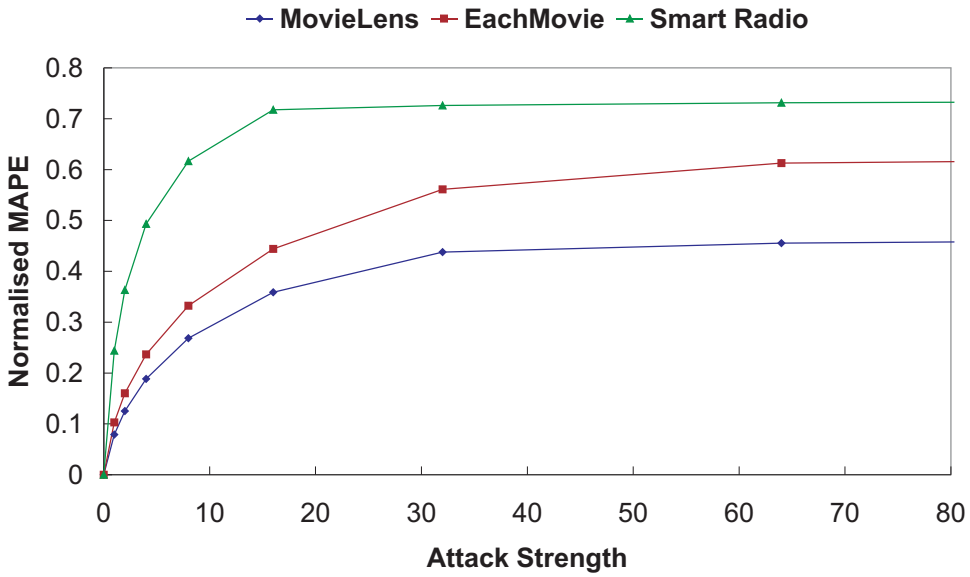


Figure 5.5: Normalised MAPE plotted against attack strength for Cosine similarity with significance weighting

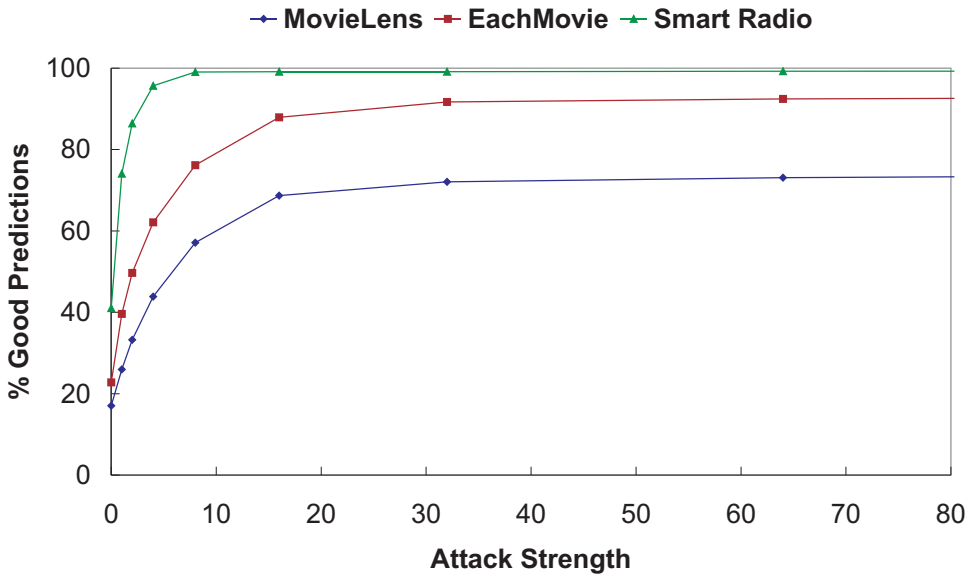


Figure 5.6: The percentage of good predictions plotted against attack strength for Cosine similarity with significance weighting

5.4.4 Effect of Item Characteristics on Robustness

The relationship between robustness and certain item characteristics is discussed in this section. In particular, using the product push attack described in Section 5.4.3, the effect of item popularity, item likability and item entropy on the robustness of a particular item is examined. Item entropy is a measure of the diversity of ratings that are received by an item.

5.4.4.1 Item Popularity

Figures 5.7 and 5.8 show the relationship between item popularity and robustness, calculated according to MAPE, for the MovieLens and EachMovie datasets respectively. In all cases, it is apparent that less popular items were more easily pushed, irrespective of similarity metric used. (Results for Spearman rank correlation were closely matched to those obtained using Pearson correlation, and thus, are not shown.) The general trend appears to be exponential in nature, with MAPE decreasing as item popularity increased. With only one exception (MovieLens using Cosine similarity), however, MAPE was generally in excess of a value of 1 even for the most popular items , which is a substantial prediction shift.

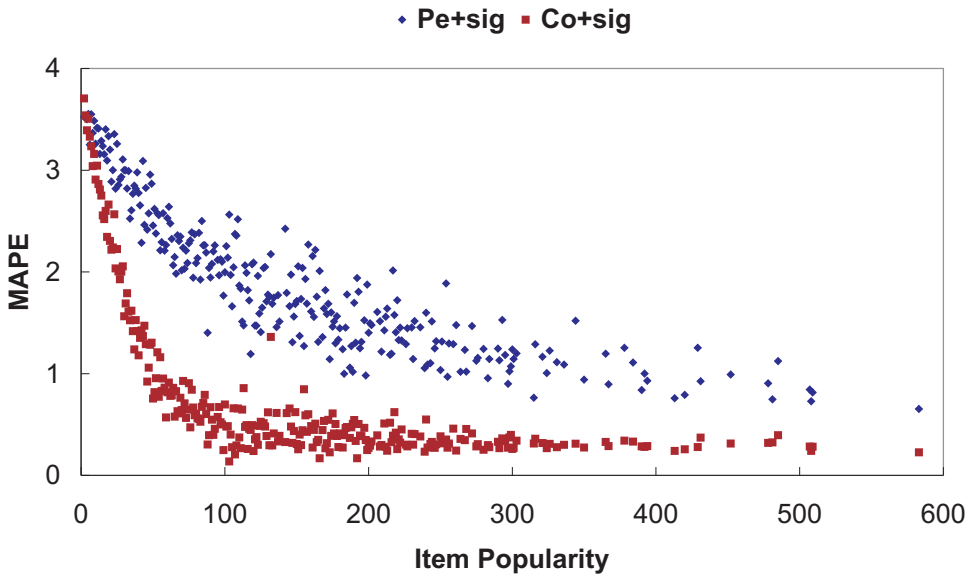


Figure 5.7: The relationship between MAPE and item popularity for the MovieLens dataset

The results for the Smart Radio dataset are presented in Figure 5.9. Given that this dataset has a relatively small number of users (63), and that the maximum item popularity is only 20, the relationship between item popularity and robustness is different in this

case and appears to be linear. Nonetheless, the same general trend applies, with system robustness improving as item popularity increased.

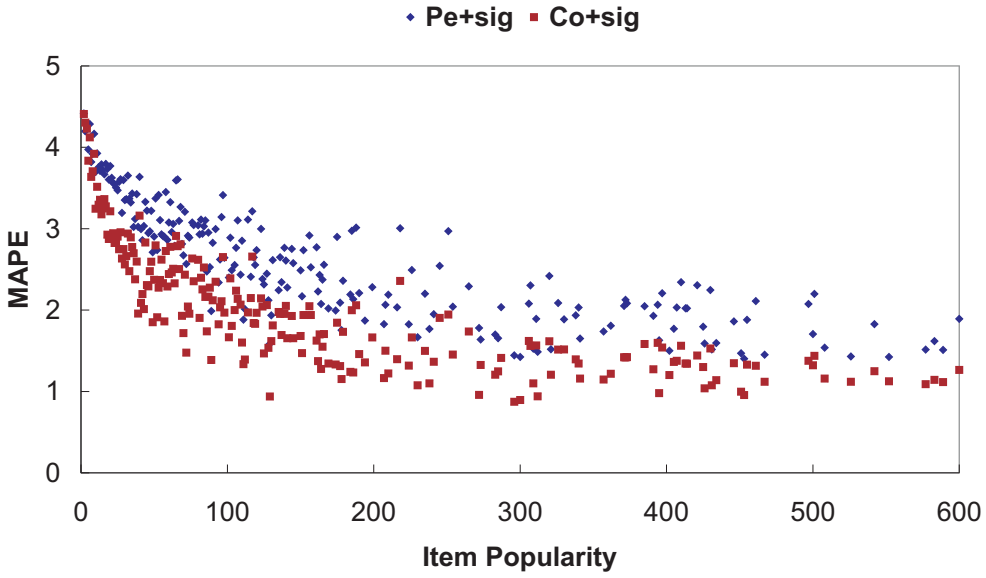


Figure 5.8: The relationship between MAPE and item popularity for the EachMovie dataset

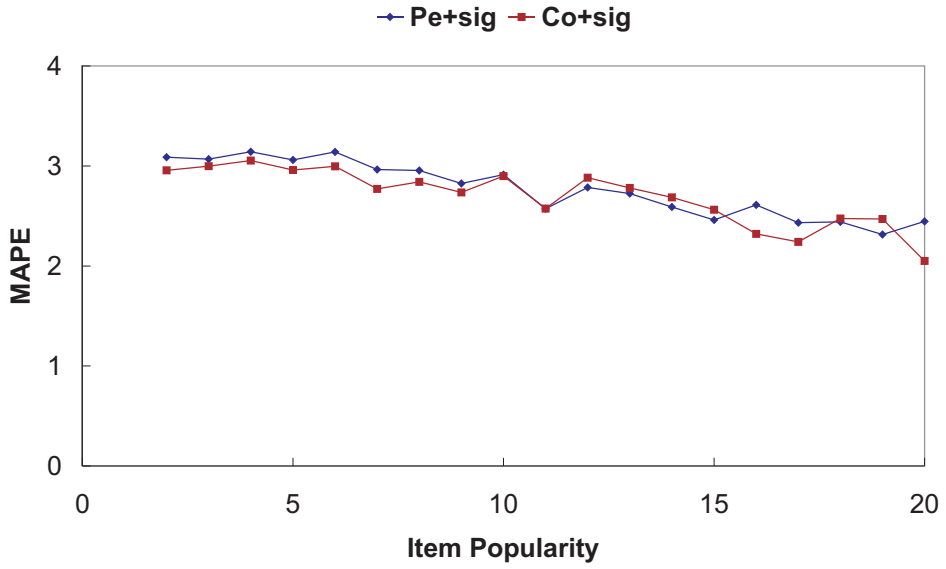


Figure 5.9: The relationship between MAPE and item popularity for the Smart Radio dataset

The above findings are intuitively correct. It is reasonable to expect that the predictions that are made for less popular items should be easier to manipulate, since the number of genuine users who have rated such items is, by definition, low. This has important

consequences for system managers, since new or recently added items are most likely to be the focus of attacks – it makes less sense for an attacker to target items about which there is a strong and established conviction. In addition, new items often sell at premium rates, providing another incentive for vested interests to further boost profits by attempting to falsely promote their products.

5.4.4.2 Item Likability

The next item characteristic that is examined is relationship between robustness and item likability. Figure 5.10 shows this relationship for the EachMovie dataset for the Pearson and Cosine similarity metrics (results obtained for Spearman rank correlation were similar to Pearson correlation, and thus, are not shown). From the results it appears that items which, on average, received lower ratings were easier to push. This relationship, however, does not appear to be particularly strong, where the correlations between robustness and item likability were $r^2 = 0.36$ and $r^2 = 0.35$ for the Pearson and Cosine metrics, respectively. The general trends for the other datasets were similar, but somewhat stronger. For MovieLens, the correlations between the variables were $r^2 = 0.69$ and $r^2 = 0.44$ for the Pearson and Cosine metrics, respectively; for Smart Radio, the corresponding correlations were $r^2 = 0.66$ and $r^2 = 0.92$, respectively. Thus, in conclusion, it can be stated that, across all datasets, items which received lower average ratings were easier to push. This finding was expected, since it seems, intuitively, that items which have received high ratings should be difficult to push, whereas poorly rated items ought to be more easily manipulated by a successful attack.

5.4.4.3 Item Entropy

The relationship between robustness and item entropy is shown in Figure 5.11 for the Smart Radio dataset. Item entropy was calculated as the standard deviation of all ratings received by an item. In this case, the results suggest that there is little or no relationship between the variables, with $r^2 = 0.08$ and $r^2 = 0.22$ for the Pearson and Cosine similarity metrics, respectively. For the MovieLens and EachMovie datasets, the maximum r^2 value was 0.07, irrespective of the similarity metric used. The relatively high r^2 value of 0.22 that was obtained for the Smart Radio dataset using Cosine similarity may be explained by the fact that only a limited amount of data was available to perform the calculation, given the small size of the dataset involved.

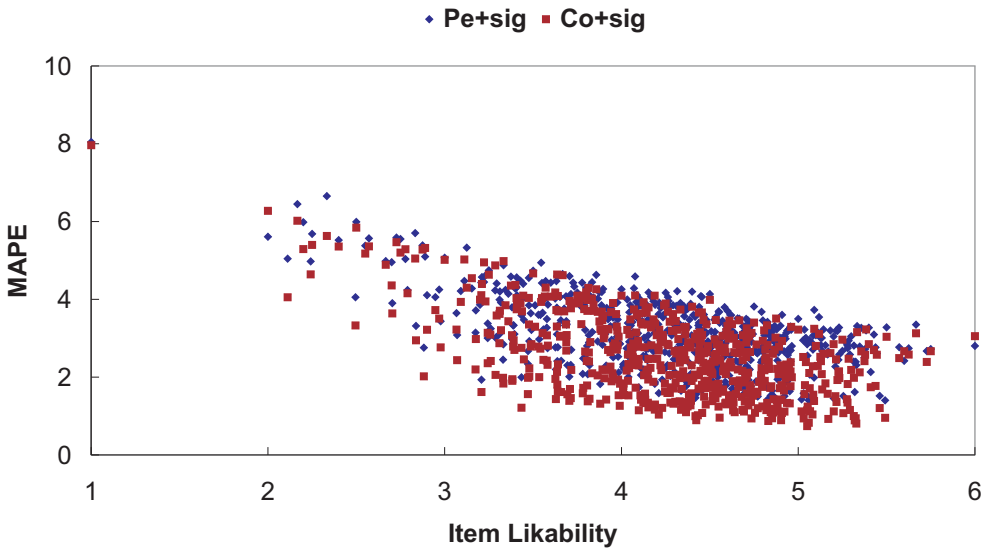


Figure 5.10: The relationship between MAPE and item likability for the EachMovie dataset

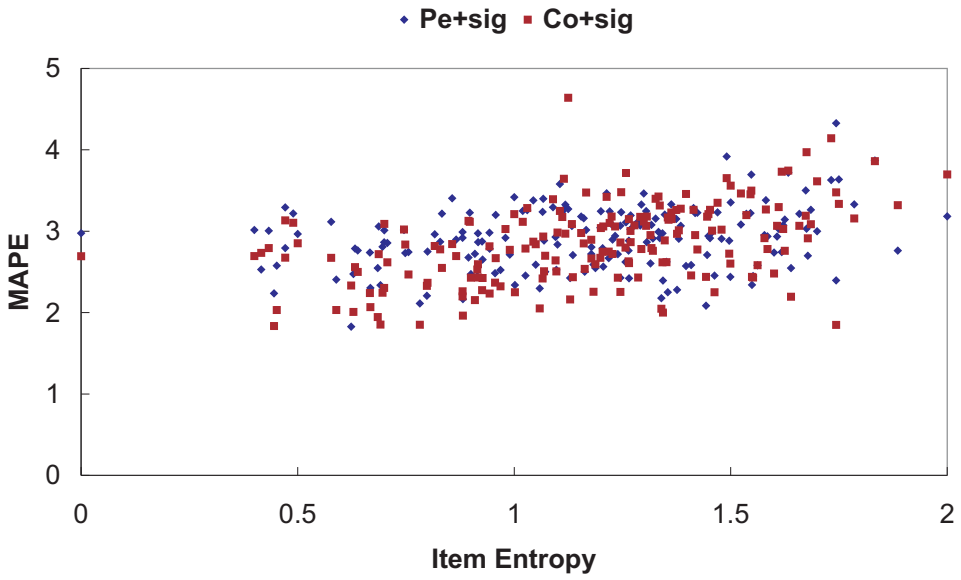


Figure 5.11: The relationship between MAPE and item entropy for the Smart Radio dataset

The results presented in this section demonstrate that all of the systems that were evaluated were vulnerable to the attacks as outlined above, irrespective of the similarity metric that was used. It is obvious that these findings ought to be a major cause for concern for system designers. Furthermore, the correlation metrics were found to be more susceptible to attack than Cosine similarity. Given that Pearson correlation is the most widely used

similarity metric for memory–based collaborative filtering algorithms, the lack of robustness that was apparent across all of the datasets that were examined clearly needs to be addressed.

5.4.5 Non–Optimal Attacks

The focused push attack of Section 5.4.3 is optimal in the sense that an attacker is assumed to have full popularity and likability knowledge for all items in a system. In this section, the effect on the success of this attack when such knowledge is only partially available is examined.

The optimal strategy for choosing items with which to create attack profiles is to select the most popular items that are contained in a system. Thus, the probability of a high degree of overlap between attack and genuine profiles is increased.

The strategy further depends on the similarity metric and the rating scale that is used. For Pearson and Spearman rank correlation metrics, to ensure positive correlations between genuine and attack profiles, item likability knowledge is also required for an optimal attack. Item likability knowledge allows the attacker to choose the most popular items from two groups of database items; where the first group contains items which are rated higher than average in a system and the second group contains items which are rated lower than average.

For Cosine similarity, the optimal strategy is more straightforward because all the systems under evaluation employ positive rating scales. Thus, the attacker simply chooses the l most popular items from a system to create attack profiles.

5.4.5.1 Item Popularity

In the analysis presented in this section, it is assumed that full item likability knowledge is known to the attacker, and that item popularity knowledge is only partially known. In order to simulate partial popularity knowledge, a number of attacks are implemented where it is assumed that only a certain percentage of the most popular items that are contained in a system are known. In each of the attacks, attack profile items are drawn from the known set of the most popular items.

As with the focused push attack of Section 5.4.3, the attack profiles that are used in all of the simulations contain 100 items and, in each case, a total of 64 attack profiles are inserted into the system. The analysis is performed using the EachMovie dataset.

Figure 5.12 shows the effect of item popularity knowledge on robustness, which is calculated according to normalised MAPE. The percentage of the most popular items from

which attack profile items are drawn is shown on the x–axis. For example, when the top–10% of the most popular items are known, attack profiles are constructed using items which are drawn from the 140 most popular items that are contained in the system[5]. Drawing from the top–100% of the most popular items simulates a complete lack of item popularity knowledge.

System robustness improved for all similarity metrics as the lack of item popularity knowledge grew. This result was to be anticipated, since the overlap between genuine and attack profiles can be expected to reduce as item popularity knowledge is decreased. Nevertheless, even when a complete lack of knowledge was simulated, normalised MAPE values only fell to 0.449, compared to 0.6996 (Pearson correlation) when the top–10% of the most popular items were known.

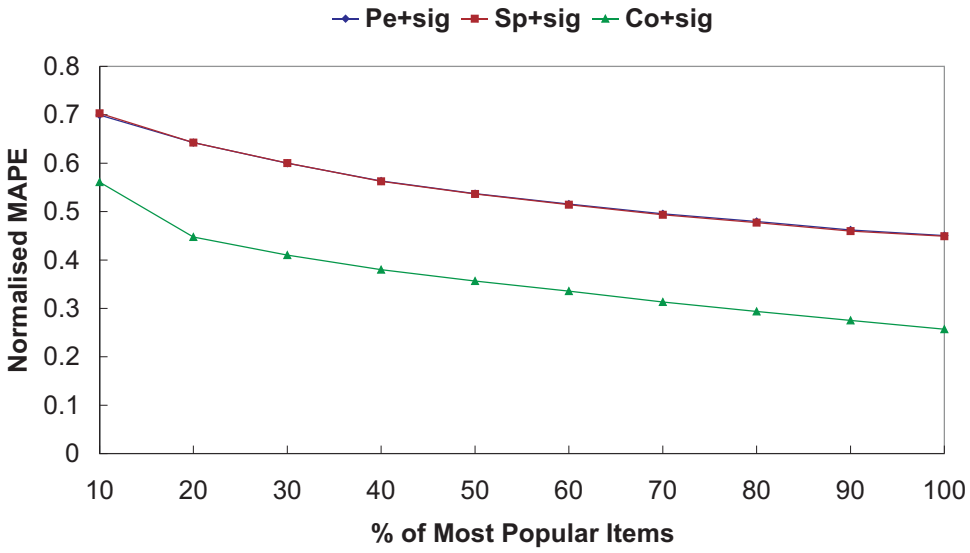


Figure 5.12: The effect of item popularity knowledge on robustness calculated according to normalised MAPE for the EachMovie dataset. Attack profiles contain items which are drawn from the top–X% (shown on the x–axis) of the most popular items.

The percentages of good predictions that were achieved by the attacks are shown in Figure 5.13. As before, the attack was less successful as item popularity knowledge decreased. In all cases, however, the post–attack percentages of good predictions for all similarity metrics were well in excess of the pre–attack level (labeled 'Baseline'). In addition, when no item popularity knowledge was assumed to be known by the attacker, the percentages of good predictions that were achieved were substantial.

[5]From a total of 1,338 items in the EachMovie dataset which received at least 1 rating.

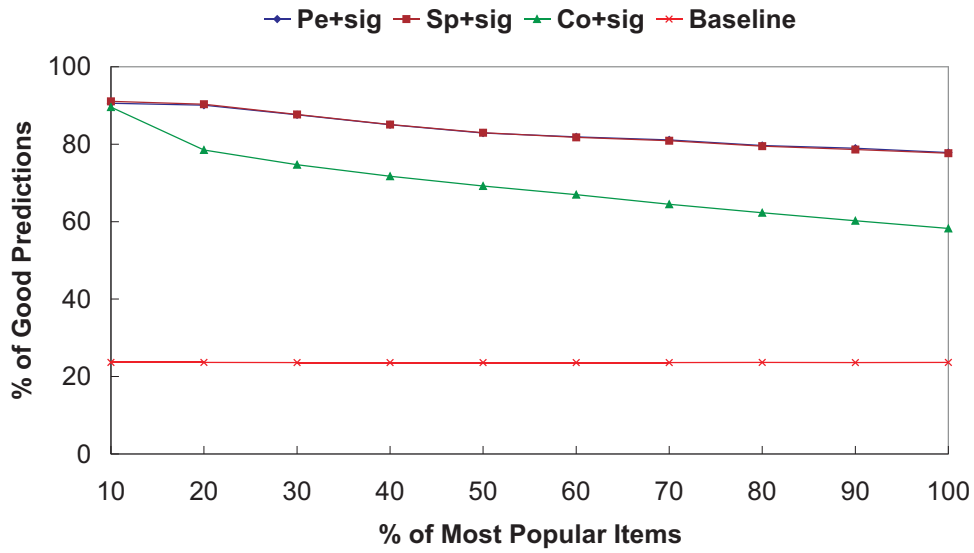


Figure 5.13: The effect of item popularity knowledge on robustness calculated according to the percentage of good predictions that are achieved by the attack for the EachMovie dataset. Attack profiles contain items which are drawn from the top–X% (shown on the x–axis) of the most popular items.

Figure 5.14 shows predictive accuracy, calculated according to normalised MAE, for the attacks. While accuracy improved as popularity knowledge was reduced, the post–attack accuracy was, in all cases, significantly worse than that provided by the non–personalised benchmark algorithm. For example, when no item popularity knowledge was assumed, the post–attack accuracy that was achieved using the Pearson and Cosine similarity metrics was 0.4936 and 0.3359, respectively, compared to 0.1986 provided by the benchmark algorithm.

Thus, for all the performance indicators shown above, the focused push attack remained successful, even in circumstances where a complete lack of popularity knowledge existed. This finding is significant, since it indicates that item popularity knowledge is not a necessary requirement for attack success.

5.4.5.2 Item Likability

We now examine the effect of item likability knowledge on attack success. Item likability knowledge is only required by an attacker when the correlation similarity metrics are used[6]. This knowledge allows an attacker to create attack profiles using items that are generally liked or disliked in a system. If higher ratings are assigned to the liked items,

[6]Recall that Cosine similarity can only result in positive values when systems operate using positive rating scales, as is the case with the datasets that are used for evaluation purposes in this thesis.

and lower ratings to the disliked items, this strategy is likely to result in consistently positive correlations between attack and genuine users.

