UNIVERSITY OF SOUTHAMPTON

User Profiling using Machine Learning

by

Thomas Charles Barnard

A thesis submitted in partial fulfillment for the
degree of Doctor of Philosophy

in the
Faculty of Physical and Applied Sciences
School of Electronics and Computer Science

August 2012

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF PHYSICAL AND APPLIED SCIENCES
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

by Thomas Charles Barnard

The goal of the Instant Knowledge project was to design a system to facilitate the sharing of knowledge and expertise within a distributed mobile environment. This system automatically builds profiles of experts interests, and automatically recommends them based on context and social networking information. This thesis describes my contributions to the IK project which involves profiling users and making recommendations using machine learning techniques.

Recommender systems are information filtering systems which recommend items to users based on a model of their preferences. Recommenders suffer from a number of problems: they do not make use of contextual information, so recommendations may be untimely or inappropriate; they often use a centralised architecture, which makes it difficult to react to the changing needs of users; they are often implemented in an ad-hoc fashion making it difficult to make principled improvements or add extra information.

In this thesis I present a probabilistic recommender based on Bayes' theorem. Rating behaviour is modelled using a Bayesian prior to improve performance in conditions of data sparsity. The best results are obtained using a Gaussian model for user ratings, and a Gaussian-gamma model for co-rating behaviour. The use of a probabilistic framework should make it easier to add context information to the recommendation process.

Generating profiles automatically carries the risk of accidentally including private information which may be discovered by querying the Instant Knowledge system. This presents a privacy risk, as private information may be accidentally incorporated into experts' profiles. I present a framework for evaluating the effect of contamination on performance, and the ability of filtering techniques to preserve privacy. Several filtering techniques are tested and I show that supervised and semi-supervised naïve Bayes classifiers can help to preserve privacy.

Contents

Declaration of Authorship xv

Acknowledgements xvii

1 Introduction 1
- 1.1 Instant Knowledge . . . 1
- 1.2 Overview . . . 2
  - 1.2.1 Recommender Systems . . . 2
  - 1.2.2 Privacy . . . 4
- 1.3 Publications . . . 5

2 Recommender Systems 7
- 2.1 Overview of Recommender Systems . . . 8
- 2.2 Content-Based Recommender Systems . . . 9
  - 2.2.1 A Simple Content-Based Recommender System . . . 10
  - 2.2.2 Problems with Content-Based Recommender Systems . . . 11
- 2.3 Collaborative Filtering . . . 12
  - 2.3.1 Problems with Collaborative Filtering . . . 13
- 2.4 Hybrid Recommender Systems . . . 13

3 Collaborative Filtering 17
- 3.1 Memory-Based Collaborative Filtering . . . 17
  - 3.1.1 Similarity Calculation . . . 18
  - 3.1.2 Neighbourhood Calculation . . . 19
  - 3.1.3 Prediction . . . 19
  - 3.1.4 Refinements . . . 19
- 3.2 Item-Based Collaborative Filtering . . . 20
- 3.3 Context-based Recommender Systems . . . 22
  - 3.3.1 Context . . . 22
  - 3.3.2 Incorporating Context into Recommendation . . . 22
    - 3.3.2.1 Context Similarity . . . 23
    - 3.3.2.2 Reduction-Based Contextual Recommendation . . . 24
    - 3.3.2.3 Challenges . . . 24
- 3.4 Distributed Recommender Systems . . . 25
  - 3.4.1 Related Work . . . 26
  - 3.4.2 Example . . . 27
  - 3.4.3 Discussion . . . 29

4 Evaluating Recommender Systems 31
4.1 Evaluation Metrics . . . 31
4.1.1 Coverage . . . 31
4.1.2 Statistical Accuracy . . . 32
4.1.3 Decision-Support Metrics . . . 32
4.1.4 Ranking . . . 33
4.2 Datasets . . . 34
4.2.1 Popular Datasets . . . 34
4.2.1.1 MovieLens Dataset . . . 35
4.3 Experiments . . . 35

5 Probabilistic Recommendation 39
5.1 Related Work . . . 39
5.2 Notation . . . 40
5.3 Bayesian Recommendation . . . 41
5.4 Multinomial Model . . . 42
5.4.1 Dirichlet Prior . . . 43
5.4.2 Dealing with Uncertainty . . . 46
5.5 Gaussian Model . . . 47
5.6 Item-Based Probabilistic Collaborative Filtering . . . 50

6 Recommendation Experiments 51
6.1 Memory-Based Collaborative Filtering . . . 51
6.2 General Results . . . 52
6.3 Sparsity Results . . . 55
6.4 Analysis . . . 59
6.5 Conclusion . . . 60