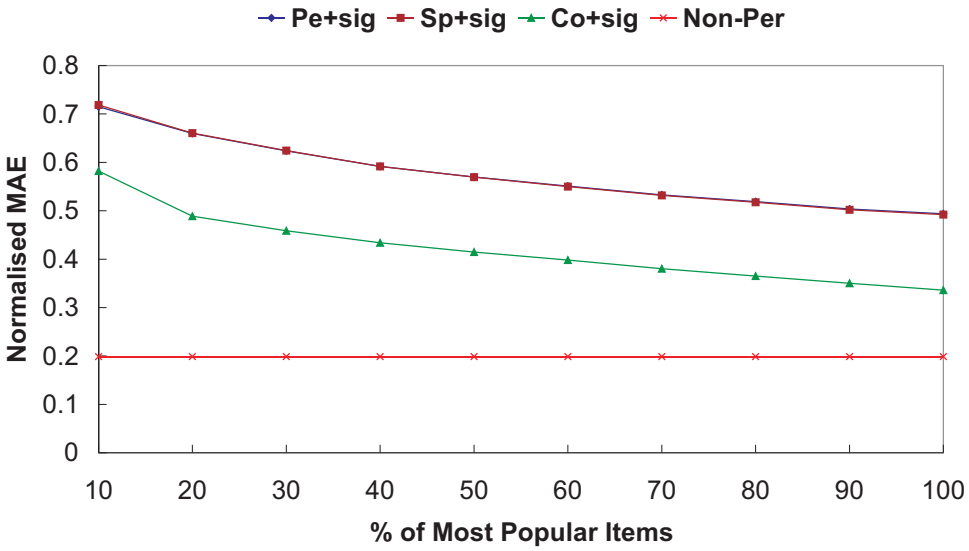


Figure 5.14: The effect of item popularity knowledge on predictive accuracy for the EachMovie dataset. Attack profiles contain items which are drawn from the top–X% (shown on the x–axis) of the most popular items.

In this analysis, it is assumed that full item popularity knowledge is known to the attacker. To simulate partial item likability knowledge, a number of attacks are implemented where a certain percentage of liked and disliked items are misclassified. For example, a percentage misclassification of 10% means that a randomly selected 10% of the liked items that are contained in attack profiles receive a lower rating and a corresponding percentage of the disliked items receive a higher rating.

As before, the attack profiles that are used in all of the simulations contain 100 items and, in each case, a total of 64 attack profiles are inserted into the system. The analysis is again performed using the EachMovie dataset.

Figure 5.15 shows the effect of item likability knowledge on system robustness, which is calculated according to normalised MAPE. It can be seen from the results that robustness against attack improved as the percentage of misclassified items approached 50%. For example, the normalised MAPE fell to approximately 0.3356 for both correlation metrics when 40% of items were misclassified, compared to a value of approximately 0.7046 when all items were correctly classified. The reason for the improvement in robustness was that lower similarities between active and genuine users were achieved as the misclassification rate was increased, and thus, fewer attack profiles were present in neighbourhoods. Nevertheless, the normalised MAPE values of 0.3356 and above that were achieved for

misclassification rates of $\leq$ 40% are still significant and, as shown below, can lead to substantial changes in the predictions that are made by systems.

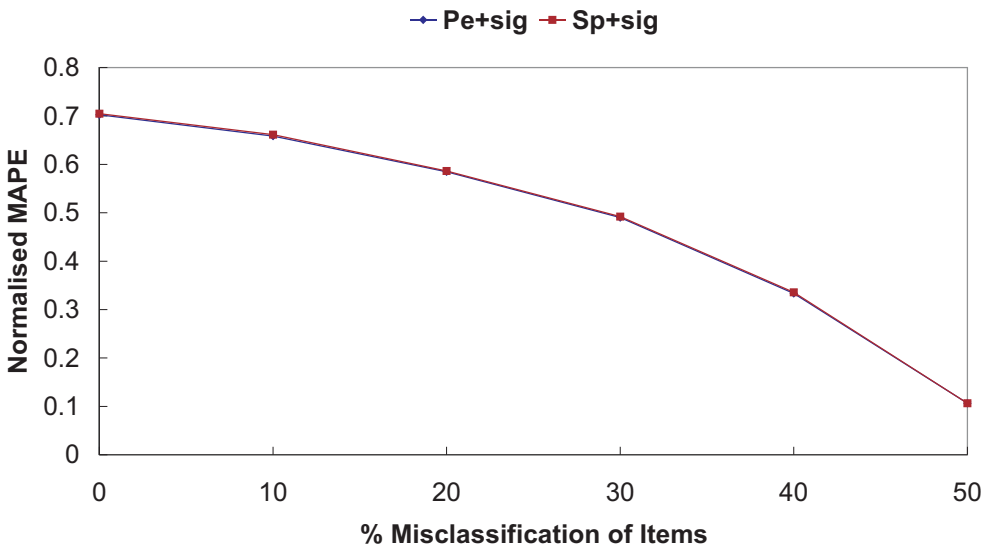


Figure 5.15: The effect of item likability knowledge on robustness calculated according to normalised MAPE for the EachMovie dataset.

Figure 5.16 shows the percentages of good predictions that were achieved by the attack. It is clear that the attack was very successful for misclassification rates of up to and including 40%. At this latter point, 72% of all predictions either exceeded or equaled the second–best rating, compared to 89% when all items were correctly classified. These values are well above the pre–attack baseline percentage of good predictions, which was 24%.

An interesting effect occurred when 50% of items were misclassified, where the percentage of good predictions achieved actually fell below the pre–attack level. The reason for this sharp decrease is evident from Figure 5.17, where the percentages of positive prediction shifts that were achieved by the attacks are shown. At this point, only 45% of predictions shifts were positive, compared to 90% when 40% of items were misclassified. Since a misclassification rate of 50% is equivalent to an attacker randomly assigning attack profile items to either the liked or disliked groups, attack success is likely to be poor and this is reflected in the results. It should be noted that misclassification rates in excess of 50% result in targeted items being *nuked*, since negative correlations between genuine and attack profiles predominately occur, which in turn lead to negative prediction shifts.

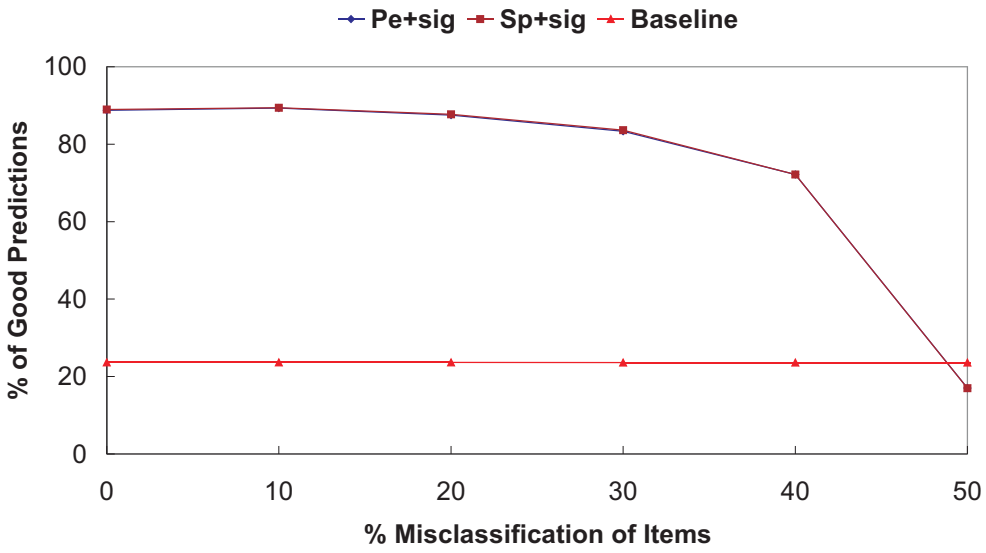


Figure 5.16: The effect of item likability knowledge on robustness calculated according to the percentage of good predictions that are achieved by the attack for the EachMovie dataset.

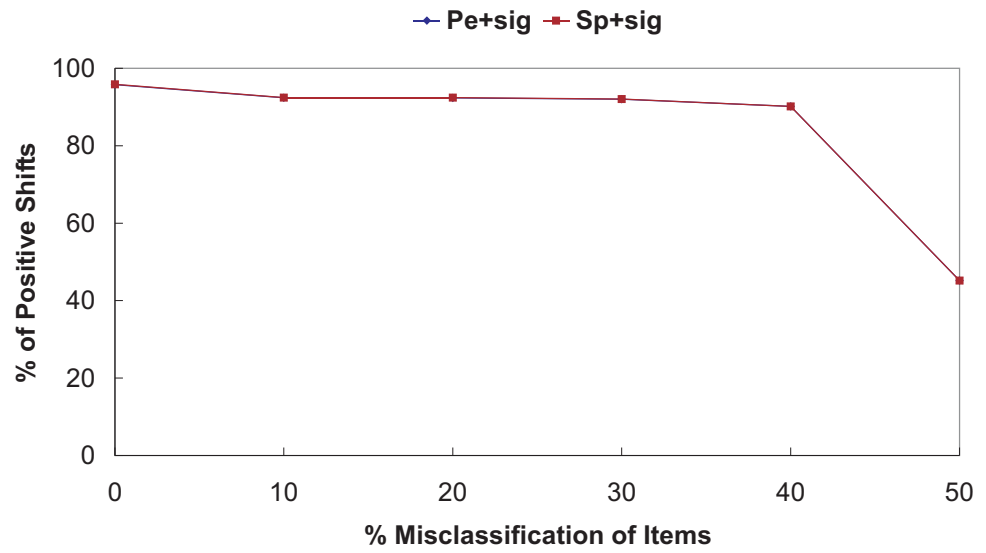


Figure 5.17: The effect of item likability knowledge on the percentage of the positive prediction shifts that are achieved by the attack for the EachMovie dataset.

Predictive accuracy versus item likability knowledge is shown in Figure 5.18. As before, accuracy improved as the misclassification rate was increased. However, the post–attack accuracy was significantly worse than that provided by the non–personalised benchmark algorithm in all cases, except for a misclassification rate of 50%, where approximately equivalent accuracy was provided. Thus, when all the performance metrics that were

evaluated above are considered, it is clear that the focused push attack was successful for misclassification rates of $\leq 40\%$

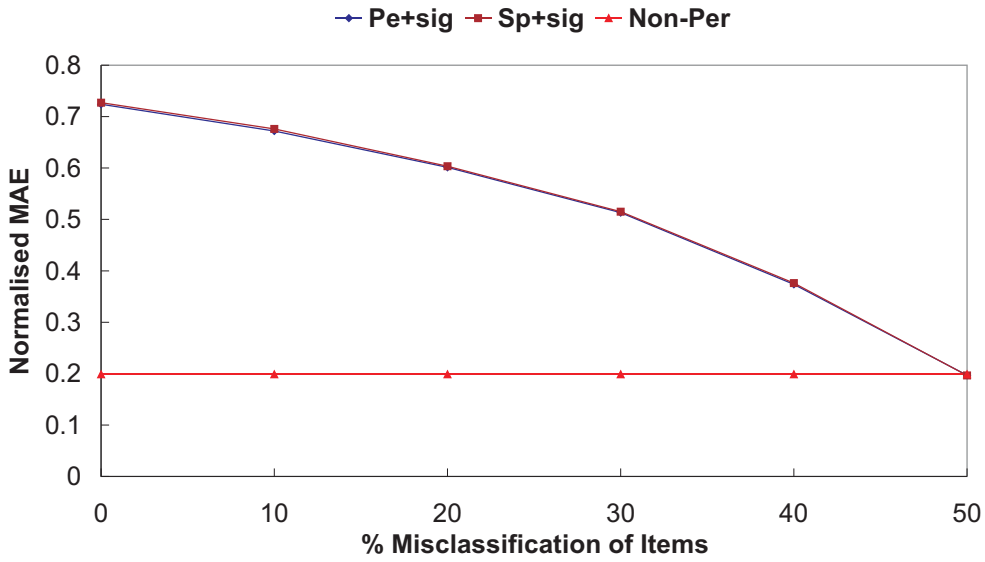


Figure 5.18: The effect of item likability knowledge on predictive accuracy for the EachMovie dataset.

In conclusion, it can be stated that the attack of Section 5.4.3 remained successful in circumstances where a considerable lack of knowledge pertaining to both item popularity and item likability existed. Since these findings indicate that successful attacks can be implemented against systems when only limited data is available to attackers, the need for robust collaborative filtering algorithms to defend against malicious attack is further demonstrated.

5.5 Effect of Algorithm Extensions

In Section 2.2.5, several extensions to the standard memory–based collaborative filtering algorithm are outlined. These extensions were introduced in other work with a view to improving the predictive accuracy of the algorithm. In this section, the effect, if any, of four of these extensions on system robustness is examined.

5.5.1 Significance Weighting

We begin by examining the significance weighting extension, noting that this extension has established predictive accuracy benefits and is now almost universally used in conjunction with memory–based collaborative filtering algorithms. In Section 5.4.2, a successful attack

was demonstrated using a 3–item attack profile and the k–NN neighbourhood formation scheme. However, as expected, when significance weighting was incorporated into the algorithm, the system was substantially more robust against attack since the extension caused the exclusion of the majority of the small attack profiles from neighbourhoods. Thus, this experiment readily establishes the importance of significance weighting from a robustness perspective, in addition to the known benefits from the point–of–view of predictive accuracy.

Significance weighting alone, however, in incapable of rendering a system secure against attack. As demonstrated in Section 5.4.3, where an attack was mounted using larger attack profiles, significance weighting had little impact on the attack. Table 5.5 shows the robustness metrics for this attack, with and without the significance weighting extension (labeled '+sig'). As can be seen, there is little to distinguish between the results. For example, consider the MovieLens dataset (Pearson correlation), where normalised MAPE and the percentage increase in good predictions were only reduced from 0.6898 and 395% to 0.6716 and 368%, respectively, as a result of this measure. In essence, we conclude that significance weighting is important from the robustness perspective by rendering cheap attacks ineffective – i.e. it has the important effect of increasing the cost of attack.

5.5.2 Case Amplification

The effect of case amplification on robustness is somewhat more interesting and complicated. In general, the results indicate that this extension had only a limited effect in reducing attack success. Since the influence of weights that are closer to unity is increased, there needs to be a significance difference in the magnitude of the weights of genuine and attack neighbours if this measure is to have an effect. From our analysis, this was not the case even though, generally, genuine neighbours tended to have greater similarities to the active user than attack neighbours. However, in the majority of cases, the difference was not sufficient to have a significant effect. Referring to Table 5.5, it can be seen that, while case amplification (labeled '+amp') did have some impact on attack success across all datasets and similarity measures evaluated, it was not decisive. As a typical example, consider the results for the EachMovie dataset using Pearson correlation, where the percentage increase in good predictions only fell from 278% to 245% due to this measure.

It is interesting to note that it is possible for case amplification to compromise system robustness in certain scenarios. Consider, for example, a situation where the active user has only a small number of good quality, genuine neighbours. In such cases, if the similarities between the active user and attack neighbours exceed those of the genuine neighbours, then case amplification will serve to increase the influence of the attack profiles. Such an effect was observed with the 3–item attack profiles of Section 5.4.2, where attack profiles had maximal correlation with genuine users (assuming significance weighting was not

used).

The conclusion reached regarding this extension is that, since it has only a limited effect on system robustness in general, and given that for certain users it has the potential to increase attack success, we believe that it should not be considered as part of the collaborative filtering algorithm.

Table 5.5: The effects of several algorithm extensions on system robustness. Results correspond to a focused push attack (using large attack profiles) carried out on the three datasets, using an attack strength of 64 attack profiles inserted into the respective systems.

	MovieLens		**EachMovie**		**Smart Radio**	
	NMAPE	% Incease in Good Preds.	NMAPE	% Incease in Good Preds.	NMAPE	% Incease in Good Preds.
Co	0.4274	314	0.6331	310	0.7515	145
Co+sig	0.3766	281	0.6172	304	0.7369	141
Co+amp	0.3723	274	0.5687	284	0.7237	134
Co+iuf	0.2590	206	0.5802	300	0.1973	61
Co+en	0.4334	312	0.6543	308	0.4548	121
Pe	0.6898	395	0.6654	278	0.7446	132
Pe+sig	0.6716	368	0.7024	276	0.7567	132
Pe+amp	0.6574	349	0.6215	245	0.6976	118
Pe+iuf	0.6952	396	0.6657	279	0.7129	132
Pe+en	0.6688	387	0.6598	279	0.7247	129
Sp	0.6890	397	0.6674	279	0.7416	130
Sp+sig	0.6699	367	0.7046	273	0.7552	127
Sp+amp	0.6569	347	0.6254	245	0.6848	111
Sp+iuf	0.6999	404	0.6697	277	0.7130	128
Sp+en	0.6653	387	0.6628	277	0.7340	128

5.5.3 Inverse User Frequency

This extension serves to diminish the influence of more popular items in the computation of similarities between users. Recall from Section 5.4.3 that a key feature of the attack strategy outlined was to construct attack profiles using the most popular items in the datasets, since such items would ensure a high degree of commonality between the attack profiles and the majority of genuine users in a system. Thus, it would seem that this particular extension should have a substantial impact on attack success.

The results (labeled '+iuf') presented in Table 5.5 are, however, disappointing. In particular, the extension had virtually no effect when the correlation similarity metrics were

used. Consider, for example, the Smart Radio dataset using Spearman rank correlation, where the percentage increase in good predictions was reduced by only 2% from a baseline of 130% when the extension was incorporated into the algorithm. The results were more encouraging when Cosine similarity was used. The greatest effect was seen for the Smart Radio dataset, where the percentage increase in good predictions achieved by the attack was reduced from 145% to 61% as a consequence of the extension. The improvement in robustness was not as strong, however, for the other datasets, particularly in the case of the EachMovie dataset.

To conclude, it can stated that the improvements in robustness brought about by the inverse user frequency extension were disappointing, especially in the case of the correlation similarity metrics. This is a surprising result, given that the extension was designed to minimise the influence of popular items in user similarity computations – the very items that were used to construct the attack profiles – and failed to do so.

5.5.4 Entropy

The item entropy extension is similar to the above, in that it attempts to minimise the influence of certain items in the calculation of similarity weights. The motivation behind this approach is that items which are rated consistently by many users are less able to discriminate between users than those that receive a more uniform distribution of ratings. Thus, the influence of consistently rated items on similarity weight calculations is reduced, while the influence of items that are rated in a more diverse fashion is increased.

The results for this extension (labeled '+en') are presented in Table 5.5. When the correlation metrics were used, the extension had little effect across all the datasets that were evaluated. For example, in the case of the MovieLens dataset using Pearson correlation, the percentage increase in good predictions achieved by the attack was only reduced from 395% to 387% as a result of the entropy extension. With the exception of the Smart Radio dataset, similar trends were observed when Cosine similarity was used. For this particular dataset, item entropy caused the percentage increase in good predictions to fall to 121%, compared to 145% when entropy was not considered. Nevertheless, the extension failed to prevent the attack from retaining a considerable measure of success.

Thus, in conclusion, item entropy did not significantly enhance the robustness of the datasets that were subjected to attack. Given the lack of relationship that existed between attack success and item entropy as outlined in Section 5.4.4, this result was not surprising.

5.6 Product Nuke Attacks

In this section, the strategy underlying product nuke attacks is developed. In essence, a very similar approach as set out above for the push attacks is employed. As before, the objective is to ensure that as large a number as possible of attack profiles contribute to predictions, but in this case the attacker needs to ensure that the deviation from mean term in (2.7) is negative, as required for a successful nuke attack. Once again, it is assumed that item likability and popularity knowledge for the system in question is known to the attacker.

The strategy that is adopted when the correlation similarity metrics are used is the same as that for a push attack. Dataset items are first assigned into liked and disliked groups, and a quantity of the most popular items from each group is then used to construct the attack profiles[7]. The item being attacked is also included.

Ratings are chosen for attack profile items in the following manner. The liked items receive the maximum rating, r_{max}, disliked items receive a rating of $r_{max} - 1$ and the item to be nuked receives the minimum rating, r_{min}. As with the product push attack strategy, this ratings scheme is likely to result in positive correlations between attack and genuine users. In addition, the difference term $(r_{i,j} - \bar{r}_i)$ in (2.7) is minimised for the attack profiles, as required for a nuke attack.

Once more, the attack strategy is simplified when Cosine similarity is used. The attacker simply chooses the l popular items contained in the dataset, which are randomly assigned ratings of r_{max} and $r_{max} - 1$. The item under attack is assigned a rating of r_{min}. Since Cosine similarity can only result in positive values for systems that operate using positive rating scales, this strategy achieves the desired result of minimising the deviation tern in (2.7).

It is important to note that the only difference between the attack profiles constructed for the product push attack as outlined in Section 5.4.3, and for nuke attack strategy described here, lies in the ratings that are assigned to items – the actual items that are selected to build the attack profiles are the same.

Table 5.6 shows the results for the MovieLens, EachMovie and Smart Radio datasets, for an attack strength of 64 attack profiles inserted into each of the systems. The results presented are average performance over all users and items in the respective datasets. Since the EachMovie dataset operates on a different rating scale compared to the other datasets, normalised values for MAPE and MAE are presented in order to permit a direct comparison between the systems. For all similarity metrics, the results indicate that the attack was

[7]Recall from Section 5.4.3 that the liked group is comprised of dataset items whose average ratings are greater than the mean item likability, and the disliked group contains dataset items whose average ratings are less than the mean item likability

successful against all datasets. In each case, the majority of the prediction shifts achieved were negative, as required for a successful nuke attack. When the correlation metrics were used, the percentage of attack profiles that were contained in neighbourhoods were identical to those achieved in the push attack of Section 5.4.3, which was expected, given the similarity between the attack profiles used in both attacks. For the EachMovie and Smart Radio datasets, the normalised MAPE and post–attack MAE values resulting from both attacks were also broadly similar. When the MovieLens dataset is considered, however, the push attack was more successful, achieving , for example, a normalised MAPE value of 0.6716 compared to 0.5778 for the nuke attack (Pearson correlation). Nevertheless, the attack against the MovieLens dataset was successful in achieving substantial prediction shifts. Furthermore, the post–attack accuracy performance for MovieLens, as for all the datasets, was significantly poorer than that provided by the non–personalised algorithm.

Table 5.6: Focused Nuke Attack (large attack profiles) carried out on the three datasets. Results shown are for an attack strength of 64 attack profiles inserted into datasets. Accuracy (normalised MAE) computed according to the non–personalised benchmark algorithm is 0.2341 (MovieLens), 0.1986 (EachMovie) and 0.2689 (Smart Radio). Any normalised MAE values that exceed this baseline are shown in boldface.

	% Neg. Shifts	% Attack Profiles in Neighbourhoods	NMAPE	% Bad Preds. (pre)	% Bad Preds. (post)	NMAE (pre)	NMAE (post)
MovieLens							
Pe+sig	85	75	0.5778	36	83	0.1969	**0.5992**
Sp+sig	84	75	0.5777	36	83	0.1974	**0.5988**
Co+sig	80	57	0.4128	39	78	0.1997	**0.4612**
EachMovie							
Pe+sig	87	81	0.6881	18	86	0.1798	**0.7062**
Sp+sig	87	82	0.6908	18	86	0.1788	**0.7087**
Co+sig	97	79	0.6433	19	92	0.1812	**0.6536**
Smart Radio							
Pe+sig	95	93	0.7923	24	94	0.2465	**0.7886**
Sp+sig	93	93	0.7934	23	92	0.2467	**0.7889**
Co+sig	98	89	0.7573	24	98	0.2566	**0.7595**

From the results, the Smart Radio dataset proved the most vulnerable attack. As before, this can be explained by the relatively small size of this dataset, which contains only 63 users, compared to the (approximately) $1,000$ users contained in the other systems. Finally, we note that the performance offered by both Pearson and Spearman rank correlations was closely matched, which has been the the case in all previous results.

5.7 Alternative Attack Strategies

In this section, we examine some attack strategies that have been proposed by Lam and Riedl (2004). In this work, the two strategies outlined are referred to as *RandomBot* and *AverageBot*. As with all the attacks considered in this thesis, both strategies involve the creation of attack profiles that are inserted into a system – no direct access to a system database is assumed for an attacker.

In both strategies, all of the items that are contained in a system are used in the attack profiles, and the issue of attack cost is not considered. The difference between the two strategies relates to the manner in which ratings are assigned to the items that are contained in the attack profiles. In the RandomBot strategy, the items being pushed or nuked are set to the maximum or minimum ratings, respectively. The remainder of the items are rated randomly on a normal distribution with mean and standard deviation equal to the ratings distribution of the system being attacked.

A more sophisticated ratings scheme is used in the AverageBot strategy, where it is assumed that the item likability for all items is known by the attacker. In this case, attack profile items are rated randomly on a normal distribution with mean equal to the average rating of the item being rated and standard deviation equal to the standard deviation of all the items contained in a system. As before, the items being pushed or nuked are set to the maximum or minimum ratings, respectively.

From the results obtained by Lam and Riedl (2004), the AverageBot attack strategy was more effective in pushing and nuking predicted ratings than the RandomBot strategy. Thus, we focus on the AverageBot attack, and compare the performance of this strategy for a push attack against the focused push attack strategy as outlined in Section 5.4.3. In each case, the number of attack profiles that are inserted into the systems is 64. Note that the same AverageBot strategy is employed irrespective of the similarity metric being used by a system. Given the ratings strategy that is applied to attack profile items, attack profiles can be expected to correlate positively with the majority of genuine users contained in a system.

The results for the AverageBot push attack are presented in Table 5.7. For all the datasets and similarity metrics that were evaluated, the attack succeeded in reducing accuracy performance beyond that provided by the benchmark non–personalised algorithm, which indicates that, in this regard, a successful attack took place. In addition, the majority of the prediction shifts that were achieved using all similarity metrics were positive, as required for a successful product push attack.

Table 5.7: AverageBot Push Attack carried out on the three datasets. Results shown are for an attack strength of 64 attack profiles inserted into datasets. Accuracy (normalised MAE) computed according to the non–personalised benchmark algorithm is 0.2341 (MovieLens), 0.1986 (EachMovie) and 0.2689 (Smart Radio). Any normalised MAE values that exceed this baseline are shown in boldface.

	% Pos. Shifts	% Attack Profiles in Neighbourhoods	NMAPE	% Good Preds. (pre)	% Good Preds. (post)	NMAE (pre)	NMAE (post)
MovieLens							
Pe+sig	99	62	0.3173	19	73	0.1969	**0.3709**
Sp+sig	98	62	0.3129	19	73	0.1974	**0.3682**
Co+sig	100	60	0.3295	17	70	0.1997	**0.3806**
EachMovie							
Pe+sig	98	74	0.2548	24	80	0.1798	**0.2897**
Sp+sig	98	73	0.2514	24	80	0.1788	**0.2880**
Co+sig	100	74	0.2721	23	80	0.1812	**0.3099**
Smart Radio							
Pe+sig	96	97	0.3370	41	95	0.2465	**0.3620**
Sp+sig	96	97	0.3342	41	95	0.2467	**0.3615**
Co+sig	100	96	0.3414	41	94	0.2566	**0.3600**

The attack did not perform as well as the focused push strategy. The magnitude of the prediction shifts that were achieved by the AverageBot attack were, in all cases, substantially reduced. For example, a normalised MAPE value of 0.3173 was achieved for MovieLens using Pearson correlation, compared to 0.6716 when the focused push attack strategy was used. Likewise, the post–attack accuracy performance was better in the case of the AverageBot attack, with lower normalised MAE values being achieved. The differences in the results between the two attack strategies are accounted for by the ratings strategies that are used in each. Recall that, in the focused push attack strategy, ratings were selected so as to maximise the difference term $(r_{i,j} - \bar{r}_i)$ for the attack profiles in (2.7), thereby resulting in maximal prediction shifts being caused by the attack. This is not the case for the AverageBot attack, where the ratings strategy that is applied does not maximise the difference term in (2.7), and consequently, smaller predictions shifts are achieved.

Notwithstanding the differences in the prediction shifts caused by the different attack strategies, the percentages of good predictions achieved by the AverageBot attack across all datasets and similarity metrics were more closely matched to those of the focused push attack. The reason for this result lies in the distributions of the item ratings contained in the datasets. As can be seen from Table 5.8, the mean rating (across all items) for each dataset exceeds the midpoint of the particular rating scale that is used. Thus, from (5.1),

Table 5.8: Mean and standard deviation of ratings contained in the MovieLens, EachMovie and Smart Radio datasets.

	Rating Scale	Mean	Standard Deviation
MovieLens	1–5	3.53	1.13
EachMovie	1–6	4.34	1.29
Smart Radio	1–5	3.60	1.29

only a small prediction shift is sufficient to cause a substantial increase in the number of good predictions that are made. It should be noted, however, that the AverageBot attack would be less successful against systems in which the mean item ratings were closer to the midpoint ratings.

In conclusion, the AverageBot attack was not as successful as the focused push attack when all performance measures are considered. In addition, the cost of the AverageBot attack was a multiple of that of the focused push attack, where all available items were used to construct the AverageBot attack profiles compared to only the 100 items that were used in the focused push attack profiles.

5.8 Neighbourhood Formation Schemes

Neighbourhood formation concerns the identification of similar users in a system that are used as predictors for the active user. In this section, the performance of the neighbourhood formation schemes outlined in Section 2.2.3 is examined. The objective is to determine whether the different schemes are more, or less, vulnerable to attack. In addition, we seek to establish if any compromises, with respect to coverage and predictive accuracy, are required in order to achieve robustness against attack (O'Mahony *et al.*, 2003b). To answer these questions, the coverage and accuracy that is provided by the schemes, prior to attack, is first examined. The analysis is performed using the EachMovie dataset. Similar results were obtained using the MovieLens and Smart Radio datasets and thus, the results for these systems are not shown.

5.8.1 Coverage

Figure 5.19 shows the coverage provided by Cosine similarity and the correlation metrics using k–NN, similarity thresholding and a combined k–NN and thresholding scheme (which is referred to as 'k–NN *min–corr*'). The latter scheme uses a threshold value of 0.1, as per Lam and Riedl (2004). For the similarity thresholding scheme, four threshold values, L, were considered, ranging from $L = 0.1$ to $L = 0.7$. For k–NN *min–corr*, the

optimal neighbourhood size, with respect to predictive accuracy, was determined by experiment and set to 45 (see Appendix C for details on the neighbourhood size selection process). The k–NN scheme provided the best coverage, achieving 88% for all similarity metrics. The next–best performing schemes were thresholding at $L = 0.1$ and k–NN *min–corr*, which provided coverage of 85% and 84%, respectively.

Similarity thresholding is known to result in reduced coverage at higher threshold values, and this was evident in the results. Coverage decreased significantly as the threshold was increased, and at $L = 0.7$, the coverage provided by all similarity metrics was almost zero. Similar trends have been noted by Shardanand and Maes (1995) and Herlocker *et al.* (1999). In addition, it can be seen that Cosine similarity gave better coverage than the correlation metrics at threshold values of 0.3 and 0.5. For example, at $L = 0.5$, Cosine similarity provided coverage of 43%, compared to approximately 29% for the correlation metrics.

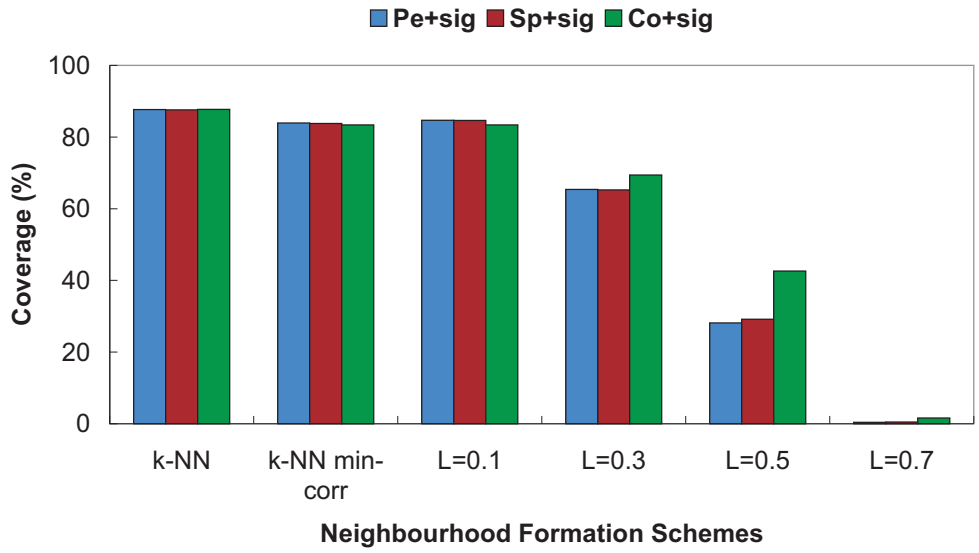


Figure 5.19: EachMovie: system coverage achieved by the Pearson, Spearman rank and Cosine similarity metrics using various neighbourhood formation schemes

5.8.2 Accuracy

System accuracy, calculated according to mean absolute error, is shown for the various similarity measures and neighbourhood formation schemes in Figure 5.20. Generally, Spearman rank correlation provided the best accuracy, closely followed by Pearson correlation. Cosine was outperformed by the correlation metrics, except for similarity thresholding at $L = 0.7$, where Spearman rank correlation was poorest. This result is unlikely to be of great significance, however, since the coverage provided at this threshold value by all similarity metrics was close to zero.

Both k–NN neighbourhood formation schemes were closely matched in terms of predictive accuracy, with k–NN *min–corr* performing only marginally better. The similarity thresholding scheme provided comparable performance at the lowest threshold value of $L = 0.1$. Thereafter, accuracy began to fall off as L was increased. These findings are in agreement with those reported by Shardanand and Maes (1995) and Herlocker *et al.* (1999).

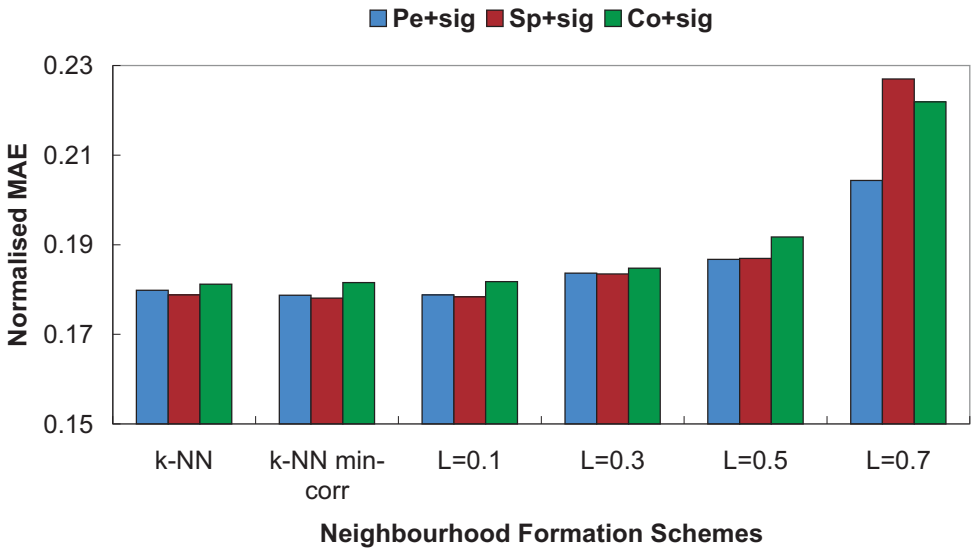


Figure 5.20: EachMovie: predictive accuracy according to Normalised MAE for the Pearson, Spearman rank and Cosine similarity metrics using various neighbourhood formation schemes

5.8.3 Robustness

In this section, the robustness provided by each of the neighbourhood formation schemes is examined. The attack strategy used in each case is that as outlined in Section 5.4.3. Results are shown in Figure 5.21 for an attack strength of 64 attack profiles inserted into the dataset. In order to facilitate a direct comparison between the various similarity metrics, robustness is presented in terms of the percentage increase in the number of good predictions caused by the attack.

It can be seen from the figure that all similarity metrics proved vulnerable to the attack. The performance provided by both the correlation metrics was similar, as has been previously noted. Cosine similarity performed the worst, except for similarity thresholding at $L = 0.5$ and $L = 0.7$, where the coverage provided by all similarity metrics was poor. The choice of neighbourhood formation scheme had a greater impact on system robustness. The k–NN scheme performed the worst, resulting in a percentage increase in the number of good predictions of 276% (Pearson correlation). The performance offered by the k–NN

min–corr scheme was only marginally better, resulting in a 257% increase in the number of good predictions achieved by the attack. For similarity thresholding, robustness improved as the threshold was increased. For example, at $L = 0.1$, the percentage increase in good predictions was 269% compared to 11% at $L = 0.7$.

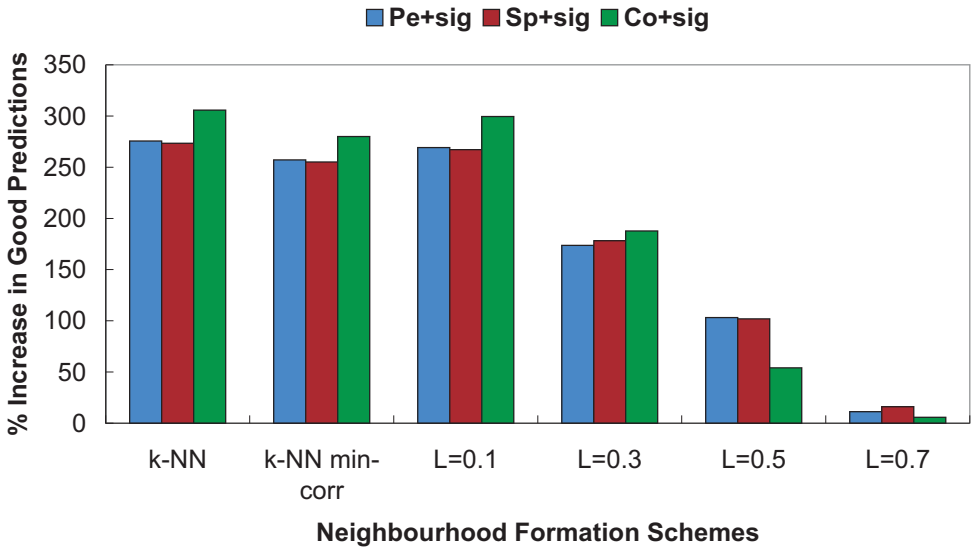


Figure 5.21: Robustness according to the percentage increase in the number of good predictions for an attack strength of 64 attack profiles inserted into the EachMovie dataset

To explain the above results, recall that the attack strategy involved creating attack profiles that had a good probability of being similar to many genuine users in the system. For the k–NN and k–NN *min–corr* schemes, neighbourhood sizes for both were fixed at $k = 45$. Since many genuine users did not typically have k *very* similar neighbours, however, the result was that attack profiles were included in many of the neighbourhoods. (For Pearson correlation, 81% and 75% on average of all neighbours were attack profiles for the k–NN and k–NN *min–corr* schemes, respectively.)

For similarity thresholding, attack profiles were located in the neighbourhoods of increasingly fewer users as the threshold, L, was increased. Thus, a corresponding improvement in robustness was observed. For example, at the lowest threshold value of $L = 0.1$, the similarities between many genuine and attack profiles exceeded this threshold value, and consequently some 66% of neighbours on average were attack profiles (Pearson correlation). At the highest threshold value of $L = 0.7$, attack profiles were almost completely filtered from the neighbourhoods, where only 2% of neighbours were attack profiles.

From the results presented in this section, it is clear that none of the similarity metrics or neighbourhood formation schemes provided satisfactory robustness against attack. The only exception was for the similarity thresholding neighbourhood formation scheme, where

robustness did improve significantly as the threshold value was increased. However the increased performance was offset by the poor coverage provided by this scheme at higher threshold values. Thus, it can be concluded that all schemes were vulnerable to attack when acceptable degrees of coverage were provided.

As was stated in Section 5.4.3, one of the main difficulties for an attacker lies in ensuring that all attack profiles correlate in the same direction (i.e. either positively or negatively) with genuine users when the Pearson and Spearman rank similarity metrics are used. While this is an issue for the k–NN and similarity thresholding neighbourhood formation schemes, it is not, however, the case with the k–NN *min–corr* scheme, where all negative correlations are ignored by the algorithm. Thus if an attacker was unsure about which ratings to assign attack profile items, he could simply create two sets of attack profiles, with the ratings schemes reversed in each. For example, if item i received a high rating in the first set, it would receive a low rating in the second set. With this system, one set of attack profiles would correlate positively with a given genuine user, while the other set would correlate negatively. Because negative correlations are ignored by the algorithm, there is no risk of one set of attack profiles canceling out the contributions of the other, and thus, the attack is likely to succeed. While such a strategy necessarily results in a higher attack cost, it nevertheless represents an additional security implication for the k–NN *min–corr* scheme.

5.9 Cost–Benefit Analysis

The objective of this section is to conduct a cost–benefit analysis for the attacks as outlined above. In particular, we consider the product push attack that is described in Section 5.4.3, which is carried out against a collaborative filtering algorithm using the k–NN neighbourhood formation scheme. Since Pearson correlation (with significance weighting) is the most widely used similarity metric in collaborative filtering algorithms, the results that are presented here relate to this particular metric. It is important to understand how system robustness varies with dataset size – for example, as the number of genuine users in a system grows, attacks may well become more difficult or costly to implement successfully. Thus, for this analysis, the EachMovie dataset is used due to the large quantity of data that is available. In order to simulate a system as it increases in size over time, a number of different–sized samples are randomly drawn from this system, where the samples, d_i, are subject to the following constraint: $d_1 \subset d_2 \subset \ldots \subset d_n$. The smallest sample contains $1,000$ genuine users[8], the largest contains $10,000$ users.

The metric that is adopted for this cost–benefit analysis is Return on Investment (ROI).

[8]This sample corresponds to that which is used in all of the previous evaluations that have been performed using the EachMovie dataset.

Pearce (1992) provides the following definition of ROI:

"*ROI is a general concept referring to Earnings from the Investment of Capital, where the earnings are expressed as a proportion of the outlay*".

ROI is frequently derived as the return from an action divided by the cost of that action, which is expressed as:

$$\text{ROI} = \frac{\text{Gains} - \text{Invested Capital}}{\text{Invested Capital}} \tag{5.3}$$

We now apply this formula to the product push attack to establish what returns, if any, can be realised by an attacker. To begin, we state a number of assumptions that are made with respect to this analysis:

(1) users can only insert ratings into a system following the actual purchase of items,

(2) all items have equal financial cost, and

(3) users will only consider the purchase of a new item if the predicted rating that is made for that item is equal to r_{max} or $r_{max} - 1$.

Assumptions (1) and (2) are not favourable to an attacker. For example, in many cases, purchases are not required in order to insert ratings. In addition, new items, which are likely to be the subject of most attacks, often sell at prices that exceed those of more established items. Thus, in our scenario, an attacker needs to carefully consider the cost that is associated with any attack. For example, if the cost of a particular attack is too high, then a significant return may not follow, or potentially, an attack could even result in financial losses.

Consider a system, d, and an attack that is carried out against an item i, where $i \in d$. The Gains and Invested Capital terms in (5.3) for such an attack are given by:

$$\text{Gains}(i) = \mathcal{U} \times \Pr(d) \times (g'_i - g_i) \times \kappa_i \times c_i \tag{5.4}$$

$$\text{Invested Capital(i)} = n_{i,a} \times c_i \tag{5.5}$$

where:

- $\mathcal{U}$ is the universe of all users,
- $\Pr(d)$ is the probability that a user accesses system d,

- $g'_i = \Pr\Big(p'_i = \{r_{max}, r_{max} - 1\}\Big)$ is the post–attack probability that a good prediction is made for item i,[9]
- $g_i = \Pr\Big(p_i = \{r_{max}, r_{max} - 1\}\Big)$ is the pre–attack probability that a good prediction is made for item i,
- $\kappa_i = \Pr\Big(b_i | p_i = \{r_{max}, r_{max} - 1\}\Big)$ is the browser–to–buyer conversion rate for item i, which is the probability that users actually purchase item i given that a good prediction is made for the item (assumed to be the same across all items),
- c_i is the purchase cost of item i (an equal cost is assumed for all items), and
- $n_{i,a}$ is the total number of bogus ratings that are inserted into the system in the course of the attack on item i.

Note that $\mathcal{U} \times \Pr(d)$ corresponds to the number of users that are contained in a particular system, d. The quantities g_i and g'_i are estimated from the data using the percentage of good predictions metric (Section 4.2.4). For optimal attack success, an attacker needs to maximise the ROI with respect to the item being attacked, which, from equations (5.3), (5.4) and (5.5), is equivalent to:

$$\text{ROI}(i)_{max} \equiv \max \left[\frac{g'_i - g_i}{n_{i,a} \times c_i} \right] \tag{5.6}$$

Using the smallest sample, d_1, drawn from the MovieLens dataset, the following experiment is carried out to determine the optimal number and sizes of attack profiles that should be inserted into a system. The results are presented in Figure 5.22, where maximum ROI according to (5.6), averaged over attacks carried out against all items that are contained in d_1, is plotted against attack cost (invested capital). For the purposes of this experiment, a unit cost is applied to all items. Each point on the curve reflects the optimum return that could be achieved for the given attack cost. For example, an attack cost of \$100 could relate to 4 attack profiles, each consisting of 25 items, or to 1 attack profile, consisting of 100 items, etc. depending on which combination achieved the greatest return.

It can be observed from the figure that the maximum ROI peaked for an attack cost of \$7. This particular attack related to the insertion of 1 attack profile, consisting of 7 items (including the targeted item), into the system. At lower attack costs, i.e. when attack profiles were created with less than 7 items, the number of co–rated items between genuine and attack profiles was reduced to the extent that significance weighting served to minimise the influence of the attack profiles. It is also apparent from the figure that

[9] Recall that a good prediction is defined as either r_{max} or $r_{max} - 1$.

the maximum ROI decreased significantly for attack costs in excess of $25, where any additional returns were more than offset by the increasing attack costs. These results represent a real cause of concern for system designers, since they indicate that cheap attacks, within the specified cost limits, are the most effective. In addition, the effort required on the part of an attacker to create just one, small–sized, attack profile is minimal. Thus, armed with this knowledge, malicious persons may well be inclined to implement such attacks, given the low risks involved.

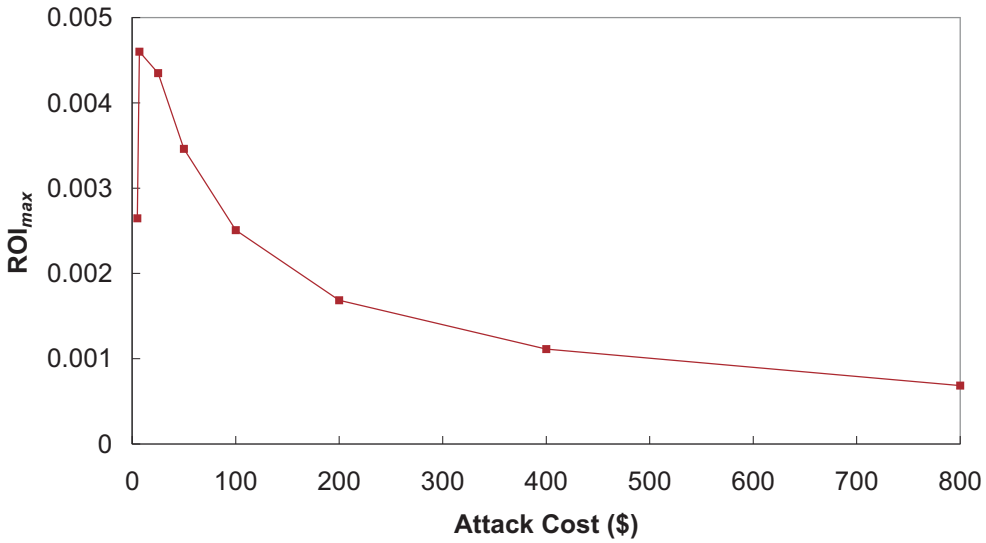


Figure 5.22: The maximum ROI, averaged over attacks carried out against all items that are contained in the EachMovie sample dataset d_1, versus attack cost

We now examine the effect of mounting an attack on 10 different–sized samples that are drawn from the EachMovie dataset. The attack strategy that is used against all the sample datasets is to create and insert 1 attack profile, which consists of a total of 25 items. While the maximum ROI that is associated with this attack is sub–optimal (refer to Figure 5.22), it is, nevertheless, significant. In addition, attack profiles of greater size are more likely to be influential against larger datasets, given that greater numbers of high quality (genuine) neighbours are likely to be present in such datasets.

The results for the attacks are presented in Table 5.9. For each of the datasets, the neighbourhood size that provided the optimum performance in terms of predictive accuracy (when no attack was present) was selected. These neighbourhood sizes are shown in the table and, as expected, it can be observed that the optimum neighbourhood size generally increased as the systems grew in size.

It is reasonable to assume that newer items are likely to be the subject of most attacks, given that these items often sell at premium rates and furthermore, the predictions that

Table 5.9: Focused Push Attacks carried out on 10 samples drawn from the EachMovie dataset. In each case, attacks are based on the insertion of just 1 attack profile, consisting of 25 items. The results shown are averaged over attacks carried out against all items with a popularity of ≤ 100 that are contained in the respective datasets.

Dataset Sample	# Users	Neighbourhood Size	% Items with Pop. ≤ 100	$\bar{g}$	$\bar{g}'$
d_1	1,000	45	88	0.24	0.37
d_2	2,000	40	79	0.24	0.35
d_3	3,000	40	72	0.22	0.31
d_4	4,000	50	67	0.20	0.29
d_5	5,000	55	64	0.19	0.29
d_6	6,000	65	60	0.17	0.25
d_7	7,000	70	57	0.16	0.24
d_8	8,000	80	55	0.15	0.24
d_9	9,000	90	54	0.17	0.25
d_{10}	10,000	95	52	0.17	0.25

are made for such items are easier to manipulate (Section 5.4.4). The results that are presented in the table therefore relate to those items which have received ≤ 100 ratings in the various sample datasets. For each of the datasets, the percentage of items with a popularity of ≤ 100 is also shown. While these percentages decreased as the sample datasets grew in size, over 50% of all items contained in the largest sample that was used still satisfied this criterion.

The average probabilities of good prediction that were achieved pre– and post–attack ($\bar{g}$ and $\bar{g}'$, respectively) were seen to decrease as the sample datasets grew in size. It is important to note, however, that the differences between the pre– and post–attack probabilities of good prediction did not continue to decrease, and stabilised at a value of approximately 0.08 for dataset sizes of $6,000$ and above.