7 User Profiling 63
7.1 The IK System . . . 63
7.2 Profile Generation . . . 65
7.3 Privacy-Preserving Profiling . . . 66
7.4 Related Work . . . 68
7.5 A Simple Information Retrieval System . . . 69
7.5.1 Text Processing . . . 69
7.5.2 Latent Semantic Analysis . . . 70
7.5.2.1 Singular Value Decomposition . . . 71
7.5.2.2 Dimensionality Reduction . . . 73
7.5.2.3 Interpretation . . . 73
7.5.2.4 Implementation . . . 74
7.5.3 Shortcomings of the System . . . 74
7.6 Evaluating Privacy-Preservation . . . 75
7.6.1 Performance . . . 75
7.6.2 Privacy Preservation . . . 76
7.6.3 Classification . . . 76
7.6.4 Datasets . . . 77
7.6.4.1 Initial Datasets . . . 77

7.6.4.2 Improved Datasets 78

8 Preserving Privacy **83**
8.1 Term Filtering 83
8.2 Projection 84
8.3 Global Document Filtering 84
8.3.1 Implementation 86
8.4 Per-User Filtering 86
8.4.1 Positive Naïve Bayes 87
8.4.2 Bayesian Prior 88
8.4.3 Choice of Prior 89
8.4.4 Document length independence. 89
8.4.5 Implementation 89

9 Evaluating Privacy **91**
9.1 Metrics 91
9.1.1 Information Retrieval 92
9.1.2 Classification 92
9.2 Model Parameters 93
9.3 Positive Naïve Bayes Parameters 94
9.3.1 Bayesian Prior 95
9.3.2 Choice of Prior 96
9.3.3 Document length independence. 96
9.4 Results 97
9.4.1 Classification 97
9.4.2 Privacy Preservation 101
9.4.3 Performance 103
9.5 Conclusion 104

10 Conclusion **109**
10.1 Recommender Systems 109
10.2 Privacy-Preserving Profiling 110

A Additional Privacy Results **113**
A.1 Results at high-dimensions 113
A.2 Additional Classification Metrics 113
A.3 RPrecision Results 120
A.4 Results on small datasets 121

B Additional PNB Parameter Results **133**
B.1 Bayesian Prior 133
B.2 Prior Multiplier 133
B.3 Choice of P(1) 134
B.4 Document Length Independence 136

Bibliography **139**

List of Figures

2.1 Hybrid Recommendation 14

3.1 Peer-to-peer Network 27
3.2 Making a Request 28
3.3 Sending a Response 28

4.1 Ratings Histogram 36
4.2 Ratings Histograms 36
4.2.1 Users 36
4.2.2 Items 36
4.3 Ratings for the Top Items 36
4.3.1 Ratings for the Top Items 36
4.3.2 Ratings for the Top Items Cumulatively 36

5.1 Different 2D Dirichlet (Beta) distributions with the same mean. 45
5.2 Different Dirichlet distributions with the same mean. 45
5.2.1 $s = 1.5$ 45
5.2.2 $s = 15$ 45
5.2.3 $s = 45$ 45

6.1 MAE using different significance weightings. 52
6.2 MAE using different neighbourhood sizes. 53
6.3 MAE using different model sizes. 53
6.4 MAE under conditions of varying sparsity for probabilistic techniques. . . 56
6.5 RMSE under conditions of varying sparsity for probabilistic techniques. . 57
6.6 F1 score under conditions of varying sparsity for probabilistic techniques. 57
6.7 MAE under conditions of varying sparsity for selected techniques. 58
6.8 RMSE under conditions of varying sparsity for selected techniques. 58
6.9 F1 score under conditions of varying sparsity for selected techniques. . . . 59

7.1 Architecture of IK System 64
7.2 Cut-down IK System 65
7.3 Dimensions 1 and 2 of LSA Projection of Datasets 79
7.4 Dimensions 2 and 3 of LSA Projection of Datasets 80

8.1 Cosine similarity between original and reconstructed document vectors. . 85

9.1 Performance under varying model dimensions 94
9.2 Effect of Bayesian prior on high-energy physics dataset. 95
9.3 Effect of prior multiplier on high-energy physics dataset. 96

9.4 Effect of prior choice on high-energy physics dataset. 97
9.5 Effect of document length independence assumptions on high-energy physics dataset. 98
9.6 Classification Performance on Erotica Dataset 99
9.6.1 Precision 99
9.6.2 Recall 99
9.6.3 ROC 99
9.7 Classification Performance on High-Energy Physics Dataset 100
9.7.1 Precision 100
9.7.2 Recall 100
9.7.3 ROC 100
9.8 Classification Performance on Out-of-Context Dataset 102
9.8.1 Precision 102
9.8.2 Recall 102
9.8.3 ROC 102
9.9 Privacy-preservation performance. 106
9.9.1 Erotica MAP 106
9.9.2 High-Energy Physics MAP 106
9.9.3 Out-of-Context MAP 106
9.10 Information retrieval performance using authorship as relevance. 107
9.10.1 Erotica MAP 107
9.10.2 High-Energy Physics MAP 107
9.10.3 Out-of-Context MAP 107
9.11 Information retrieval performance using binary relevance. 108
9.11.1 Erotica MAP 108
9.11.2 High-Energy Physics MAP 108
9.11.3 Out-of-Context MAP 108