This data can be used to estimate the profit that accrued from the attacks, where the profit that is realised for an attack on an item i is given by $\text{Gains}(i) - \text{Invested Capital}(i)$. The average profit, calculated over attacks that are carried out against all items with a popularity of ≤ 100, is shown for each of the 10 datasets in Figure 5.23. The results that are presented correspond to a browser–to–buyer conversion rate of 10%. In addition, a unit cost is assigned to all items.

Of critical importance is the fact that profit was found to increase with dataset size, since more and more users became available to be targeted by the attack. While financial losses actually resulted for the attacks against datasets of size $2,000$ or less, thereafter profits were in the black. For example, a profit of \$50.13 was realised for an attack on the largest

sample dataset, which contained 10,000 users. Increased profits were realised at higher conversion rates.

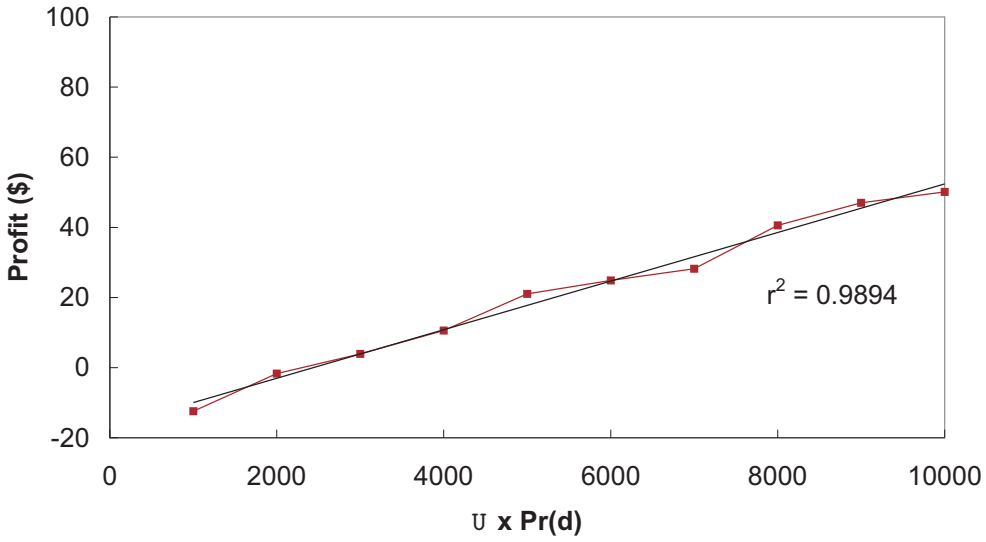


Figure 5.23: The average profit that was achieved when each the 10 sample datasets were subjected to a product push attack, which was based on just 1 attack profile consisting of 25 items

The next step in the analysis is to extrapolate these results out to systems of greater size. As can be seen from Figure 5.23, the relationship between profit and dataset size appears to be linear, with $r^2 = 0.9894$. Accordingly, it is possible to estimate the profit that can be realised when larger systems are attacked. Figure 5.24 shows the minimum system size, in terms of the number of users that access a system, that is required to provide a zero profit margin for different values of the browser–to–buyer conversion rate. Thus, systems must be at least the indicated sizes before attacks become financially worthwhile. At higher conversion rates, the zero profit margin was achieved at smaller system sizes. For example, a minimum size of 16,700 was required for a conversion rate of 2%, compared to a size of 2,450 for a conversion rate of 10%.

Projected profit versus system size in shown in Figure 5.25. For large systems, very significant profits were realised by the attack, particularly at the higher browser–to–buyer conversion rates. For example, consider the results for a system containing 5 million users, where profit increased from \$17,479 to \$34,383 when the conversion rate was increased from 5% to 10%. Note that these results correspond to unit costs being assigned to all items – thus, actual profits would be multiples of these amounts. Furthermore, recall that a number of unfavourable assumptions (from an attackers perspective) were made at the beginning of this analysis. Firstly, it was assumed that identical costs applied for all items that are contained in a system. However, new items, which are likely to be the

focus of most attacks, often sell at premium rates. Thus, attackers could expect to realise even greater profits than those that are recorded above. Secondly, it should be noted that it is often unnecessary to make an actual purchase of an item before ratings can be inserted into a system. In such cases, the cost associated with any attack is independent of financial concerns and relates only to the amount of effort required to insert attack ratings, bandwidth considerations, etc.

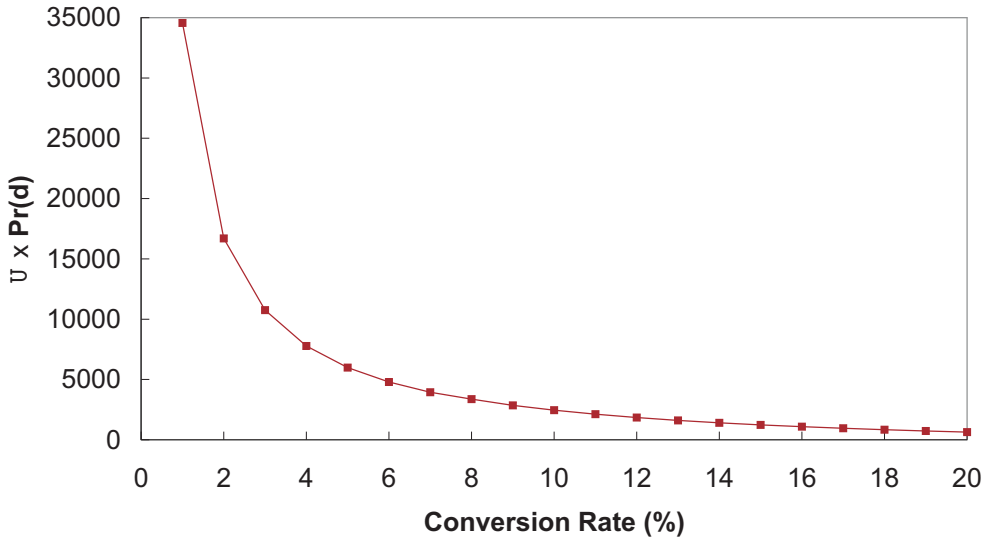


Figure 5.24: The minimum system size that is required to provide a zero profit margin versus browser–to–buyer conversion rate

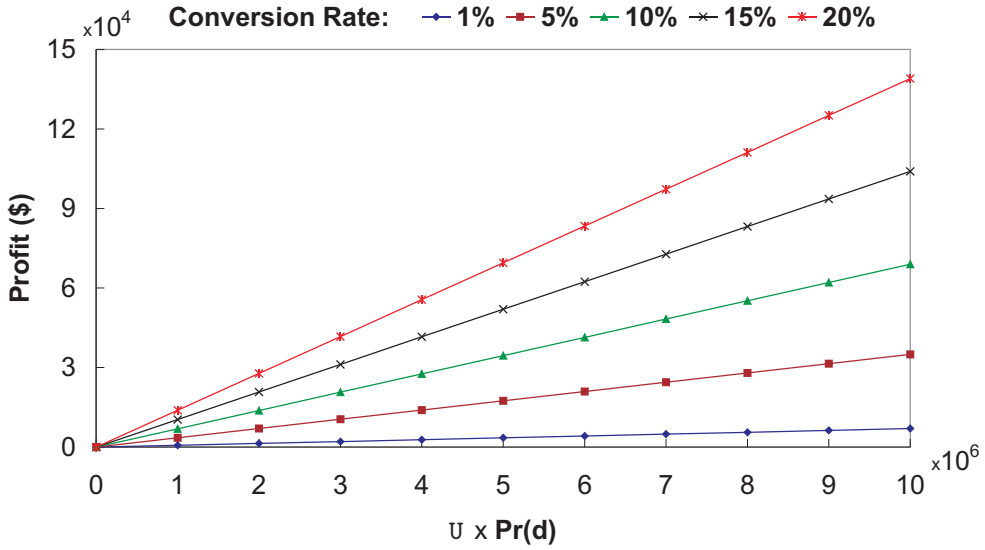


Figure 5.25: Projected profit versus system size, shown for various browser–to–buyer conversion rates

With this latter assumption in mind, an experiment is performed using the smallest sample that is drawn from the EachMovie dataset.[10] In this experiment, the relationship between the percentage of good predictions that are achieved and the quantities and sizes of the attack profiles that are inserted is examined. The results are shown in Figure 5.26, which indicate, not surprisingly, that the highest gain was achieved by inserting large quantities of the biggest attack profiles. It is also apparent, however, that little additional gain was achieved by inserting more than 16 attack profiles into the system. In addition, an attack profile size (APS) of greater than 200 did not yield significant extra gains. Thus, if financial costs are *not* associated with the insertion of ratings into a system, and if other attack costs are not considered, then the best strategy for an attacker would be to create 16 attack profiles, each containing 200 items. The key point to note is that much more substantial gains can be achieved by attacking even small–sized systems when users are permitted to insert ratings free–of–charge. Consequently, we believe that, in the real–world, recommender systems ought to place at least some restrictions on rating entry in order to provide a greater degree of robustness against attack.

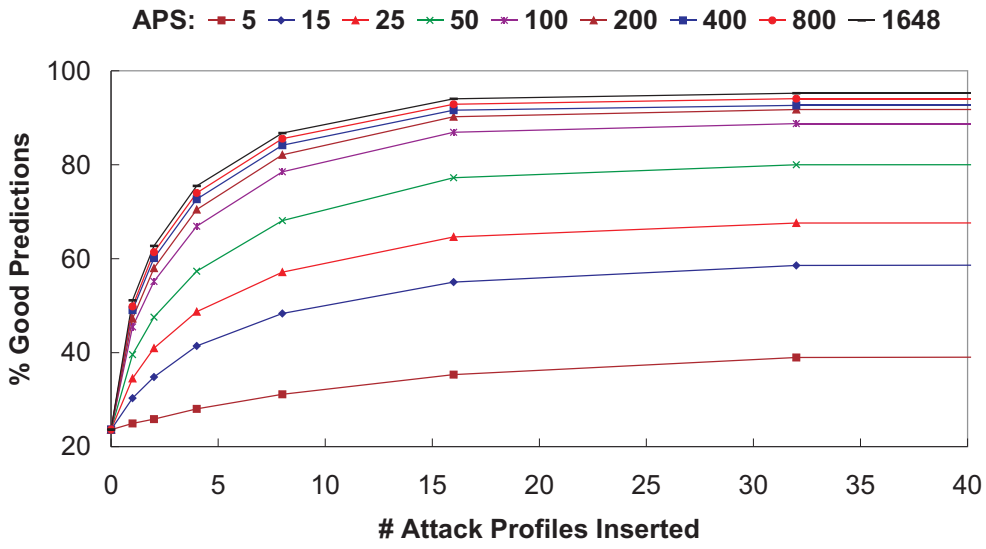


Figure 5.26: The gain in the percentage of good predictions that was achieved, using the smallest sample drawn from the EachMovie dataset, when attack profiles of various quantities and sizes were inserted

It is clear from the results that are presented in this section that attacks are capable of resulting in substantial profits for attackers. While it is true that the above analysis is based on samples that are taken from only one dataset, there is little doubt that similar trends would indeed apply to the other datasets that are considered in this thesis – as has been shown in previous sections of this chapter, all of the datasets that were evaluated exhibited a distinct lack of robustness against attack.

[10] This particular sample dataset consists of 1,000 users.

5.10 Summary

In this chapter, attack strategies were developed which succeeded in significantly compromising the robustness of the three real–world datasets that were evaluated. It was found that the strategy as described in Section 5.4.3 achieved the greatest success, and remained successful when only partial item likability and item popularity knowledge was available to an attacker. None of the algorithm extensions that were evaluated proved robust against attack, although significance weighting was beneficial against the smallest attack profiles that were considered.

In addition, it was shown that, while some differences in performance were seen among the various similarity metrics and neighbourhood formation schemes that were examined, none of the possible combinations achieved an acceptable degree of robustness against attack when configured to provide good performance across other dimensions. Critically, it was demonstrated in Section 5.9 that substantial profits can be realised by attackers, even when systems impose financial costs on the insertion of attack data.

To conclude, the analysis that is presented in this chapter has revealed the significant vulnerability of the widely used class of collaborative filtering algorithms that has been considered in this work. Of course, many other approaches to collaborative filtering have also been proposed in the literature, e.g. LSI and trust–aware filtering schemes. While some of these approaches may offer a greater degree of protection against attack, our experience leads us to believe that attack strategies could be devised in order to also compromise the integrity of these other systems.

Chapter 6

Preventing Attack

6.1 Introduction

In this chapter, the objective is to develop techniques to defend against the very successful attack strategies that were outlined in Chapter 5. Two different algorithms are proposed, both of which operate by attempting to identify and exclude from neighbourhoods any attack profiles that may be present in a system (O'Mahony *et al.*, 2004b, c).

The first approach is based on related work from the field of reputation reporting systems. These systems aim to provide reputation estimates for buyers and sellers who are engaged in on–line marketplaces, and need to be sufficiently robust to deal with any malicious agents who may attempt to interfere with the system. In our related approach, neighbourhoods are first formed in the usual manner and are then clustered into two groups. An analysis is performed based on cluster statistics to determine whether or not attack data is present and, if so, which of the clusters contain this data. We show that this approach provides improved robustness against attack, but is susceptible to informed attacks where an attacker has knowledge of the underlying algorithm.

The second algorithm that is proposed introduces novel approaches to neighbourhood formation. In place of traditional similarity–based neighbourhoods, the concept of *profile utility* is introduced, which evaluates the usefulness of potential neighbours based on certain characteristics of the items that have been rated by each. In addition, similarity weight transformations are proposed which are designed to protect the predictions that are made for new or infrequently rated items. With this approach, we show that the cost of mounting successful attacks can be made prohibitive, thereby removing the motivation for those who may be inclined to carry out such attacks. In addition, our scheme provides predictive accuracy and coverage comparable to that of a benchmark k–NN algorithm, while suffering none of scalability problems that are associated with this latter technique.

6.2 Neighbourhood Filtering

The attacks on the collaborative filtering algorithms as presented in Chapter 5 were successful because it was possible to create attack profiles which had a high probability of being included in the neighbourhoods of many genuine users. One possible approach to defend against attack is to devise mechanisms for identifying and filtering any attack profiles that may be present in neighbourhoods. In this regard, related work from the field of reputation reporting systems (Zacharia *et al.*, 1999; Resnick *et al.*, 2000) is of interest.

Reputation systems facilitate buyers and sellers by enabling each to obtain reputation estimates for potential business partners. These systems are particularly useful to members of electronic communities, where trust between members can be difficult to establish. This is especially true in situations where members are unrelated to each other, have never met and where people are unlikely to interact with each other on a regular basis. For example, it is reasonable to assume that the majority of eBay's users rarely have multiple transactions with the same buyers and sellers.

Reputation systems have to be sufficiently robust in order to ensure protection against any malicious behaviour that may be carried out by community members. For example, in an effort to increase sales, buyers and sellers may attempt to inflate or reduce their own or others' reputation by inserting bogus feedback into a system. Here, we focus on one such reputation system that has been developed by Dellarocas (2000), and we show how this system can be adapted to facilitate robust collaborative filtering.

The goal of the system developed by Dellarocas (2000) is to enable a buyer, b, to obtain a robust reputation estimate for a particular seller, s, based on the previous ratings given by other buyers for s. (In this way, the reputation system is directly analogous to our work, with buyers taking on the role of users and sellers taking on the role of items.) In situations where legitimate differences in opinion can exist among buyers, four attack scenarios are identified where buyers and/or sellers can seek to bias the reputation estimates that are calculated by a system:

(a) **Ballot stuffing**. A group of buyers, in collusion with a particular seller, assign biased high ratings to the seller, thereby falsely promoting the seller's reputation.

(b) **Bad–mouthing**. Buyers, in collusion with sellers, assign biased low ratings to targeted sellers in order to unfairly demote reputations causing them to be driven from marketplaces.

(c) **Negative discrimination**. Sellers provide poor quality service to a targeted subset of buyers, while offering good service to others. If the number of targeted buyers is small, the cumulative reputation of sellers will remain high and the victimised buyers are externalised.

(d) **Positive discrimination**. Sellers provide average quality service to the majority of buyers, but provide excellent service to a select few. If the latter group is of a sufficient size, the positive ratings that are received from these buyers can promote the sellers' reputation.

The reputation system that is proposed by Dellarocas (2000) develops two approaches to deal with the above attacks. By enforcing a controlled anonymity scheme for marketplaces, where the identities of buyers and sellers are concealed, bad–mouthing and negative discrimination can be prevented, since these attacks depend on the ability to identify and demote the reputations of a small, specific subset of users. A number of issues, relating to the successful enforcement of controlled anonymity, are explored in the above work. These are not relevant to the robustness of recommender systems, however, and thus, are not considered further.

The second approach is designed to deal ballot stuffing and positive discrimination, which cannot be avoided through the use of controlled anonymity. The first step is to form a neighbourhood of the most similar buyers to b that have given a rating for s. These ratings are then used to form a reputation estimate for s. The next step is to *filter* the neighbourhood and remove any possible malicious ratings that may be present. Since controlled anonymity removes the possibility of any biased low ratings being present, the system needs only to consider biased high ratings. Thus, filtering takes place by clustering the ratings for s into two groups, and the group with the highest mean is assumed to contain the biased ratings. Filtering is performed using the divisive clustering algorithm proposed by Macnaughton-Smith *et al.* (1964) – see Appendix D for details. It should be noted that filtering is performed in all cases – no attempt is made to determine whether any malicious ratings are actually present in the neighbourhood. In the final step, a reputation estimate based on those ratings for s that are contained in the genuine group only is calculated.

6.2.1 Robust Collaborative Filtering

With regard to the robustness of collaborative filtering, the above approach is adopted with the following modifications[1]. To begin, it is not automatically assumed that biased ratings are present in a system. The modified algorithm first attempts to discover an attack 'signature', and filtering is only performed when such a signature is deemed to be present. Since the goal of an attacker is to force the predicted ratings of targeted items to a particular value, it is reasonable to expect that the ratings for targeted items that are contained in any attack profiles would be centered on the attack value, which is likely to

[1]For convenience, an overview of the algorithms that are used in the reputation reporting system proposed by Dellarocas (2000) and the derived collaborative filtering approach is presented in Table 6.1.

Table 6.1: Comparison between the approach used in the Reputation System Algorithm of Dellarocas (2000) and the derived Collaborative Filtering Algorithm.

Reputation System Algorithm	**Collaborative Filtering Algorithm**
Buyer b obtains a reputation estimate for a seller s as follows: • Form a neighbourhood of the most similar buyers to b who have rated s • Cluster the neighbours, based on the ratings for s, into two groups • Discard the group with the highest mean rating • Form a reputation estimate for s using the remaining group	User a obtains a prediction for an item j as follows: • Form a neighbourhood of the most similar users to a who have rated j • Cluster the neighbours, based on the ratings for j, into two groups • If attack signature is deemed to be present, discard the group with the smallest standard deviation across the item ratings • Calculate a prediction for a using non–bogus ratings

deviate significantly from the distribution of the genuine neighbours' ratings. Thus, the attack signature that is used is the difference in the mean ratings of the item, for which a prediction is being sought, between the two extracted clusters. It is determined that an attack has taken place only if the difference in cluster means is significant in comparison to that of an un–attacked system. This significance is evaluated empirically. A limitation of this particular approach is that it necessarily requires knowledge of the distribution of the difference in cluster means of an un–attacked system.

The next step is to a select a threshold value, with respect to the difference in cluster means, above which an attack is considered to have taken place. From the algorithm designer's perspective, the goal is to carry out sufficient filtering to neutralise the effect of any attack profiles that may be present in the system, without removing so many genuine users that the system competence is decreased. It was established in Section 5.4.3, however, that the presence of only small numbers of attack profiles in neighbourhoods had a substantial impact on system robustness. Thus, a conservative approach is required and a low threshold value is selected above which an attack data is considered to be present. For the MovieLens dataset, it was found that a threshold value of 2 was the most suitable in terms of correctly detecting whether or not an attack had occurred while maintaining acceptable system accuracy.

Once an attack is deemed to have occurred, there is a further need to determine the nature of the attack that has occurred. In the context of recommender systems, it cannot be

Figure 6.1: Ratings for an item j clustered into two groups using the Macnaughton–Smith clustering algorithm. If an attack signature is deemed to be present, the cluster with the lowest standard deviation is assumed to contain the attack ratings.

assumed that only a particular type of biased ratings are present in a system[2]. If the set of attack types is limited to product push and nuke attacks, a suitable heuristic is that the cluster containing the attack profiles is the one with the lowest standard deviation across the ratings for the item for which a prediction is being sought. The motivation behind this approach is that, for example, in a product nuke attack, any biased ratings that are inserted into the system are likely to be consistently low (Figure 6.1). It should be observed, however, that a misclassification of the attack cluster would result in genuine neighbours being excluded from the prediction generation process, with predictions thus being made from biased data.

6.2.1.1 Evaluation

In order to evaluate the neighbourhood filtering approach as outlined above, the MovieLens dataset was subjected to a product nuke attack as described in Section 5.6. The performance of the filtering algorithm for Cosine similarity is shown in Figure 6.2 (similar trends were observed for the correlation similarity metrics). It can be seen from the figure that the approach was generally quite successful, particularly at the higher attack strengths. For example, for all attack strengths greater than 8 attack profiles inserted, the approach correctly determined whether or not an attack had occurred in 4 out of 5 cases. The approach was equally successful in correctly identifying the cluster containing the attack profiles. In addition, it can be seen that the clusters were principally formed from the correct type of users – i.e.. from either genuine or attack users – with genuine

[2]In contrast to the reputation system described by Dellarocas (2000), where the identities of buyers and sellers are concealed from each other, all of the items that are contained in a recommender system are typically made known to all the users of the system, although the identity of users is often protected.

users forming upward of 75% of all clusters identified as those that should contain genuine users.

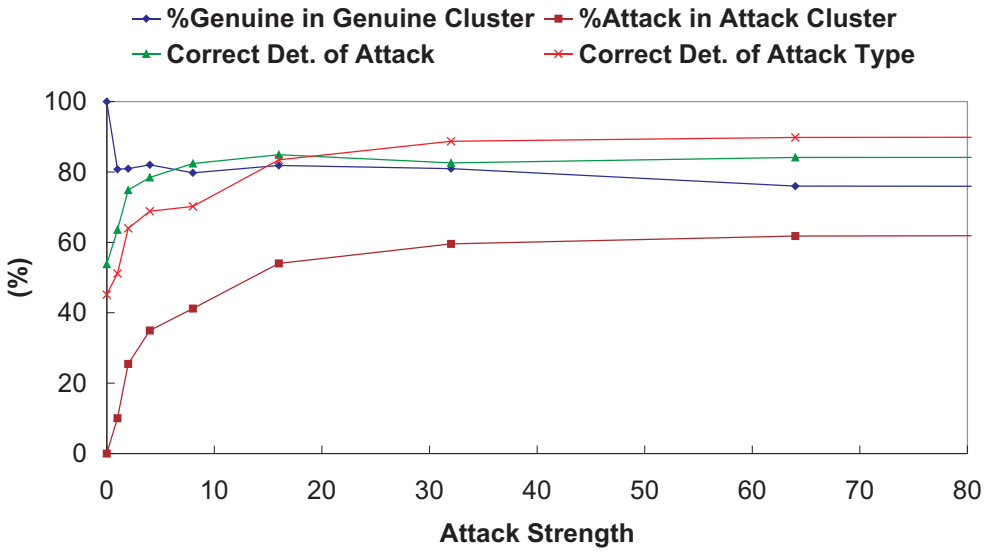


Figure 6.2: Performance of the neighbourhood filtering algorithm for the MovieLens dataset using the Cosine similarity metric with significance weighting

Figure 6.3 shows the sizes of the identified genuine and attack clusters against attack strength. The results indicate that the size of the genuine cluster decreased and that of the attack cluster increased as more attack profiles were inserted into the system. These findings were to be expected, since the number of attack profiles that were contained in the unfiltered neighbourhoods grew with increasing attack strength. For example, 57% of all neighbourhoods were formed by attack profiles at an attack strength of 64, compared to 8% at an attack strength of just 1 attack profile inserted into the system.

Further results for the product nuke attack are presented in Table 6.2. From this table, the effect of neighbourhood filtering on system robustness and accuracy can be readily appreciated, irrespective of which similarity metric is used by the algorithm. Consider, for example, Cosine similarity, where the percentage of bad predictions achieved pre– and post–attack actually fell from 40% to 38% when neighbourhood filtering was applied, indicating that the attack had certainly failed to nuke targeted items, compared to corresponding values of 39% and 78% when filtering was not applied. Significant improvements in performance were also seen for the other similarity metrics. Thus, it can be stated that the neighbourhood filtering approach was generally successful in correctly identifying attack profiles and thereby enabling them to be removed from the prediction generation process, resulting in a significantly more robust algorithm.

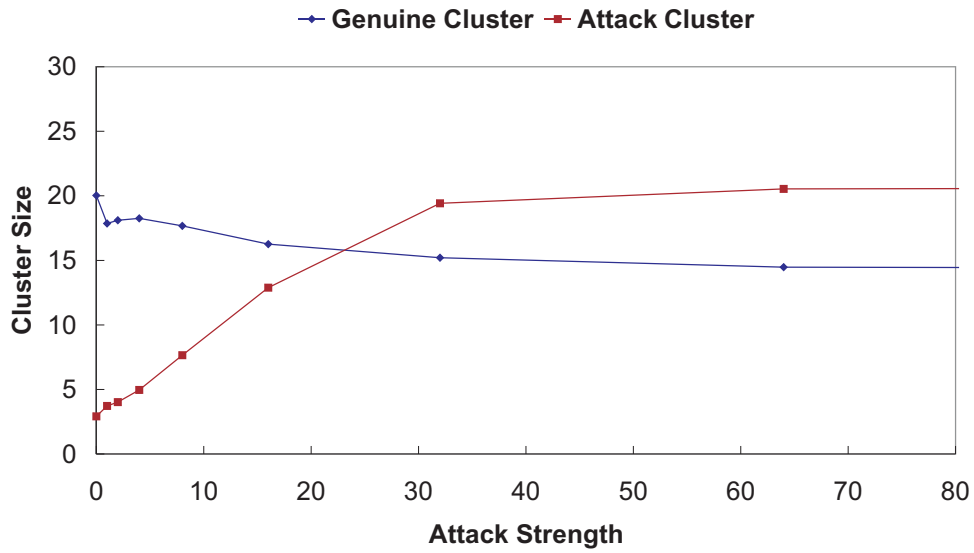


Figure 6.3: The sizes of the genuine and attack clusters, as identified by the neighbourhood filtering algorithm, for the MovieLens dataset using the Cosine similarity metric with significance weighting

As stated above, it is important that any modifications made to the collaborative filtering algorithm do not result in significantly reduced overall performance. While neighbourhood filtering reduced the predictive accuracy for all similarity metrics when no attack was present, the accuracy achieved was still superior to that of the non–personalised benchmark algorithm. When Cosine similarity was used, for example, neighbourhood filtering provided a mean absolute error of 0.8837 compared to 0.7987 when filtering was not performed and 0.9364 for the non–personalised algorithm. It should be observed, however, that the post–attack accuracy provided by neighbourhood filtering did in fact deteriorate beyond that provided by the benchmark algorithm. Nevertheless, the post–attack accuracy performance was significantly better compared to the when filtering was not applied.

6.2.2 Robustness Against Informed Attacks

It is possible, however, that a more sophisticated attacker could attempt to defeat the neighbourhood filtering algorithm. In general, a truly robust system should be robust even when an attacker is fully informed about the underlying algorithm. Given that filtering based on the means and standard deviations of the ratings assigned to targeted items is being performed, an attacker can tailor the ratings distribution of the attack profiles so that they are less likely to be successfully filtered. Thus, from the attacker's perspective, the goal is to construct a ratings distribution so that attack profiles are not identified by the filtering system, but which still bias the predictions that are made in the desired

Table 6.2: Results for three Focused Nuke Attacks (using large attack profiles) carried out on the MovieLens dataset. Results shown are for an attack strength of 64 attack profiles inserted into the system. Accuracy (MAE) computed according to the non–personalised benchmark algorithm is 0.9364. Any MAE values that exceed this baseline are shown in boldface.

	% Neg. Shifts	% Attack Profiles in Neighbourhoods	MAPE	% Bad Preds. (pre)	% Bad Preds. (post)	MAE (pre)	MAE (post)
Product Nuke Attack							**Filtering Not Applied**
Pe+sig	85	75	2.3111	36	83	0.7874	**2.3969**
Sp+sig	84	75	2.3108	36	83	0.7895	**2.3950**
Co+sig	80	57	1.6510	39	78	0.7987	**1.8447**
Product Nuke Attack							**Filtering Applied**
Pe+sig	57	44	0.9179	37	44	0.8746	**1.2867**
Sp+sig	57	44	0.9184	37	44	0.8772	**1.2859**
Co+sig	35	17	0.8167	40	38	0.8837	**1.2037**
Informed Product Nuke Attack							**Filtering Applied**
Pe+sig	74	67	1.5011	37	70	0.8746	**1.6797**
Sp+sig	74	67	1.5025	37	70	0.8772	**1.6795**
Co+sig	63	46	1.0937	40	59	0.8837	**1.3840**

manner.

Following the above discussion, a revised product nuke attack is performed on the MovieLens dataset where it is assumed that the attacker has complete knowledge of the filtering algorithm. Attack profiles are constructed as before, except, in this case, the ratings assigned to targeted items follow the distribution as shown in Figure 6.4, rather than being uniformly assigned the minimum rating of 1. This particular ratings strategy is designed to have multiple effects on the filtering algorithm. In the first instance, it may cause the algorithm to fail to detect the attack since the difference in cluster means is likely to be reduced; in addition, if the attack is detected, then the probability of the correct classification of cluster types is likely to deteriorate, since the standard deviation of the biased ratings has been increased.

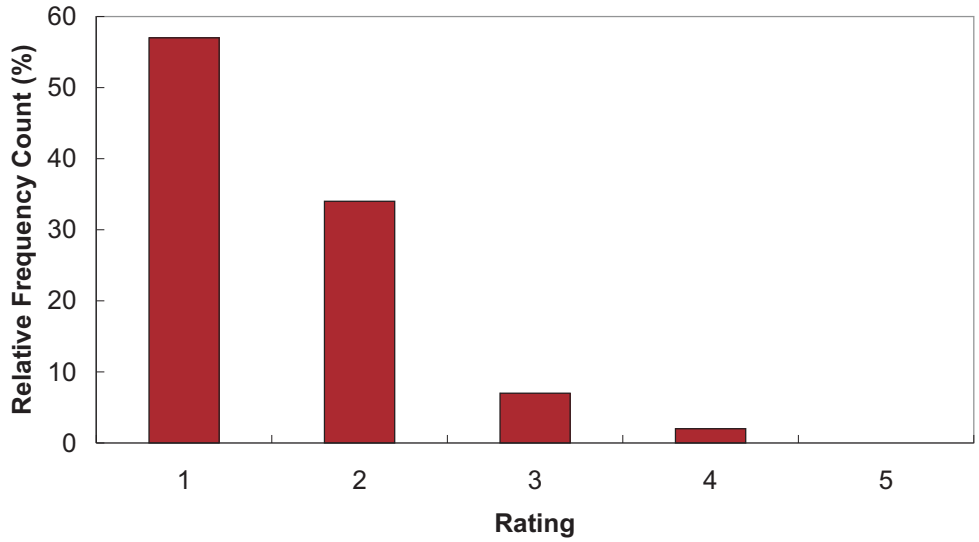


Figure 6.4: Ratings distribution of targeted items for the informed product nuke attack, with $\mu = 1.54$ and $\sigma = 0.72$

6.2.2.1 Evaluation

The performance provided by the filtering algorithm against the informed attack is shown in Figure 6.5. The results presented are for Cosine similarity; similar trends applied for the correlation metrics. As can be seen, the effectiveness of the algorithm has been significantly compromised by the attack. For example, at an attack strength of 32 attack profiles inserted, the rate of correct detection of attack has occurred and the the rate correct identification of attack cluster have both decreased by approximately 20% to values of 60%. In addition, the composition of the clusters as identified by the algorithm has deteriorated, with only 52% and 31% of the correct type of profiles (i.e. genuine or attack profiles) being contained in the genuine and attack clusters, respectively. Figure 6.6, which shows the sizes of the identified genuine and attack clusters against attack strength, also indicates a deterioration in algorithm performance. As stated above, the number of attack profiles that were present in unfiltered neighbourhoods grew with increasing attack strength. This trend, however, was not reflected in the results, where in fact the size of the cluster identified as containing the genuine profiles continued to grow as the attack strength was increased.

Referring to Table 6.2, the robustness metrics indicate that the informed attack was indeed successful in nuking the predictions made for targeted items. For example, the percentage of bad predictions achieved pre– and post–attack were 40% and 59% for Cosine similarity, respectively. The corresponding values for both of the correlation metrics were 37% and 70%, respectively. The post–attack accuracy was also significantly worse than that provided by the non–personalised algorithm. While these results represented an

improvement over the case where neighbourhood filtering was not applied, the filtering algorithm did not prove to be robust against this attack.

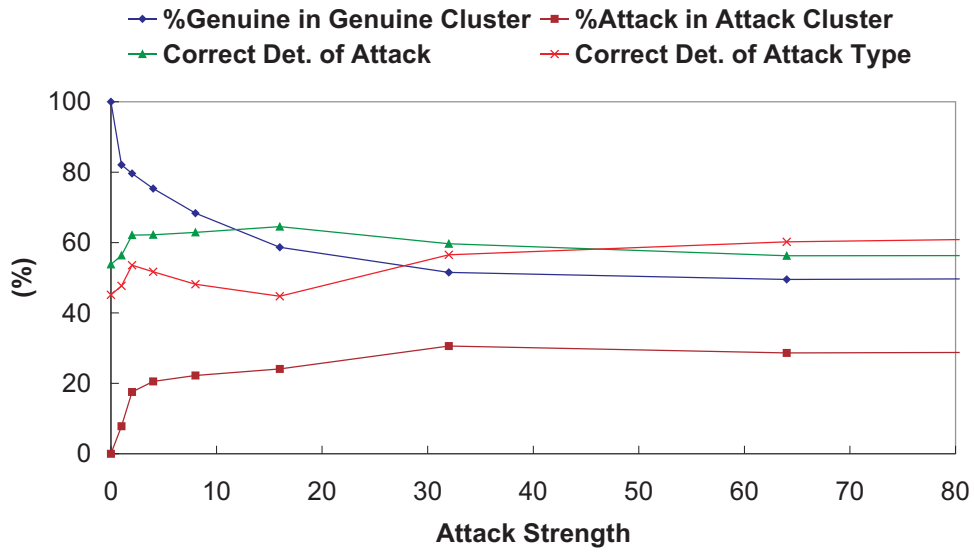


Figure 6.5: Performance of the neighbourhood filtering algorithm for the informed product nuke attack carried out the MovieLens dataset using Cosine similarity with significance weighting

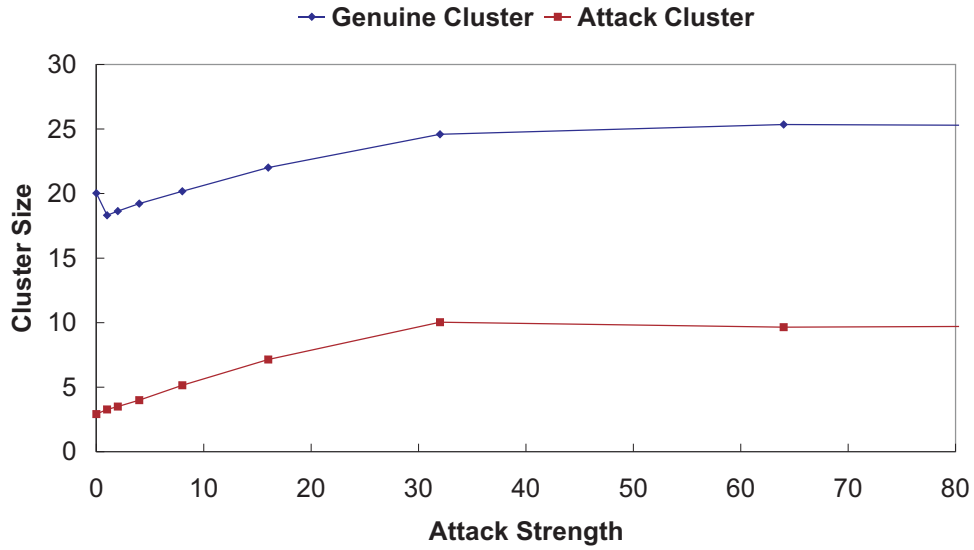


Figure 6.6: The sizes of the genuine and attack clusters, as identified by the neighbourhood filtering algorithm, for the informed product nuke attack carried out the MovieLens dataset using Cosine similarity with significance weighting

Thus, it can be concluded that while neighbourhood filtering based on the means and standard deviations between clusters provided some degree of protection against attack,

it did not provide complete robustness, and was especially vulnerable to sophisticated attackers armed with knowledge of the underlying algorithm. In the next section, a new approach is outlined which radically changes the neighbourhood formation process, and leads to a robust, accurate and efficient collaborative filtering algorithm.

6.3 Intelligent Neighbourhood Formation

A key element of collaborative filtering algorithms is the user–user similarity measurement. It is typically used for two separate purposes:

(a) it is used to select the set of neighbours whose ratings are combined to form the prediction – the most similar users are selected, and

(b) it is used to form the weights in the computation of the prediction as a weighted average of the neighbours' ratings.

It is important to distinguish between these two stages of the algorithm and we argue that different approaches can be appropriate for each. In particular, the goal of neighbour selection is to identify the most useful neighbours on which to base predictions. While similarity is one measure of usefulness when considering the accuracy of the algorithm, the notion of neighbour *utility* can be extended to include other performance measures such as efficiency and robustness. For example, the on–line efficiency of collaborative filtering algorithms is directly proportional to the number of similarity measurements that are made. To form a neighbourhood based on similarity, the similarity between *all* potential neighbours (i.e. all users who have rated the item for which a prediction is being sought) must be calculated. Neighbour selection that is based on a less costly utility measure may lead to improved efficiency performance.

Figure 6.7 shows the alternative utility–based approach to neighbourhood formation, where neighbourhoods that are formed on the basis of utility criteria may not necessarily include the most similar users. One utility criterion, proposed by O'Donovan and Smyth (2005) and Massa and Avesani (2004), employs the concept of trust between users to augment the neighbour selection process. In other work, clustering (Sarwar *et al.*, 2002) and dimension–reduction (Sarwar *et al.*, 2000b; Canny, 2002) have been proposed to improve the efficiency of the neighbourhood formation process. In the next section, a novel utility–based approach to neighbourhood formation is described, which enables efficient, accurate, scalable and robust collaborative filtering.

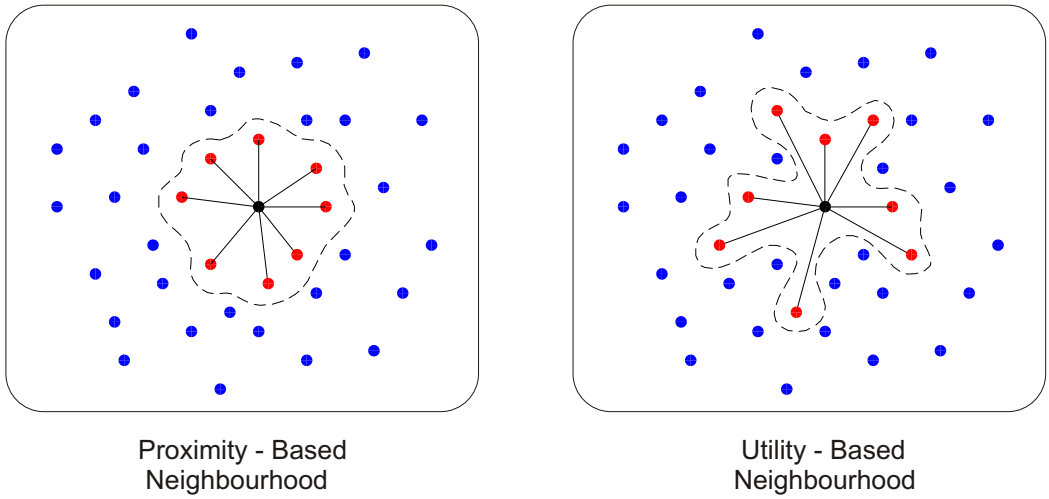


Figure 6.7: Different approaches to neighbourhood formation. Proximity–based neighbourhoods are formed by selecting the most similar users. Utility–based neighbourhoods are formed using alternative criteria to model neighbour usefulness.

6.3.1 Profile Utility

It is well recognised that frequently occurring words are not a good basis for calculating the similarity between documents in information retrieval. In Section 2.2.5, the analogous inverse user frequency approach, which was proposed by Breese *et al.* (1998) for collaborative filtering algorithms, was described. This approach did not, however, prove successful in dealing with attack profiles constructed using popular items (Section 5.5).

Here, we go a step further by proposing a *profile utility* measure that is based on certain characteristics of the items that are contained in a profile. Each user profile is assigned a global weight that models that profile's usefulness as a potential neighbour. Users with the greatest utilities, who have rated the item for which a prediction is being sought, are then selected as neighbours. In general, the profile utility of user a is defined as follows:

$$U(a) = f(j_1, j_2, \ldots, j_n) \tag{6.1}$$

where $j_i \in I_a$ is the set of items user a has rated. To begin, profile utility is first considered in terms of item popularity, with the motivation that users who have rated predominantly popular items are unlikely to serve as good predictors (Rafter and Smyth, 2001). Specifically, utility is defined as the summation of the inverse popularity of items contained in a profile as follows:

$$U(a) = \frac{1}{|I_a|} \sum_{j \in I_a} \text{InvPop}(j) \tag{6.2}$$

The inverse popularity of an item j is defined as:

$$\text{InvPop}(j) = \begin{cases} 1 - \frac{n_j}{m} & \text{if } n_j < m \\ 0 & \text{otherwise} \end{cases} \tag{6.3}$$

where n_j is the number of users who have rated item j and m is a threshold popularity. Profiles containing less popular items are thus assigned a higher weight, and the parameter m can be used to control the set of possible neighbours by assigning a zero contribution from items that exceed the specified popularity threshold. This approach is shown in Figure 6.8. Neighbourhoods are formed by simply selecting k users with the highest utility that contain the item for which a prediction is being sought. Once neighbours have been selected, predictions are calculated as in (2.7).

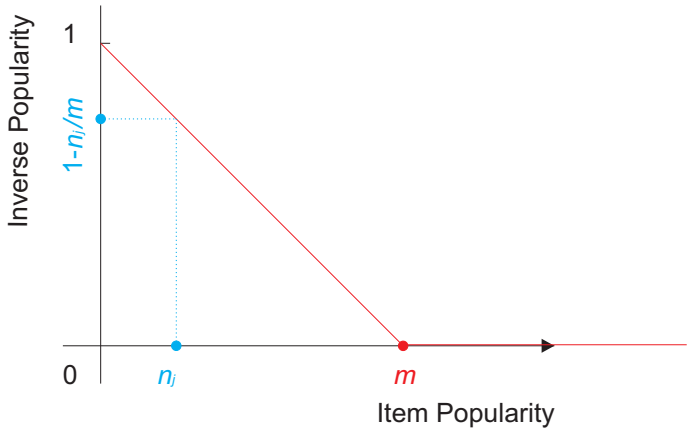


Figure 6.8: The definition of inverse item popularity as a function of the popularity threshold, m. The inverse popularity of an item j is set to 0 if $n_j \geq m$.

In a similar manner, utility can also be defined in terms of item rating distribution. The greater the variation of ratings across items, the more likely it may be that such items might be useful to the prediction process (Rafter and Smyth, 2003). Thus, utility can be defined in terms of the entropy of the set of items contained in a user's profile, where the entropy of item j is defined as:

$$H(j) = -\sum_r p_{r,j} \log_2(p_{r,j}) \tag{6.4}$$

where $p_{r,j}$ is the probability that the rating of item j is r. Profile utility is now defined as:

$$U(a) = \frac{1}{|I_a|} \sum_{j \in I_a} \frac{H(j)}{H(j)_{max}} \tag{6.5}$$

where $H(j)_{max}$ is the maximum entropy of item j which assumes that the distributions over all classes of ratings are identical.

The definition of profile utility may also be extended to encompass other measures of item usefulness. For example, utility may be defined in terms of item likability, or various combinations of item popularity, item entropy and item likability, etc. In addition, thresholds may be applied to all criteria in order to fine–tune the neighbourhood selection process.

6.3.1.1 Evaluation

An evaluation of the above approach is now performed. It is important to establish that any new approach provides reasonable performance across the various criteria – i.e. if an increase in system robustness is brought about by a significant deterioration in predictive accuracy, for example, then such an approach is unlikely to gain widespread acceptance. Thus, the performance of the profile utility algorithm is benchmarked against the k–NN approach, which was shown in Chapter 5 to provide good performance in terms of system accuracy and coverage (though not, of course, in terms of robustness against attack).

6.3.1.2 Accuracy and Coverage

Results are presented for the EachMovie dataset which was subjected to the product push attack as described in Section 5.4.3. In all, 64 attack profiles were inserted into the system, each containing 100 items. Profile utility was defined in terms of item popularity, according to (6.2). Figure 6.9 shows the predictive accuracy performance achieved by the intelligent neighbourhood selection algorithm for the various similarity metrics, against a range of values for the popularity threshold, m. The figure also shows the accuracy provided by the non–personalised algorithm and by the k–NN algorithm[3], corresponding to the case where no attack was present. The results provided by all similarity metrics were significantly better than that provided by the non–personalised algorithm, and were only marginally worse than that provided by k–NN. Performance was seen to improve slightly as the the threshold, m, was increased. This finding was expected, since more of the potential neighbours that were available were excluded at the lowest values of m. Similar trends were observed for system coverage, where the coverage achieved matched that of k–NN for $m \geq 10$, and where only a 1% decrease in coverage was seen with respect to k–NN when $m = 5$.

[3]The results shown for the k–NN algorithm correspond to Spearman rank correlation with significance weighting, which was the best performing similarity metric in terms of predictive accuracy.

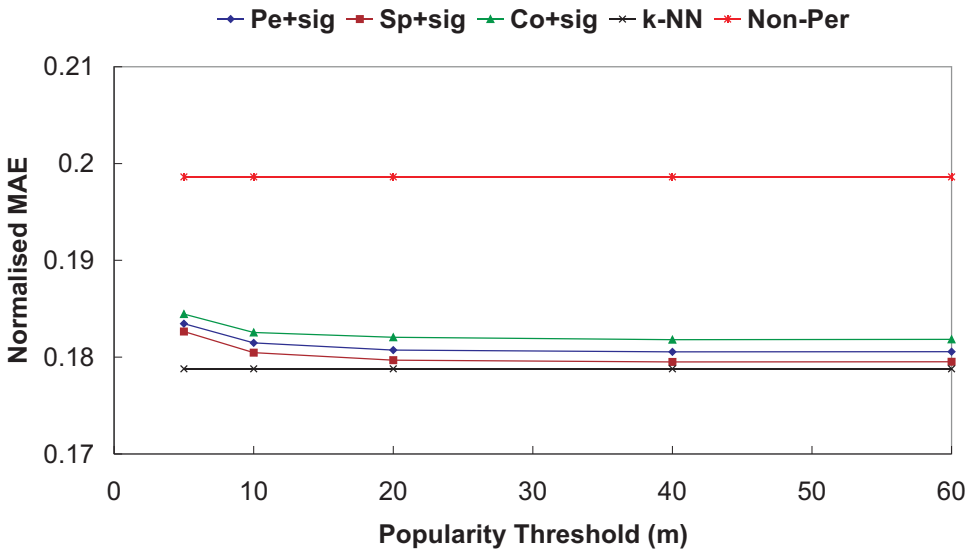


Figure 6.9: The predictive accuracy achieved by the intelligent neighbourhood selection algorithm against the popularity threshold parameter m for the EachMovie dataset.

6.3.1.3 Robustness

Regarding system robustness, the utility–based neighbourhood selection algorithm was completely robust for all values of the threshold shown in the figure (i.e. for $5 \leq m \leq 60$). In contrast, recall from Section 5.4.3 that the percentage of good predictions was increased from 24% to 89% when the k–NN algorithm was subjected to the same attack. The approach was no longer robust, however, for $m > 60$. This was particularly the case when predictions were made for new or infrequently rated items – i.e. where the number of genuine users who have rated such items was less than the neighbourhood size. In these circumstances, attack profiles that were created to target such items began to appear in neighbourhoods as the popularity threshold was increased beyond $m = 60$. Given that new items are likely to be the subject of most attacks and given that these items often sell at premium rates, it is critical that recommender systems are able to provide robust predictions for such items.

6.3.2 Similarity Weight Transformation

Similarity weight transformation schemes have been proposed in the literature in the past, mainly with a view to improving the accuracy performance of collaborative filters. Examples of such transformations include significance weighting and case amplification, which have been described in Section 2.2.5. In this section, we propose a similarity weight transformation scheme from a robustness perspective.

In order to protect the predictions that are made for new or infrequently rated items – where the probability of attack profiles being included in neighbourhoods is high – the collaborative filtering algorithm needs to reduce the similarity weights for any attack profiles that are present to zero. It is only in this manner that the influence of such biased data can be removed, thus permitting robust predictions to be made. These observations motivate the following scheme, in which similarity weights that are calculated using Cosine similarity, Pearson correlation, etc. are transformed as follows:

$$w'(a,i) = w(a,i) \times \frac{1}{|I_a \cap I_i|} \sum_{j \in I_a \cap I_i} \text{InvPop}(j) \tag{6.6}$$

where the inverse popularity of items is calculated as before. By choosing a low value for the popularity threshold m, the transformed weights for attack profiles (that consist primarily of popular items) will approach zero, since the probability of less popular items being co–rated by any two users is, by definition, low. By combining this scheme with the intelligent neighbourhood formation approach that is described above, a system is much more likely to be robust against the type of attacks that are considered in this thesis. As before, weights can also be transformed using item entropy, item likability, or various combinations of these characteristics, etc.

6.3.2.1 Evaluation

In this section, we compare the combined utility–based neighbourhood formation and similarity weight transformation approach against the k–NN algorithm using several performance criteria, and discuss the selection of a suitable value for the popularity threshold value m.

6.3.2.2 Accuracy and Coverage

As stated previously, it is important that any new approach does not have significant consequences for any of the basic performance criteria. Figures 6.10, 6.11 and 6.12 show the accuracy performance of the combined approach for the EachMovie, MovieLens and Smart Radio datasets, respectively, which were subjected to the product push attack as described in Section 5.4.3. The number of attack profiles inserted into the EachMovie and MovieLens datasets was 64. For Smart Radio, given that only 63 genuine users are contained in the dataset, the number of attack profiles inserted was 4. In all cases, the number of items contained in the attack profiles was 100. The levels of coverage achieved for the three datasets are shown in Figures 6.13, 6.14 and 6.15. As before, the accuracy and coverage provided by the k–NN algorithm, corresponding to the case where no attack was present, are presented in order to facilitate comparisons in performance. The accuracy

offered by the non–personalised algorithm is also shown. For each of the datasets, the combined algorithm provided complete robustness over the range of popularity threshold that are included in the figures.

The trends observed for the EachMovie and MovieLens datasets were similar. The accuracy provided by the combined approach was significantly better than that provided by the non–personalised algorithm, except at the lowest value of the threshold shown ($m = 5$). At this point, many users were unable to form neighbourhoods, or only small–sized neighbourhoods, and thus the accuracy and coverage provided by the combined approach was diminished. For both datasets, performance offered by the combined approach approached that of k–NN as the popularity threshold was increased, and it can be seen that comparable performance to that provided by k–NN was achieved when $m \geq 40$. For the Smart Radio dataset, because of its smaller size, a different range of threshold values were used. In this case, accuracy performance was superior to the non–personalised algorithm for all values of the popularity shown. Although zero coverage was provided at the lowest threshold values, i.e. when $m \leq 2$, coverage matched that of the k–NN algorithm when $m \geq 4$.

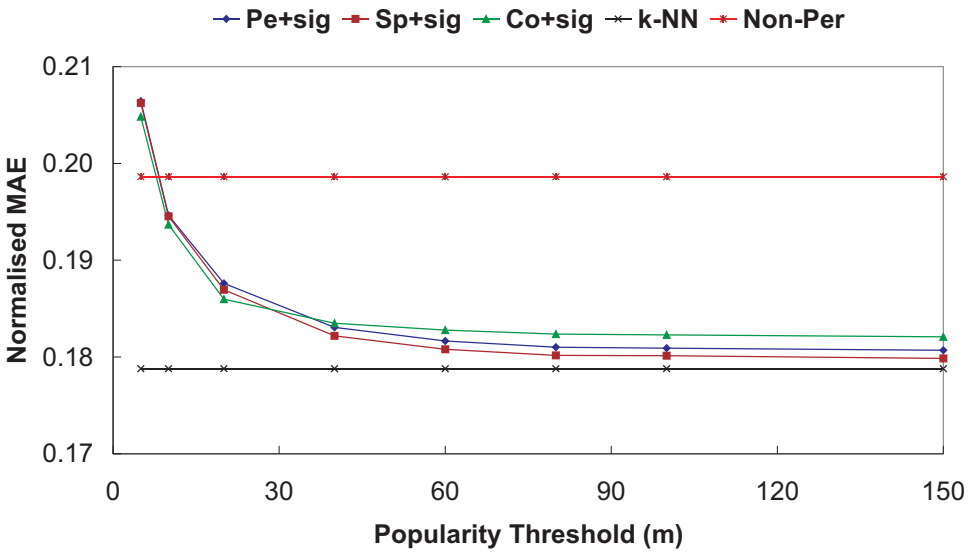


Figure 6.10: The predictive accuracy achieved by the intelligent neighbourhood selection algorithm combined with the similarity weight transformation scheme for the EachMovie dataset.

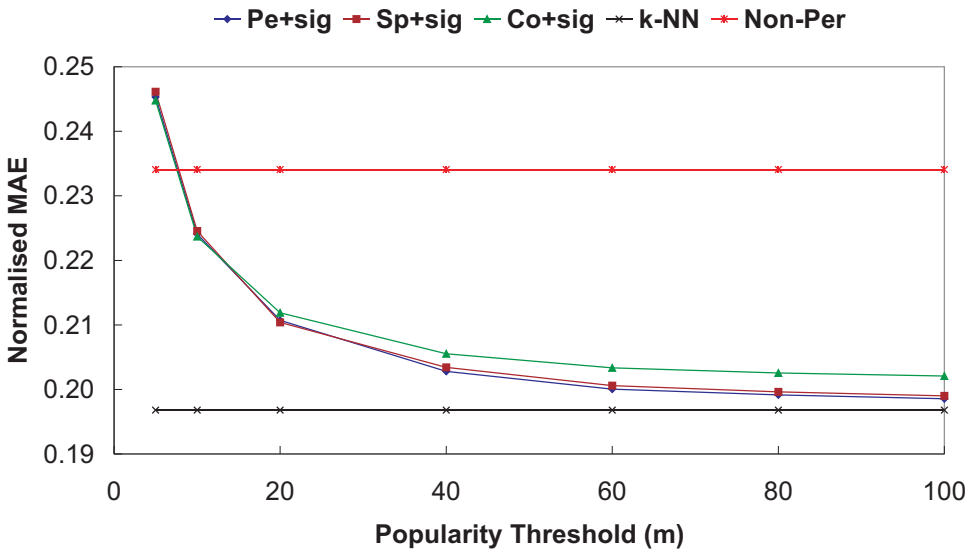


Figure 6.11: The predictive accuracy achieved by the intelligent neighbourhood selection algorithm combined with the similarity weight transformation scheme for the MovieLens dataset.

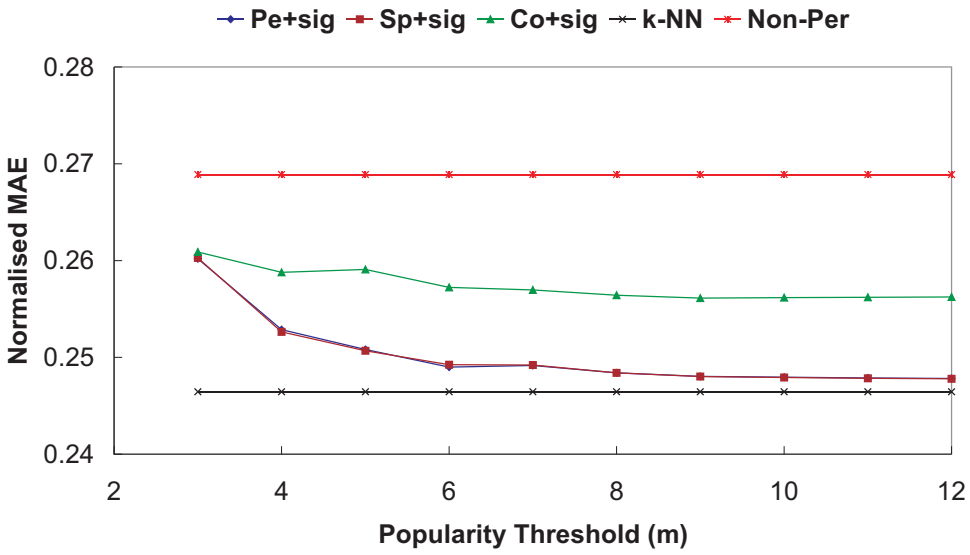


Figure 6.12: The predictive accuracy achieved by the intelligent neighbourhood selection algorithm combined with the similarity weight transformation scheme for the Smart Radio dataset.

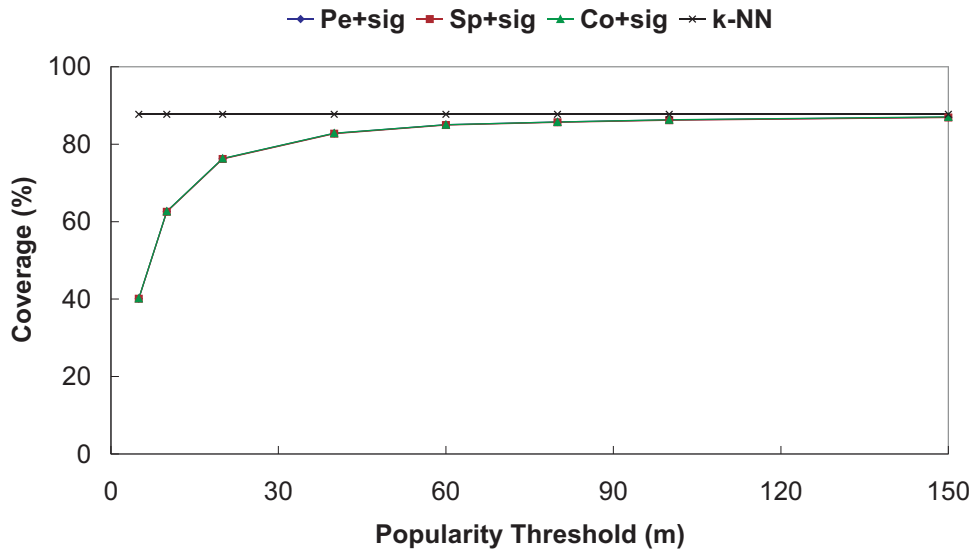


Figure 6.13: The coverage achieved by the intelligent neighbourhood selection algorithm combined with the similarity weight transformation scheme for the EachMovie dataset.

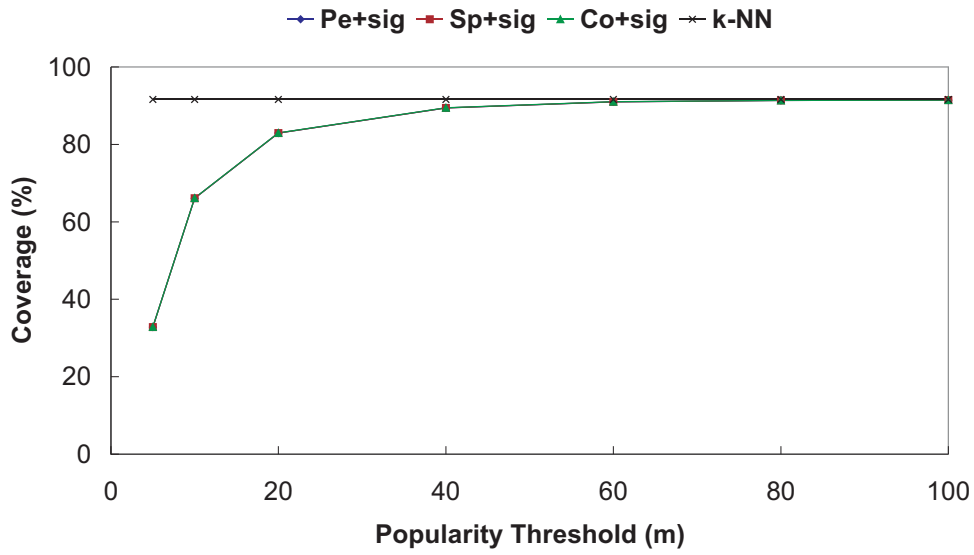


Figure 6.14: The coverage achieved by the intelligent neighbourhood selection algorithm combined with the similarity weight transformation scheme for the MovieLens dataset.

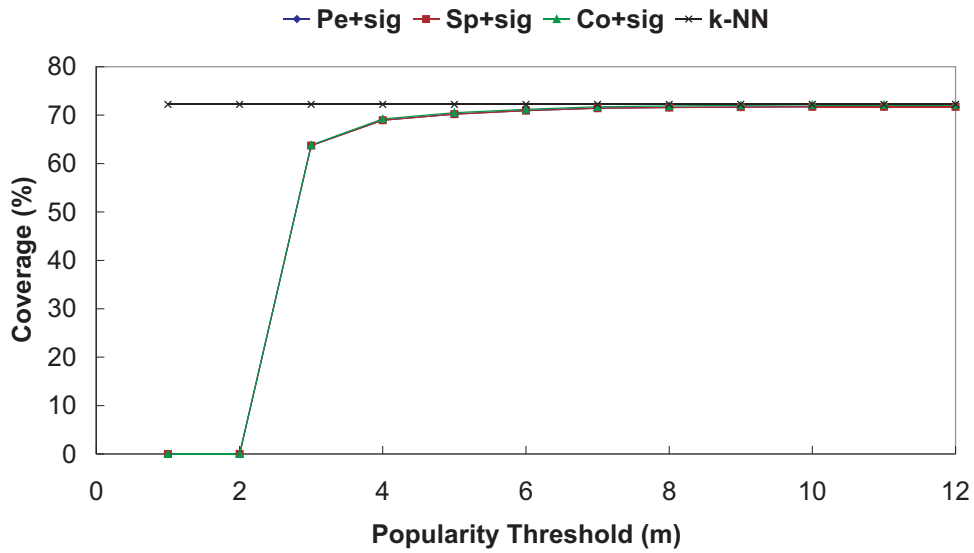


Figure 6.15: The coverage achieved by the intelligent neighbourhood selection algorithm combined with the similarity weight transformation scheme for the Smart Radio dataset.

6.3.2.3 Robustness

The strategy underlying the attack that was described in Section 5.4.3 was to choose popular items to build the attack profiles to (a) ensure a high degree of similarity between genuine and attack profiles and (b) minimise the number of different attack profiles required to target as many genuine users as possible. However, neighbour selection using the combined approach favours profiles that contain less popular items. Even if attack profiles were constructed using a number of less popular items in an attempt to defeat the algorithm, the effect is not likely to be significant since, by definition, unpopular items have proportionally less probability of being co–rated by users. As stated, all attack profiles were removed from neighbourhoods for the range of threshold values presented in the figures above. Thus, for all the datasets evaluated, the combined approach was completely robust against attack with MAPE = 0.

For the EachMovie dataset, the system was no longer robust when $m > 150$ because attack profiles began to appear in the neighbourhoods of genuine users. For the MovieLens and Smart Radio datasets, the threshold values below which the systems were robust were $m = 100$ and $m = 12$, respectively. These findings underline the importance of including the popularity threshold value in the computation of inverse item popularities, as defined in (6.3).

Choosing a threshold value. The value that is selected needs to ensure robustness against attack, while providing accuracy and coverage comparable to that of k–NN (which

was shown in Chapter 5 to provide good performance in terms of accuracy and coverage). The results shown above indicate that a trade–off exists for the combined approach between these criteria: low values for m offer increased robustness while higher values achieve the desired accuracy and coverage. For the EachMovie dataset, we see that a range of values for m exists that satisfies these criteria: i.e. $40 \leq m \leq 150$. Notice that when the similarity weight transformation scheme was not applied (Section 6.3.1.3), the corresponding range was $5 \leq m \leq 60$. Thus, the weight transformation scheme had two effects. Firstly, it increased the upper popularity threshold limit at which the system was still robust. It also, however, increased the lower limit above which good performance in terms of accuracy and coverage was achieved. This was caused by the fact that items with low popularities were infrequently co–rated by users. Since a wider range was obtained using the combined approach, the importance of the weight transformation scheme has been demonstrated.

If the above range of threshold values are expressed as a ratio of the number of users present in the dataset, a range of $(0.038, 0.141)$ is obtained. The corresponding ranges obtained for the MovieLens and Smart Radio datasets are $(0.040, 0.099)$ and $(0.060, 0.179)$, respectively. Thus, for all the datasets that were evaluated, the results indicate that significant windows exist in which to choose suitable values for m.

It is interesting to note that the ranges changed as the attack cost is varied. For example, for the EachMovie dataset, the above range referred to an attack which consisted of 64 bogus profiles being inserted into the system, each of which contained 100 items. For a fixed number of 64 attack profiles inserted into the system, the range extended to $(0.038, 0.376)$ when 25 items were contained in the attack profiles, and reduced to $(0.038, 0.075)$ when large attack profiles of size 400 were used. Similar trends were observed for the other datasets.

The reason for the change in ranges is as follows. In all cases, attack profiles were constructed using the most popular items in the respective datasets. Thus, as the size of the attack profiles was increased, the popularity of the least–popular item that was contained in them decreased. As a result, attack profiles began to appear in neighbourhoods at lower values of the popularity threshold, m.

It should be noted, however, that in the above results, attack profiles which contain 400 items are very large in the context of the EachMovie dataset, since only $1,338$ items in total[4] are contained in this system. Since many real–world recommender systems contain a vastly greater number of products (consider, for example, Amazon.com), the cost associated with creating attack profiles which consist of large percentages of available items is likely to be excessive. Thus, the probability of such large attacks occurring can be expected to be low, and the popularity threshold range within which a system meets

[4]This number refers to the number of items which received at least 1 rating.

all the relevant performance criteria is unlikely to be reduced beyond a point where the algorithm is no longer able to function.

6.3.2.4 Efficiency and Scalability

Finally, the efficiency and scalability provided by the profile utility approach are examined. In this approach, recall that the k users with the highest utility, which have rated the item for which a prediction is being sought, are simply chosen as neighbours. In contrast, when the k–NN neighbourhood formation scheme[5] is considered, for example, similarities must be computed between the active user and all other users who have rated the item in question before neighbours can be selected. Since profile utility can be computed off–line and is easily updated when new items are rated, it represents a considerable increase in efficiency when compared to k–NN. Formally, the profile utility algorithm is $O(j)$ compared to $O(r.j)$ for k–NN, where j is the average number of items co–rated by the active user and neighbour profiles and r is the number of users who have rated the item for which a prediction is sought. Critically. in contrast to k–NN, the efficiency of the profile utility approach is independent of r, and thus does not suffer from scalability problems as a system grows in size.

Various other approaches to neighbourhood formation have also been proposed in the literature. For example, Sarwar *et al.* (2002) propose database clustering as an efficient neighbourhood formation strategy. The set of users is partitioned into clusters in an off–line process. To make a prediction for a given user, the entire cluster containing the user becomes the neighbourhood. Hence, the performance of the algorithm depends on the (fixed) size of the clusters. In general, however, clustering algorithms are prone to attack unless they succeed in grouping all attack profiles into one cluster. By varying the items that are used to create attack profiles, a careful attacker can ensure that this is unlikely to occur.

The usefulness of inverse item popularity has also been recognised by other work. As described in Section 2.2.5, Rafter and Smyth (2003) propose a modified similarity measurement that calculates the correlation between users using only a fixed number of the least popular co–rated items. This algorithm does not apply a popularity threshold when calculating inverse item popularities and attack profiles, which are created using popular items as described in Section 5.4.3, are not necessarily excluded from neighbourhoods. In addition, the efficiency of this algorithm depends on the number of users in a system and is thus susceptible to scalability problems.

In conclusion, it can be stated that the neighbourhood formation scheme based on profile utility, combined with the similarity weight transformation scheme, did succeed in pro-

[5] A similar analysis applies for the similarity thresholding and k–NN *min–corr* neighbourhood formation schemes.

tecting the systems that were evaluated against attack. The results presented above relate to profile utility defined in terms of item popularity. Experiments were also conducted with utility defined in terms of item entropy and item likability. These methods did not, however, succeed in securing the systems against attack. Since the attack profiles used were constructed using popular items, the lack of success using these methods was not unexpected.

6.4 Summary

In this chapter, two different approaches were introduced with the objective of devising robust collaborative filtering algorithms. In the first approach, a neighbourhood filtering scheme that is based on related work from the field of reputation reporting systems was developed. The scheme operates by clustering neighbourhoods into two groups, and using statistical analysis to uncover any attack data that may be present. While the approach did offer improved robustness against attack, it did not provide complete protection, and, in particular, it was vulnerable to informed attacks carried out by attackers with knowledge of the underlying algorithm.

The second technique that was developed introduced the concept of profile utility, a novel approach to neighbourhood formation that takes the place of traditional similarity–based neighbourhoods. In addition, similarity weight transformations were proposed in order to protect the predictions that are made for new or infrequently rated items, which are likely to be the subject of most attacks. It was demonstrated that our approach did not adversely affect system accuracy or coverage, but dramatically improved system robustness. Another important advantage of our neighbourhood formation scheme is that profile utility can be calculated off–line and is easily updated when new items are rated by users. Thus, neighbourhoods can be formed in a very efficient manner, regardless of database size.

Conclusion

This thesis has focused on the performance evaluation of an important information filtering algorithm, namely automated collaborative filtering. In this final chapter, we will conclude by emphasising the major contributions of our work and highlighting some directions in which this research can be extended in the future. To begin, a brief summary of the earlier chapters of the thesis is presented.

Chapter 1 presents an overview of recommender systems and the benefits that these systems have brought to e–commerce applications is outlined. The information filtering algorithms that are used to implement these systems are reviewed. While the focus of the review is on the collaborative filtering technique, alternative techniques are also described.

In Chapter 2, the algorithmic framework that is used to examine the robustness of collaborative recommender systems is outlined. Resnick's deviation from mean algorithm, which is widely implemented in many settings, forms the basis of the research. The various similarity metrics, neighbourhood formations schemes and extensions that are most frequently used in conjunction with this algorithm are also described.

A review of the accuracy, coverage, efficiency and scalability performance criteria that are typically used to evaluate recommender systems is presented in Chapter 3. The principle metrics that have been developed to quantify system performance are described. Chapter 4 introduces the concept of robustness as an additional performance measure for recommender systems. Given the security implications that are typical for many recommender systems, the importance and relevance of such a measure is clear. A definition of robustness for recommender systems is proposed, which is as follows:

The ability of a system to provide stable predictions and recommendations given some degree of noise present in the data.

In addition, the following metrics are introduced in order to evaluate system robustness: Percentage of Attack Profiles in Neighbourhoods, Mean Absolute Prediction Error and the Percentage of Good and Bad Predictions. These metrics complement one another and serve to provide a complete measure of system robustness.

Attacks

In Chapter 5, attack strategies are developed for product push and nuke attacks. Recall that the objectives of these attacks is to promote or demote the predicted ratings that are

made for targeted items. All the attacks that are considered in this thesis are implemented by creating and inserting bogus user profiles through the normal system interface. No other access to a system's database is assumed for attackers.

Similarity Metrics

Strategies are developed for a product push attack using the k–NN neighbourhood formation scheme in Section 5.4. It is discussed in Section 2.2.2 that the type of rating scale that is used by a system affects the weights that are calculated using Cosine similarity. Specifically, if positive rating scales are used, then any negative similarities that may exist between users cannot be captured. This is not the case for the correlation metrics, which can result in positive or negative weights irrespective of the rating scales that are used. The evaluation datasets that are used in this thesis all use positive rating scales, which are commonly used in recommender systems. Thus, if a system uses Cosine similarity to compute the weights between users **and** if positive rating scales are used, attacks are easier to implement and therefore more likely to succeed because there is no risk of the contributions of multiple attack profiles being canceled out.

Random Push Attack

The first attack that is considered is a random push attack, in which attack profiles are created randomly (Section 5.4.1). This attack is only successful when Cosine similarity is used, because all of the attack profiles that are present in neighbourhoods contribute to pushing the targeted item. Thus, Cosine similarity is not a suitable metric for systems that operate using positive rating scales given that it is susceptible to this simple form of attack. The correlation metrics proved robust against this attack because the contributions of multiple attack profiles are largely canceled out. The trends observed for both correlation metrics in this attack, and in all subsequent attacks, are closely matched. As stated in Section 2.2.2, this finding is to be expected, given that ratings are used to rank items, where such ratings are, in essence, ranks.

Focused Push Attack

Our initial focused attack strategy (Section 5.4.2) consists of creating 3–item attack profiles with the objective of exploiting an important weakness that exists in regard to the correlation similarity metrics. The weakness in question is that correlations between users are computed using only co–rated items. Thus, high similarities can be achieved when users have only a small number of items in common. This attack is successful against the genuine users who have rated the items that are contained in the attack profiles. For

these users, the percentage of good predictions increases substantially to 64% when 64 attack profiles are inserted, from a pre–attack baseline of 15%. The predictive accuracy provided by the algorithm also deteriorates significantly as a result of the attack. When Cosine similarity is used, the attack, while less successful than for the correlation metrics, nevertheless achieves a 100% increase in the percentage of good predictions that are delivered by the system. The attack is less successful for Cosine given the small size of the attack profiles that are used. When the significance weighting algorithm extension is used, the effect of the attack is reduced but remains significant in the case of the correlation metrics.

In order to target a greater number of users in a system, the next strategy involves the creation of larger attack profiles (Section 5.4.3). For the correlation metrics, attack profiles are constructed using groups of popular items which are liked and disliked by the genuine users of a system. Ratings are chosen to ensure positive correlations between genuine and attack profiles and to maximise the difference term $(r_{i,j} - \bar{r}_i)$ in (2.7). For Cosine similarity, the strategy is more straightforward because all the systems that are evaluated use positive rating scales. In this case, the l most popular items are simply selected to build the attack profiles. This attack is very successful across all the datasets that are evaluated, irrespective of the similarity metric that is used. For example, in the case of the EachMovie dataset, the percentage of good predictions that are achieved using Cosine Similarity increases from 23% to 92% when 64 attack profiles are inserted. Similar increases are achieved when the correlation metrics are used. Thus, the k–NN neighbourhood formation scheme proves very vulnerable to this form of attack. In Section 5.6, a similar strategy, but with ratings chosen to minimise the difference term in (2.7), is successfully used for product nuke attacks.

For the above attacks, item popularity and item likability knowledge is assumed to be known by an attacker. In Section 5.4.5, it is shown that, when such knowledge is only partially available, attack success is not substantially reduced.

Algorithm Extensions

Several extensions that are most widely used in conjunction with Resnick's algorithm are evaluated from a robustness perspective. The extensions that are evaluated are significance weighting, case amplification, inverse user frequency and item entropy. In general, with the exception of significance weighting, none of the other extensions had a major impact on system robustness. While significance weighting does not provide complete robustness against attack, it is nevertheless a useful extension since it increases the cost of successful attacks.

Alternative Attack Strategies

Alternative attack strategies, which are proposed by Lam and Riedl (2004), are examined in Section 5.7. The strategies are referred to as 'AverageBot' and 'RandomBot' – the former is the more sophisticated of the two and performs the best. In each case, all available items are used in the construction of attack profiles. The AverageBot attack, when compared to the push attack of Section 5.4.3, has a much greater associated cost and proves less successful in terms of the magnitude of the predictions that are achieved – mainly because the ratings strategy that is applied does not maximise the difference term in (2.7).

Neighbourhood Formation Schemes

Section 5.8 examines the robustness provided by the neighbourhood formation schemes that are described in Section 2.2.3. The performance of the previously examined k–NN scheme is compared to that provided by a similarity thresholding scheme and a combined k–NN and thresholding scheme, when each is subjected to a product push attack. From the results, it is clear that none of the neighbourhood formation/similarity weight combinations are robust against attack when configured to provide good performance across all dimensions.

As is stated in Section 5.4.3, the use of correlation similarity metrics requires a more sophisticated attack strategy to ensure that all attack profiles correlate in the same direction with genuine users. This is not an issue, however, in the case of the combined k–NN and thresholding scheme, where all negative correlations are ignored by the algorithm. Thus if an attacker is unsure about which ratings to assign to attack profile items, he could simply create two sets of attack profiles, with the rating schemes reversed in each. While such a strategy necessarily results in a higher attack cost, it nevertheless represents an additional security implication for this particular neighbourhood formation scheme.

Cost–Benefit Analysis

In Section 5.9, a cost–benefit analysis is performed for the product push attack outlined in Section 5.4.3. For this analysis, the widely used k–NN neighbourhood formation scheme and the Pearson correlation similarity metric are considered. The objective is to determine whether or not financial gains can be realised by attackers. The analysis assumes (a) that rating entry incurs a financial cost and (b) that all items sell at the same price. These assumptions do not favour attackers and, indeed, are typically not true for real–world recommender systems. The analysis shows that the most effective attack, from an ROI perspective, is to insert just 1, small–sized attack profile into a system. Furthermore, it is

demonstrated that substantial profits can be realised by any malicious persons who may be inclined to implement such attacks. These results represent a real cause of concern for system managers and designers, since they indicate that cheap attacks, involving little effort on the part of an attacker, are the most effective. In addition, much more substantial gains can be achieved by attacking even small–sized systems when users are permitted to insert ratings free–of–charge. Thus, we believe that, in the real–world, recommender systems ought to place at least some restrictions on rating entry in order to provide a greater degree of robustness against attack.

Preventing Attack

It is clear from the results of Chapter 5 that the class of collaborative filtering algorithms that are considered in this thesis prove very vulnerable to attack. In Chapter 6, two different approaches are introduced with the objective of improving robustness against attack. Both operate by attempting to identify and exclude from neighbourhoods any attack profiles that may be present in a system. The first technique is based on related work from the field of reputation reporting systems. These systems aim to provide a reputation estimate for buyers and sellers engaged in on–line marketplaces, and need to be sufficiently robust to deal with any malicious agents who may attempt to interfere with the system. While the technique that is proposed for collaborative filtering offers improved robustness against attack, it did not provide complete protection, and, in particular, it is vulnerable to more sophisticated attacks that are carried out by attackers with knowledge of the underlying algorithm.

The second technique introduces novel approaches to neighbourhood formation and similarity weight transformation. The profile utility technique evaluates the usefulness of possible neighbours based on certain characteristics of the items that have been rated by each. It is demonstrated that this approach provides robustness against attack, while also achieving good performance in terms of predictive accuracy and coverage. With this approach, the cost of mounting successful attacks can be made prohibitive and thus are malicious users dissuaded against such practice. Another important advantage of our neighbourhood formation scheme is that profile utility can be calculated off–line and is easily updated when new items are rated by users. Thus, neighbourhoods can be formed in a very efficient manner, regardless of database size.

Future Work

Security is a prime consideration for all software applications. We argue that malicious attacks against recommender systems must be expected, and that solutions need to be developed to counter such activities. In this work, we have considered the effects of

product push and product nuke attacks on one class of collaborative filtering algorithms. Many other approaches to collaborative filtering have been proposed in the literature – these algorithms also need to be subjected to a robustness analysis.

Hybrid recommender systems, which incorporate multiple filtering techniques, may provide a greater measure of robustness against attack. These systems most often employ a collaborative filter which is used in conjunction with other types of filters, such as content– and demographic–based filters. It is necessary to establish the performance of these and other filtering techniques when subjected to attack, both individually and when combined in hybrid recommenders. It should not be automatically assumed that hybrid recommender systems are robust against attack; as we have shown in this work, collaborative filters are vulnerable, which implies that any system using this particular type of filter as a component part may also be compromised.

In this thesis, the analysis deals specifically with the robustness of predictions that are made by systems which are subjected to attack. Recommender systems are also used to generate recommendations, where ranked lists of relevant items are delivered to users. It is likely that different recommendation techniques have varying levels of robustness against attack. For example, the most liked item recommendation strategy (Section 2.2.4) is unlikely to be robust given that items are ranked according to predicted ratings, which we have shown to be very susceptible to attack. While some analysis has been carried out in this regard by Lam and Riedl (2004), it is necessary to further examine the robustness of the recommendation process to noise in the data.

Dataset characteristics, such as the distributions of users and items that are contained in a system, the sparseness of the data, the domain of a system, etc. are likely to have an effect on system robustness. A related study has been carried out by Rafter and Smyth (2001), where certain domain features that influence the success of collaborative filters are identified and examined. A similar study is required from a robustness perspective.

Several attack strategies have been outlined in this thesis, but other approaches are also possible. For example, an attacker could probe a recommender system, through the normal system interface, and use the system's output to select items and ratings with which to construct attack profiles. The robustness of recommender systems against this and other attack strategies also needs to be investigated.

It was found both in this work and by Breese *et al.* (1998) that the predictive accuracy provided by the Cosine similarity metric does not match that provided by the correlation metrics. However, these comparisons are all made using datasets that operate over positive rating scales. As stated in Section 2.2.2, the angle θ between any pair of user vectors is constrained for such systems to lie in the range $0 \leq \theta \leq \frac{\pi}{2}$. It would be interesting to examine the accuracy performance of Cosine similarity if rating scales were translated and centered around 0, thereby permitting any negative similarities that exist between users

to be utilised. Indeed, such an approach is likely to lead to improved robustness, since attackers would then have to ensure that all attack profiles have either positive or negative similarity with genuine users, as is the case when the correlation similarity metrics are used.

Recommender systems have now been widely and successfully implemented in many e–commerce applications. These systems represent a significant business investment – both financially, given the implementation and maintenance costs, and from a reputation perspective, given that customers are unlikely to remain loyal if it is perceived that recommendations of suspect quality are being made. Thus, the security of these systems is of prime importance. In this thesis, we have proposed a framework within which to analyse the robustness of recommender systems, and we have established new performance measures for robustness. We have demonstrated that malicious attacks are indeed capable of substantially manipulating the predictions that are made by one class of collaborative filtering algorithms. These results clearly represent a significant cause of concern for recommender system managers and designers. We have proposed solutions that are designed to protect systems against attack – in particular, our utility–based neighbourhood formation and similarity weight transformation schemes prove to be very effective. Other researchers are now following in this direction – for example Lam and Riedl (2004), O'Donovan and Smyth (2005) and Massa and Avesani (2004), who all consider the robustness of recommender systems against attack.

In summary, the key message of this thesis is that companies which employ recommender systems must take robustness against malicious attack into account – otherwise they run the risk of having their recommender systems turned into marketing and promotion tools for the products of unscrupulous individuals.

Bibliography

Anton H. (1994). *Elementary Linear Algebra, 7th Edition*. John Wiley & Sons, Inc. ISBN 0471587427.

Avesani P., Massa P., and Tiella R. (2004). "Moleskiing: A trust–aware decentralised recommender system." *In Proceedings of the 1st Workshop on Friend of a Friend, Social Networking and the Semantic Web*, Galway, Ireland, September 1–2.

Balabanovic M. (1997). "An adaptive web page recommendation service." *In Proceedings of the 1st International Conference on Autonomous Agents (Agents'97)*, pp. 378–385, ACM Press, Marino Del Ray, California, USA, February 5–8.

Balabanovic M. and Shoham Y. (1997). "Fab: Content–based, collaborative recommendation." *Communications of the ACM*, vol. 40, no. 3, pp. 66–72, ACM Press, March.

Basu C., Hirsh H., and W.Cohen (1998). "Recommendation as classification: Using social and content–based information in recommendation." *In Proceedings of the 15th National Conference on Artificial Intelligence (AAAI'98)*, pp. 714–720, AAAI Press, Madison, Wisconsin, USA, July 26–30.

Berry M.W., Dumais S.T., and O'Brien G. (1995). "Using linear algebra for intelligent information retrieval." *SIAM Review*, vol. 37, no. 4, pp. 573–595.

Breese J.S., Heckerman D., and Kadie C. (1998). "Empirical analysis of predictive algorithms for collaborative filtering." *In Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence*, pp. 43–52, Morgan Kaufmann, Madison, Wisconsin, USA, July 24–26.

Bridge D. and Kelleher J. (2002). "Experiments in sparsity reduction: Using clustering in collaborative recommenders." *In Proceedings of the 13th Artificial Intelligence and Cognitive Science Conference (AICS'02)*, vol. 2464, pp. 144–149, Springer, Limerick, Ireland, September 12–13.

Burke R. (1999). "Integrating knowledge–based and collaborative filtering recommender systems." *In Proceedings of the Workshop on Artificial Intelligence and Electronic Commerce (AIEC'99)*, pp. 69–72, AAAI Press/MIT Press, Orlando, Florida, July 18.

Burke R. (2000). "Knowledge–based recommender systems." *Encyclopedia of Library and Information Science*, vol. 69, no. 32, Marcel Dekker, Inc.

Burke R. (2002). "Hybrid recommender systems: Survey and experiments." *User Modeling and User–Adapted Interaction*, vol. 12, no. 4, pp. 331–370, Kluwer Academic Publishers, November.

Canny J. (2002). "Collaborative filtering with privacy via factor analysis." *In Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 238–245, ACM Press, Tampere, Finland, August 11–15.

Claypool M., Gokhale A., Miranda T., Murnikov P., Netes D., and Sartin M. (1999). "Combining content and collaborative filters in an on–line newspaper." *In Proceedings of the ACM SIGIR Workshop on Recommender Systems: Algorithms and Evaluation, 22nd International Conference on Research and Development in Information Retrieval*, pp. 15–22, ACM Press, Berkeley, California, USA, August 15–19.

Clerkin P., Cunningham P., and Hayes C. (2003). "Concept discovery in collaborative recommender systems." *In Proceedings of the 14th Irish Artificial Intelligence and Cognitive Science Conference (AICS'03)*, pp. 34–39, Dublin, Ireland, September 17–19.

Condliff M., Lewis D., Madigan D., and Posse C. (1999). "Bayesian mixed–effects models for recommender systems." *In Proceedings of the ACM SIGIR Workshop on Recommender Systems: Algorithms and Evaluation, 22nd International Conference on Research and Development in Information Retrieval*, pp. 23–30, ACM Press, Berkeley, California, USA, August 15–19.

Dellarocas C. (2000). "Immunizing on–line reputation reporting systems against unfair ratings and discriminatory behavior." *In Proceedings of the 2nd ACM Conference on Electronic Commerce (EC'00)*, pp. 150–157, ACM Press, Minneapolis, Minnesota, USA, October 17–20.

Deshpande M. and Karypis G. (2004). "Item–based top–N recommendation algorithms." *ACM Transactions on Information Systems (TOIS)*, vol. 22, no. 1, pp. 143–177, ACM Press, January.

Foltz P.W. (1990). "Using latent semantic indexing for information filtering." *In Proceedings of the Conference on Office Information Systems*, pp. 40–47, ACM Press, Cambridge, Massachusetts, USA, April 25–27.

Geyer-Schulz A. and Hahsler M. (2002). "Evaluation of recommender algorithms for an internet information broker based on simple association rules and on the repeat–buying theory." *In Proceedings of the Fourth WebKDD Workshop: Web Mining for Usage Patterns & User Profiles*, pp. 100–114, Edmonton, Canada, July 23.

Goldberg D., Nichols D., Oki B.M., and Terry D. (1992). "Using collaborative filtering to weave an information tapestry." *Communications of the ACM*, vol. 35, no. 12, pp. 61–70, ACM Press, December.

Goldberg K., Roeder T., Gupta D., and Perkins C. (2001). "Eigentaste: A constant time collaborative filtering algorithm." *Information Retrieval*, vol. 4, no. 2, pp. 133–151, Kluwer Academic Publisher, July.

Groot P., van Harmelen F., and ten Teije A. (2000). "Torture tests – a quantitative analysis for the robustness of knowledge based systems." *In Proceedings of the 12th International Conference on Knowledge Acquisition, Modeling and Management (EKAW'00)*, pp. 403–418, Springer, Juan-les-Pins, France, October 2–6.

Gupta D., Digiovanni M., Narita H., and Goldberg K. (1999). "Jester 2.0: Evaluation of a new linear–time collaborative filtering algorithm." *In Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 291–292, ACM, Berkeley, California, USA, August 15–19.

Hayes C., Cunningham P., Clerkin P., and Grimaldi M. (2002). "Programme driven music radio." *In Proceedings of the 15th European Conference on Artificial Intelligence (ECAI'02)*, pp. 633–637, IOS Press, Lyon, France, July 21–26.

Herlocker J., Konstan J., Borchers A., and Riedl J. (1999). "An algorithmic framework for performing collaborative filtering." *In Proceedings of the 22nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 230–237, ACM Press, Berkeley, California, USA, August 15–19.

Hill W., Stead L., Rosenstein M., and Furnas G. (1995). "Recommending and evaluating choices in a virtual community of use." *In Proceedings of SIGCHI Conference on Human Factors in Computing Systems*, pp. 194–201, ACM Press, Denver, Colorado, USA, May 7–11.

Hsu C.N. and Knoblock C.A. (1995). "Estimating the robustness of discovered knowledge." *In Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, pp. 156–161, AAAI Press, Montreal, Quebec, Canada, August 20–21.

Hsu C.N. and Knoblock C.A. (1996). "Discovering robust knowledge from dynamic closed–world data." *In Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eight Innovative Applications of Artificial Intelligence Conference, (AAAI-96, IAAI'96)*, vol. 1, pp. 820–827, AAAI Press/The MIT Press, Portland, Oregon, USA, August 4–8.

IEEE (1990). *IEEE standard glossary of software engineering terminology*. IEEE Standard 610.12–1990. ISBN 155937067X.

Karypis G. (2001). "Evaluation of item–based top–N recommendation algorithms." *In Proceedings of the Tenth ACM International Conference on Information and Knowledge Management (CIKM'01)*, pp. 247–254, ACM Press, Atlanta, Georgia, USA, November 5–10.

Krulwich B. (1997). "Lifestyle finder: Intelligent user profiling using large–scale demographic data." *Artificial Intelligence Magazine*, vol. 18, no. 2, pp. 37–45, AAAI Publications.

Lam S.K. and Riedl J. (2004). "Shilling recommender systems for fun and profit." *In Proceedings of the 13th International World Wide Web Conference*, pp. 393–402, ACM Press, New York, New York, USA, May 17–20.

Macnaughton-Smith P., Williams W.T., Dale M., and Mockett L. (1964). "Dissimilarity analysis – a new technique of hierarchical sub–division." *Nature*, vol. 202, pp. 1034–1035.

Massa P. and Avesani P. (2004). "Trust–aware collaborative filtering for recommender systems." *In Proceedings of the International Conference on Cooperative Information Systems (CoopIS'04)*, pp. 492–508, Springer, Agia Napa, Cyprus, October 25–29.

Massa P. and Bhattacharjee B. (2004). "Using trust in recommender systems: An experimental analysis." *In Proceedings of the 2nd International Conference on Trust Management (iTrust'04)*, pp. 221–235, Springer, Oxford, England, March 29 – April 1.

Melville P., Mooney R.J., and Nagarajan R. (2002). "Content–boosted collaborative filtering for improved recommendations." *In Proceedings of the 18th National Conference on Artificial Intelligence (AAAI'02)*, pp. 187–192, AAAI Press, Edmonton, Alberta, Canada, July 28–August 1.

Miller B., Riedl J., and Konstan J.A. (1997). "Experiences with GroupLens: Making Usenet useful again." *In Proceedings of the USENIX 1997 Annual Technical Conference*, pp. 219–231, Anaheim, California, USA, January 6–10.

Miller B.N. (2003). "Towards a personal recommender system." *PhD Thesis*, University of Minnesota, USA.

Mirza B.J., Keller B.J., and Ramakrishnan N. (2003). "Studying recommendation algorithms by graph analysis." *Journal of Intelligent Information Systems*, vol. 20, no. 2, pp. 131–160.

Montaner M., Lopez B., and de la Rosa J.L. (2002). "Developing trust in recommender agents." *In Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 304–305, ACM Press, Bologna, Italy, July 15–19.

Mooney R. and Roy L. (2000). "Content–based book recommending using learning for text categorization." *In Proceedings of the 5th ACM Conference on Digital Libraries*, pp. 195–204, ACM Press, San Antonio, Texas, USA, June 2-7.

Oard D.W., Leuski A., and Stubblebine S. (2003). "Protecting the privacy of observable behaviour in distributed recommender systems." *In Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Workshop on Implicit Measures of User Interests and Preferences*, ACM Press, Toronto, Canada, August 1.

O'Connor M. and Herlocker J. (1999). "Clustering items for collaborative filtering." *In Proceedings of the ACM SIGIR Workshop on Recommender Systems: Algorithms and Evaluation, 22nd International Conference on Research and Development in Information Retrieval*, pp. 11–14, ACM Press, Berkeley, California, USA, August 15–19.

O'Donovan J. and Smyth B. (2005). "Trust in recommender systems." *In Proceedings of the 10th International Conference on Intelligent User Interfaces (IUI'05)*, pp. 167–174, ACM Press, San Diego, California, USA, January 10–13.

O'Mahony M.P., Hurley N.J., Kushmerick N., and Silvestre G.C.M. (2004a). "Collaborative recommendation: A robustness analysis." *ACM Transactions on Internet Technology (TOIT), Special Issue on Machine Learning for the Internet*, vol. 4, no. 4, pp. 344–377, ACM Press, November.

O'Mahony M.P., Hurley N.J., and Silvestre C.C.M. (2004b). "An evaluation of neighbourhood formation on the performance of collaborative filtering." *Artificial Intelligence Review*, vol. 21, no. 1, pp. 215–228, Kluwer Academic Publishers, March.

O'Mahony M.P., Hurley N.J., and Silvestre G.C.M. (2002a). "Promoting recommendations: An attack on collaborative filtering." *In Proceedings of the 13th International Conference on Database and Expert Systems Applications (DEXA'02)*, vol. 2453, pp. 494–503, Springer–Verlag, Aix-en-Provence, France, September 2–6.

O'Mahony M.P., Hurley N.J., and Silvestre G.C.M. (2002b). "Towards robust collaborative filtering." *In Proceedings of the 13th Irish International Conference on Artificial Intelligence and Cognitive Science (AICS'02)*, vol. 2464, pp. 87–94, Springer, Limerick, Ireland, September 12–13.

O'Mahony M.P., Hurley N.J., and Silvestre G.C.M. (2003a). "Collaborative filtering – safe and sound?" *In Proceedings of the 14th International Symposium on Methodologies for Intelligent Systems (ISMIS'03), Foundations of Intelligent Systems*, vol. 2871, pp. 506–510, Maebashi City, Japan, October 28–31.

O'Mahony M.P., Hurley N.J., and Silvestre G.C.M. (2003b). "An evaluation of the performance of collaborative filtering." *In Proceedings of the 14th Irish International Conference on Artificial Intelligence and Cognitive Science (AICS'03)*, pp. 164–168, Dublin, Ireland, September 17–19.

O'Mahony M.P., Hurley N.J., and Silvestre G.C.M. (2004c). "Efficient and secure collaborative filtering through intelligent neighbour selection." *In Proceedings of the 16th European Conference on Artificial Intelligence (ECAI'04)*, pp. 383–387, IOS Press, Valencia, Spain, August 23–27.

O'Mahony M.P., Hurley N.J., and Silvestre G.C.M. (2004d). "Utility-based neighbourhood formation for efficient and robust collaborative filtering." *In Proceedings of the 5th ACM Conference on Electronic Commerce (EC'04)*, pp. 260–261, ACM Press, New York, New York, USA, May 17–20.

O'Sullivan D., Wilson D., and Smyth B. (2002). "Using collaborative filtering data in case–based recommendation." *In Proceedings of the 15th International Florida Artificial Intelligence Research Society (FLAIRS'02)*, pp. 121–128, AAAI Press, Pensacola Beach, Florida, USA, May 14–16.

Pazzani M.J. (1999). "A framework for collaborative, content–based and demographic filtering." *Artificial Intelligence Review*, vol. 13, no. 5–6, pp. 393–408, Kluwer Academic Publishers, December.

Pearce D.W., Editor (1992). *The MIT Dictionary of Modern Economics, 4th Edition*. The MIT Press. ISBN 0262660784.

Popescul A., Ungar L.H., Pennock D.M., and Lawrence S. (2001). "Probabilistic models for unified collaborative and content–based recommendation for sparse–data environments." *In Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence (UAI'01)*, pp. 437–444, Morgan Kaufmann, Seattle, Washington, USA, August 2–5.

Rafter R., Bradley K., and Smyth B. (2000). "Automated collaborative filtering applications for on–line recruitment services." *In Proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web–based Systems (AH'00)*, vol. 1892, pp. 363–368, Springer–Verlag Heidelberg, Trento, Italy, August 28–30.

Rafter R. and Smyth B. (2001). "Towards a domain analysis methodology for collaborative filtering." *In Proceedings of the 23rd BCS–IRSG European Colloquium on Information Retrieval Research, (ECIR'01)*, pp. 172–185, Darmstadt, Germany, April 4–6.

Rafter R. and Smyth B. (2003). "Item selection strategies for collaborative filtering." *In Proceedings of the 18th International Joint Conference on Artificial Intelligence, (IJCAI'03)*, pp. 1437–1439, Morgan Kaufmann, Acapulco, Mexico, August 9–15.

Ramakrishnan N., Keller B.J., Mirza B.J., Grama A.Y., and Karypis G. (2001). "Privacy risks in recommender systems." *IEEE Internet Computing*, vol. 5, no. 6, pp. 54–62, IEEE Educational Activities Department, November.

Resnick P., Iacovou N., Suchak M., Bergstrom P., and J.Riedl (1994). "Grouplens: An open architecture for collaborative filtering of netnews." *In Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW'94)*, pp. 175–186, ACM Press, Chapel Hill, North Carolina, USA, October 22–26.

Resnick P. and Varian H.R. (1997). "Recommender systems – introduction to the special section." *Communications of the ACM*, vol. 40, no. 3, pp. 56–58, ACM Press.

Resnick P. and Zeckhauser R. (2002). "Trust among strangers in internet transactions: Empirical analysis of ebay's reputation system." *The Economics of the Internet and E-Commerce, Advances in Applied Microeconomics*, vol. 11, pp. 127–157, Elsevier Science.

Resnick P., Zeckhauser R., Friedman E., and Kuwabara K. (2000). "Reputation systems." *Communications of the ACM*, vol. 43, no. 12, pp. 45–48, ACM Press.

Salton G. and McGill M. (1983). *Introduction to Modern Information Retrieval*. McGraw–Hill, New York. ISBN 0070665265.

Sarwar B., Karypis G., Konstan J., and Riedl J. (2000a). "Application of dimensionality reduction in recommender system – a case study." *In Proceedings of The 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Workshop on Web Mining for E–Commerce – Challenges and Opportunities (WEBKDD'00)*, Boston, Massachusetts, USA, August 20.

Sarwar B., Karypis G., Konstan J., and Riedl J. (2001). "Item–based collaborative filtering recommendation algorithms." *In Proceedings of the Tenth International World Wide Web Conference*, pp. 285–295, ACM Press, Hong Kong, May 1–5.

Sarwar B., Karypis G., Konstan J., and Riedl J. (2002). "Recommender systems for large–scale e–commerce: Scalable neighborhood formation using clustering." *In Proceedings of the Fifth International Conference on Computer and Information Technology (ICCIT'02)*, Dhaka, Bangladesh, December 27–28.

Sarwar B.M., Karypis G., Konstan J.A., and Riedl J. (2000b). "Analysis of recommendation algorithms for e–commerce." *In Proceedings of the 2nd ACM Conference on Electronic Commerce (EC'00)*, pp. 158–167, ACM, Minneapolis, Minnesota, USA, October 17–20.

Sarwar B.M., Konstan J.A., Borchers A., Herlocker J., Miller B., and J.Riedl (1998). "Using filtering agents to improve prediction quality in the grouplens research collabo-

rative filtering system." *In Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW'98)*, pp. 345–354, ACM Press, Seattle, Washington, USA.

Schafer J.B., Konstan J., and Riedl J. (1999). "Recommender systems in e–commerce." *In Proceedings of the 1st ACM Conference on Electronic Commerce*, pp. 158–166, ACM, Denver, Colorado, USA, November 3–5.

Shardanand U. (1994). "Social information filtering for music recommendation." *EECS M. Eng. Thesis, (also TR–94–04)*, Learning and Common Sense Group, MIT Media Laboratory, MIT, Massachusetts, USA.

Shardanand U. and Maes P. (1995). "Social information filtering: Algorithms for automating word of mouth." *In Proceedings of ACM Conference on Human Factors in Computing Systems (CHI'95)*, pp. 210–217, ACM, Denver,Colorado, USA, May 7–11.

Smyth B. and Cotter P. (2000). "A personalized TV listings service for the digital TV age." *Journal of Knowledge–Based Systems*, vol. 13, no. 2–3, pp. 53–59.

Steinbach M., Karypis G., and Kumar V. (2000). "A comparison of document clustering techniques." *In Proceedings of Workshop on Text Mining, 6th ACM SIGKDD International Conference on Data Mining (KDD'00)*, pp. 109–110, ACM, Boston, Massachusetts, USA, August 20–23.

Terveen L., Hill W., Amento B., McDonald D., and Creter J. (1997). "PHOAKS: A system for sharing recommendations." *Communications of the ACM*, vol. 40, no. 3, pp. 59–62, ACM Press, March.

Tran T. and Cohen R. (2000). "Hybrid recommender systems for electronic commerce." *In Proceedings of the 17th National Conference on Artificial Intelligence (AAAI'00), Workshop on Knowledge-Based Electronic Markets*, pp. 78–84, AAAI Press, Austin, Texas, USA, July 30–31.

Ungar L.H. and Foster D.P. (1998). "Clustering methods for collaborative filtering." *In Workshop on Recommender Systems at the 15th National Conference on Artificial Intelligence (AAAI'98)*, pp. 112–125, AAAI Press, Madison, Wisconsin, USA, July 27.

Watson I. (1997). *Applying Case–Based Reasoning: Techniques for Enterprise Systems*. Morgan Kaufmann Publishers, Inc. ISBN 1558604626.

Yu K., Wen Z., Xu X., and Ester M. (2001). "Feature weighting and instance selection for collaborative filtering." *In Proceedings of the 2nd International Workshop on Management of Information on the Web – Web Data and Text Mining (MIW'01)*, pp. 285–290, September 3–7.

Zacharia G., Moukas A., and Maes P. (1999). "Collaborative reputation mechanisms in electronic marketplaces." *In Proceedings of the 32nd International Conference on System Sciences (HICSS'99)*, vol. 8, p. 8026, IEEE Computer Society, Hawaii, USA, January 5–8.

Appendix A

Derivation of Inverse User Frequency

The objective of inverse user frequency is to minimise the influence of very popular items in the calculation of similarity weights. The motivation behind this approach is that less popular items are better able to discriminate between users than more popular items. Inverse user frequency is implemented as follows. The term f_j is defined as $\log\frac{n_d}{n_j}$, where n_j is the popularity of item j in the database and n_d is the total number of users in the database. In this appendix, a derivation of inverse user frequency, when applied to the Pearson correlation similarity weight, is presented. A similar analysis applies to the Spearman rank correlation similarity weight. Note that the equation for inverse user frequency that appears below corrects that which is presented by Breese *et al.* (1998).

From Section 2.2.2, the Pearson correlation weight between users a and i is given by:

$$w(a,i) = \frac{\sum_j \Big(r_{a,j} - \bar{r}_a\Big)\Big(r_{i,j} - \bar{r}_i\Big)}{\sqrt{\sum_j \Big(r_{a,j} - \bar{r}_a\Big)^2 \sum_j \Big(r_{i,j} - \bar{r}_i\Big)^2}}$$

where $j \in I_a \cap I_i$.

Applying the f_j term as a frequency, the above weight becomes:

$$w(a,i) = \frac{\sum_j f_j\Big(r_{a,j} - \bar{r}_a\Big)\Big(r_{i,j} - \bar{r}_i\Big)}{\sqrt{\sum_j f_j\Big(r_{a,j} - \bar{r}_a\Big)^2 \sum_j f_j\Big(r_{i,j} - \bar{r}_i\Big)^2}} \tag{A.1}$$

First, expand the numerator of (A.1). Substituting $\bar{r}_a = \frac{\sum_k f_k r_{a,k}}{\sum_k f_k}$ and simplifying, we obtain:

$$\begin{aligned}\sum_j f_j\big(r_{a,j} - \bar{r}_a\big)\big(r_{i,j} - \bar{r}_i\big) &= \frac{\sum_j f_j\Big(\sum_k f_k\ \ r_{a,j} - \sum_k f_k r_{a,k}\Big)\Big(\sum_k f_k\ \ r_{i,j} - \sum_k f_k r_{i,k}\Big)}{\Big(\sum_k f_k\Big)^2} \\ &= \sum_j f_j r_{a,j} r_{i,j} - \frac{\Big(\sum_k f_k\Big)\Big(\sum_k f_k r_{i,k}\Big)\Big(\sum_j f_j r_{a,j}\Big)}{\Big(\sum_k f_k\Big)^2} \\ &\quad - \frac{\Big(\sum_k f_k\Big)\Big(\sum_k f_k r_{a,k}\Big)\Big(\sum_j f_j r_{i,j}\Big)}{\Big(\sum_k f_k\Big)^2} \\ &\quad + \frac{\Big(\sum_j f_j\Big)\Big(\sum_k f_k r_{a,k}\Big)\Big(\sum_k f_k r_{i,k}\Big)}{\Big(\sum_k f_k\Big)^2}\end{aligned}$$

Since $k \in I_a \cap I_i$, the above reduces to:

$$\sum_j f_j\big(r_{a,j} - \bar{r}_a\big)\big(r_{i,j} - \bar{r}_i\big) = \frac{\Big(\sum_j f_j\Big)\Big(\sum_j f_j r_{a,j} r_{i,j}\Big) - \Big(\sum_j f_j r_{a,j}\Big)\Big(\sum_j f_j r_{i,j}\Big)}{\sum_j f_j} \quad \text{(A.2)}$$

Let $P = \sum_j f_j\big(r_{a,j} - \bar{r}_a\big)^2$. Substituting $\bar{r}_a = \frac{\sum_k f_k r_{a,k}}{\sum_k f_k}$ and simplifying, we obtain:

$$\begin{aligned}P &= \frac{\sum_j f_j\Big(\sum_k f_k\ \ r_{a,j} - \sum_k f_k r_{a,k}\Big)^2}{\Big(\sum_k f_k\Big)^2} \\ &= \frac{\Big(\sum_k f_k\Big)^2\Big(\sum_j f_j r_{a,j}^2\Big) - 2\Big(\sum_k f_k\Big)\Big(\sum_k f_k r_{a,k}\Big)\Big(\sum_j f_j r_{a,j}\Big) + \Big(\sum_j f_j\Big)\Big(\sum_k f_k r_{a,k}\Big)^2}{\Big(\sum_k f_k\Big)^2}\end{aligned}$$

Since $k \in I_a \cap I_i$, the above reduces to:

$$P = \frac{\Big(\sum_j f_j\Big)\Big(\sum_j f_j r_{a,j}^2\Big) - \Big(\sum_j f_j r_{a,j}\Big)^2}{\sum_j f_j} \quad \text{(A.3)}$$

Similarly, let $Q = \sum_j f_j\big(r_{i,j} - \bar{r}_i\big)^2$ and, after simplification, we obtain:

$$Q = \frac{\Big(\sum_j f_j\Big)\Big(\sum_j f_j r_{i,j}^2\Big) - \Big(\sum_j f_j r_{i,j}\Big)^2}{\sum_j f_j} \quad \text{(A.4)}$$

Substituting (A.2), (A.3) and (A.4) into (A.1), we obtain:

$$w(a,i) = \frac{\big(\sum_j f_j\big)\big(\sum_j f_j r_{a,j} r_{i,j}\big) - \big(\sum_j f_j r_{a,j}\big)\big(\sum_j f_j r_{i,j}\big)}{\sqrt{UV}}$$

where $j \in I_a \cap I_i$ and

$$U = \Big(\sum_j f_j\Big)\Big(\sum_j f_j r_{a,j}^2\Big) - \Big(\sum_j f_j r_{a,j}\Big)^2$$

$$V = \Big(\sum_j f_j\Big)\Big(\sum_j f_j r_{i,j}^2\Big) - \Big(\sum_j f_j r_{i,j}\Big)^2$$

Appendix B

k–Nearest Neighbour Neighbourhood Size Experiments

The k–nearest neighbour neighbourhood formation scheme forms fixed sized neighbourhoods which consist of the k users that are most similar to the active user.

The optimal neighbourhood size for the k–nearest neighbour scheme is chosen by experiment. The neighbourhood size that yields the best performance in terms of predictive accuracy is selected. The general trend for these experiments is that predictive accuracy improves until a certain neighbourhood size is reached, and thereafter begins to deteriorate. These observations are explained by Herlocker *et al.* (1999) as follows. When small sized neighbourhoods are used, poor quality predictions can result for users who do not have any very similar neighbours. In contrast, large sized neighbourhoods can lead to noise for those users who do have close neighbours. Thus, the neighbourhood size that provides the best accuracy, when averaged over all the users that are contained in a system, is selected.

Figures B.1, B.2 and B.3 show accuracy calculated according to normalised MAE versus neighbourhood size for the MovieLens, EachMovie and Smart Radio datasets, respectively. The trends outlined above are evident for the MovieLens and EachMovie datasets. It can be seen from the figures that the neighbourhood sizes that provide the best accuracy performance across the similarity metrics evaluated[1] are 35 and 45 for MovieLens and EachMovie, respectively. For the Smart Radio dataset, predictive accuracy initially improves as neighbourhood size is increased, but did not deteriorate when neighbourhood size is further increased. From the above discussion, this result indicates that few, if any, users in the Smart Radio dataset have very similar neighbours. The lack of closely matched users can be expected in the context of this dataset, since it contains only 63 users in total, compared to the 943 and $1,000$ users that are contained in the MovieLens and EachMovie datasets. In this thesis, a neighbourhood size of 14 is selected for experiments that are performed using the Smart Radio dataset. Note that, in all cases, the

[1]The significance weighting algorithm extension is used in conjunction with all similarity metrics.

accuracy provided by the correlation metrics exceeds that provided by Cosine similarity. This finding is in agreement with Breese *et al.* (1998).

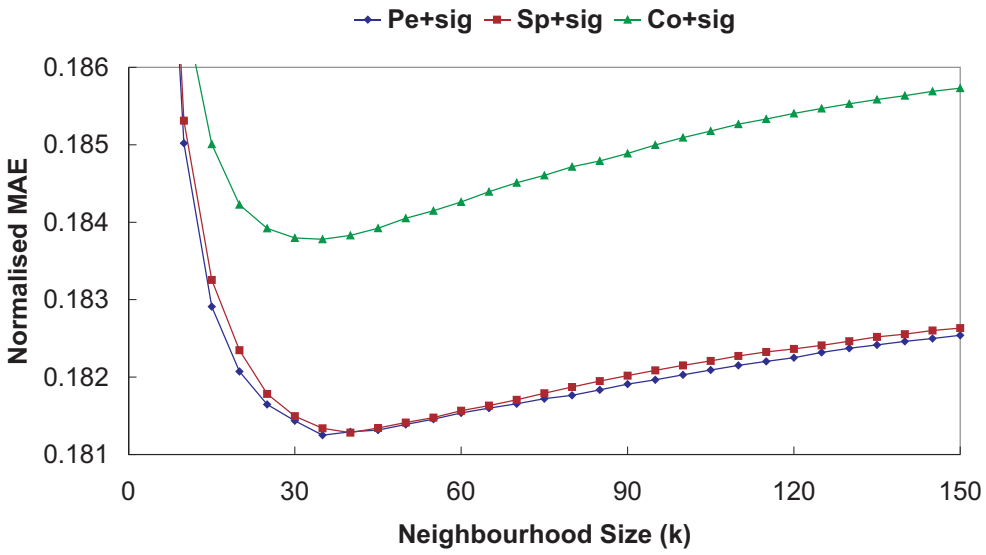


Figure B.1: k–Nearest Neighbour: accuracy versus neighbourhood size for the MovieLens dataset

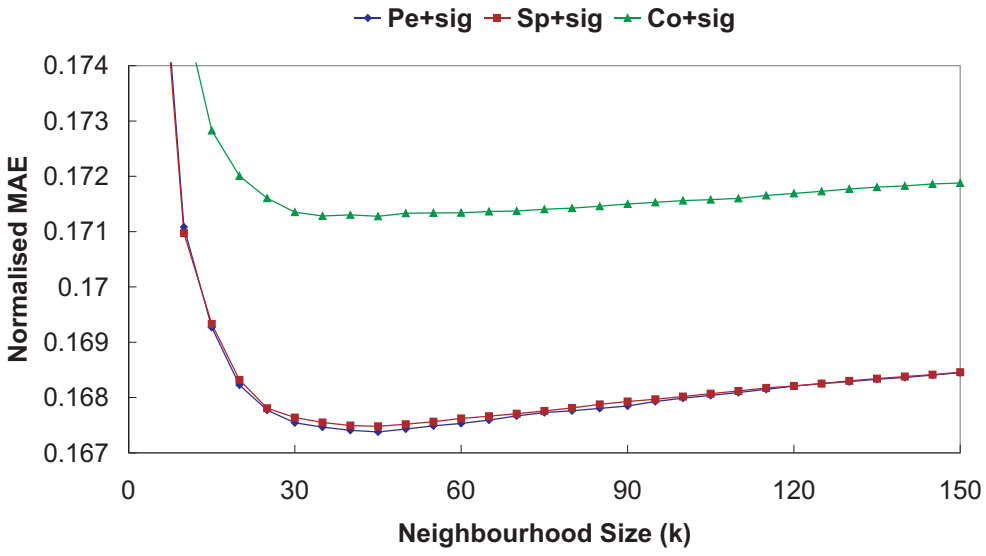


Figure B.2: k–Nearest Neighbour: accuracy versus neighbourhood size for the EachMovie dataset

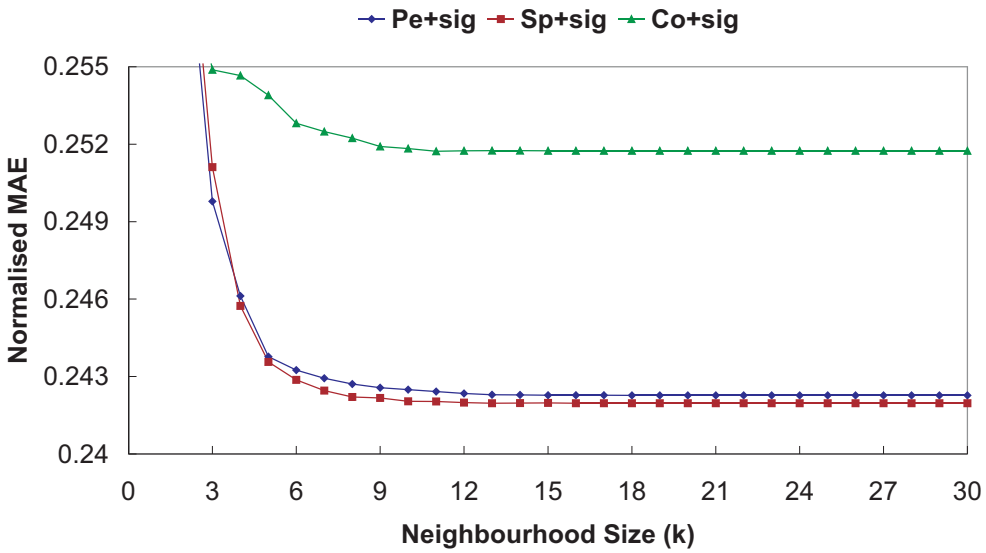


Figure B.3: k–Nearest Neighbour: accuracy versus neighbourhood size for the Smart Radio dataset

Appendix C

Combined Nearest Neighbour and Thresholding Neighbourhood Size Experiments

This scheme combines the features of the k–nearest neighbour and similarity weight thresholding schemes, where neighbourhoods of fixed size are constructed with users whose similarity to the active user exceeds a particular threshold value.

The threshold criterion that is applied by Lam and Riedl (2004) is 0.1, and all users with similarities that are less than the threshold value are excluded, including those that have a *negative* similarity with the active user. The same approach is adopted for experiments conducted in this thesis. As is the case for the k–nearest neighbor neighbourhood formation scheme, the optimal neighbourhood size for this scheme is also chosen by experiment, with the neighbourhood size which yields the best performance in terms of predictive accuracy being selected.

Figures C.1, C.2 and C.3 show accuracy calculated according to normalised MAE versus neighbourhood size for the MovieLens, EachMovie and Smart Radio datasets, respectively. The same general trends are observed in the figures as are noted in Appendix B for the k–nearest neighbor scheme. Note, however, that for all the similarity metrics that are evaluated, the k–nearest neighbor scheme outperforms the combined nearest neighbour and thresholding scheme in terms of accuracy. This finding suggests that, for the Pearson and Spearman rank correlation metrics, the use of negative correlations between users is beneficial to the prediction process. The neighbourhood sizes selected for the MovieLens, EachMovie and Smart Radio datasets, that provide the best performance across all the similarity metrics evaluated, are 35, 45 and 14, respectively.

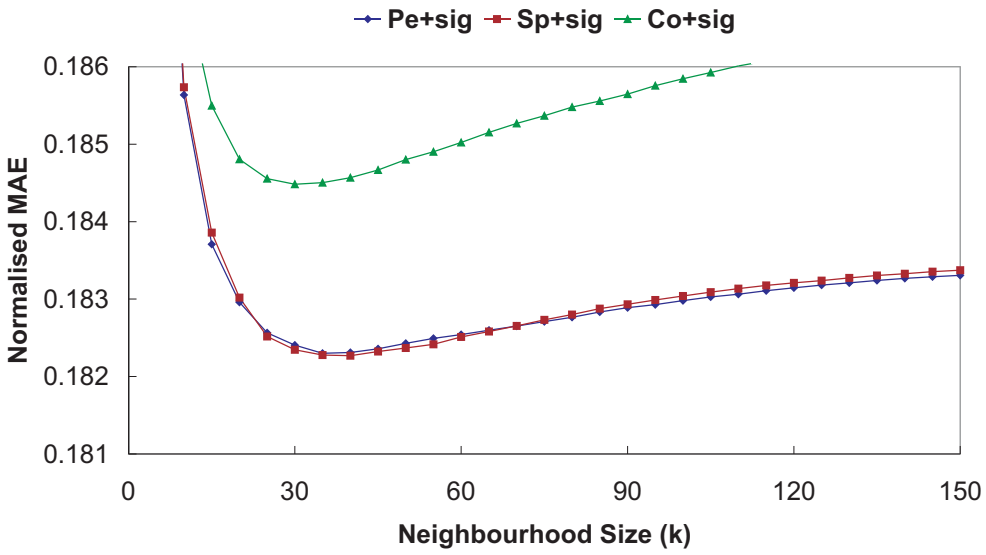


Figure C.1: Combined Nearest Neighbour and Thresholding: accuracy versus neighbourhood size for the MovieLens dataset

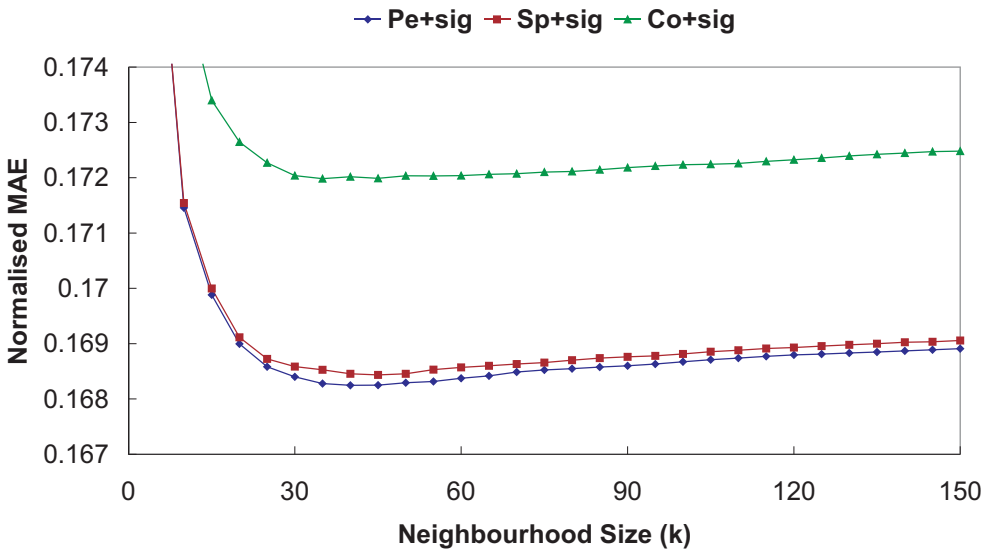


Figure C.2: Combined Nearest Neighbour and Thresholding: accuracy versus neighbourhood size for the EachMovie dataset

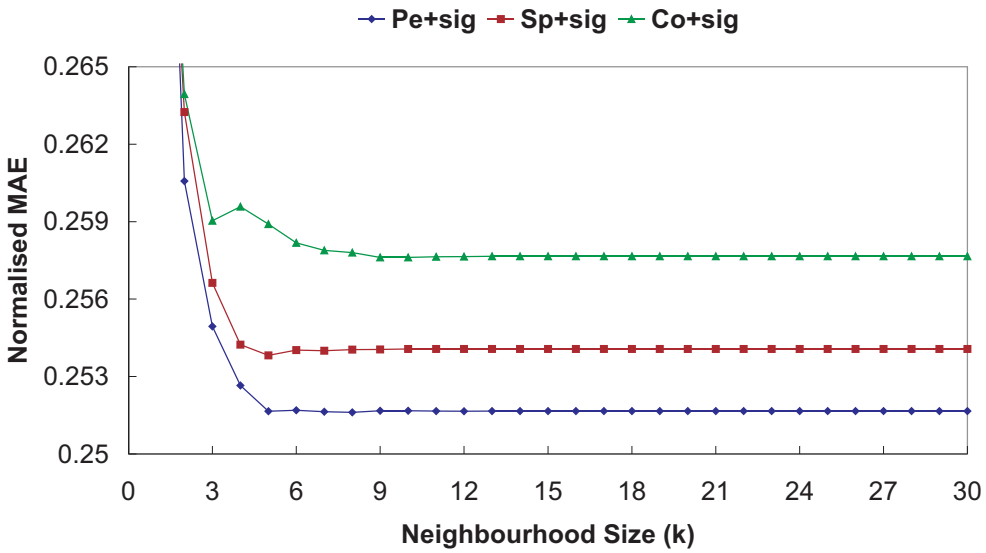


Figure C.3: Combined Nearest Neighbour and Thresholding: accuracy versus neighbourhood size for the Smart Radio dataset

Appendix D

Macnaughton–Smith *et al.* Clustering Algorithm

The divisive clustering algorithm that is proposed by Macnaughton-Smith *et al.* (1964) divides a set into two clusters. Let S be the set of objects to be clustered, and let S_1 and S_2 be the resulting clusters. The algorithm consists of the following steps:

1. Initialise $S_1 = S$ and $S_2 = \emptyset$. It is assumed that $|S| > 1$.

2. **for all** $j \in S_1$ **do**
   $$D(j, S_1 - \{j\}) = \frac{1}{|S_1| - 1} \sum_{k \in S_1,\, j \neq k} d(j, k) \tag{D.1}$$
   end for
   where $d(\cdot, \cdot)$ is some distance function.

3. Move the object for which (D.1) is maximised from S_1 to S_2.

4. **while** $|S_1| > 1$ **do**
   for all $j \in S_1$ **do**
   $$D(j, S_1 - \{j\}) - D(j, S_2) = \frac{1}{|S_1| - 1} \sum_{k \in S_1,\, j \neq k} d(j, k) - \frac{1}{|S_2|} \sum_{l \in S_2} d(j, l) \tag{D.2}$$
   end for
   Let j' be the object for which (D.2) is maximised.
   if $\Big(D(j', S_1 - \{j'\}) - D(j', S_2)\Big) > 0$ **then**
   Move object j' from S_1 to S_2.
   else
   Exit – clustering process is complete.
   end if
   end while