A.1 Privacy-preservation MAP 114
A.1.1 Erotica 114
A.1.2 High-Energy Physics 114
A.1.3 Out-of-Context 114
A.2 Privacy-preservation RPrecision 115
A.2.1 Erotica 115
A.2.2 High-Energy Physics 115
A.2.3 Out-of-Context 115
A.3 Performance MAP 116
A.3.1 Erotica 116
A.3.2 High-Energy Physics 116
A.3.3 Out-of-Context 116
A.4 Performance RPrecision 117
A.4.1 Erotica 117
A.4.2 High-Energy Physics 117
A.4.3 Out-of-Context 117
A.5 Class Performance MAP 118
A.5.1 Erotica 118
A.5.2 High-Energy Physics 118

A.5.3 Out-of-Context . . . 118
A.6 Class Performance RPrecision . . . 119
A.6.1 Erotica . . . 119
A.6.2 High-Energy Physics . . . 119
A.6.3 Out-of-Context . . . 119
A.7 Classification Performance on Erotica Dataset . . . 120
A.7.1 Accuracy . . . 120
A.7.2 F1 Score . . . 120
A.8 Classification Performance on High-Energy Physics Dataset . . . 120
A.8.1 Accuracy . . . 120
A.8.2 F1 Score . . . 120
A.9 Classification Performance on Out-of-Context Dataset . . . 121
A.9.1 Accuracy . . . 121
A.9.2 F1 Score . . . 121
A.10 Privacy-preservation RPrecision . . . 122
A.10.1 Erotica . . . 122
A.10.2 High-Energy Physics . . . 122
A.10.3 Out-of-Context . . . 122
A.11 Performance RPrecision . . . 123
A.11.1 Erotica . . . 123
A.11.2 High-Energy Physics . . . 123
A.11.3 Out-of-Context . . . 123
A.12 Class Performance RPrecision . . . 124
A.12.1 Erotica . . . 124
A.12.2 High-Energy Physics . . . 124
A.12.3 Out-of-Context . . . 124
A.13 Privacy-preservation MAP . . . 126
A.13.1 Erotica . . . 126
A.13.2 High-Energy Physics . . . 126
A.13.3 Out-of-Context . . . 126
A.14 Privacy-preservation RPrecision . . . 127
A.14.1 Erotica . . . 127
A.14.2 High-Energy Physics . . . 127
A.14.3 Out-of-Context . . . 127
A.15 Performance MAP . . . 128
A.15.1 Erotica . . . 128
A.15.2 High-Energy Physics . . . 128
A.15.3 Out-of-Context . . . 128
A.16 Performance RPrecision . . . 129
A.16.1 Erotica . . . 129
A.16.2 High-Energy Physics . . . 129
A.16.3 Out-of-Context . . . 129
A.17 Class Performance MAP . . . 130
A.17.1 Erotica . . . 130
A.17.2 High-Energy Physics . . . 130
A.17.3 Out-of-Context . . . 130
A.18 Class Performance RPrecision . . . 131

A.18.1 Erotica . . . 131
A.18.2 High-Energy Physics . . . 131
A.18.3 Out-of-Context . . . 131

B.1 Addition of a Bayesian Prior . . . 134
B.1.1 Erotica Dataset . . . 134
B.1.2 Out-Of-Context Dataset . . . 134
B.2 Parameter Multipliers. . . . 135
B.2.1 Erotica Dataset . . . 135
B.2.2 Out-Of-Context Dataset . . . 135
B.3 Choice of P(1). . . . 137
B.3.1 Erotica Dataset . . . 137
B.3.2 Out-Of-Context Dataset . . . 137
B.4 Document Length Class Independence . . . 138
B.4.1 Erotica Dataset . . . 138
B.4.2 Out-Of-Context Dataset . . . 138

List of Tables

2.1 User-Item Matrix for a Film Recommender system 8

3.1 User-Item Matrix for a Film Recommender system 27

4.1 MovieLens 100,000 Ratings Dataset Details. 35

6.1 Mean Absolute Error. Lines in bold are statistically significant compared with the line directly below at the 95% level. 54
6.2 Root Mean Square Error 55
6.3 F1 Score 56

7.1 Top-25 words in large datasets. 81
7.2 Top-25 words in small datasets. 82
7.3 Large dataset statistics. 82
7.4 Small dataset statistics. 82

Declaration of Authorship

I, *Thomas Charles Barnard*, declare that the thesis entitled

User Profiling using Machine Learning

and the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- where any part of this thesis has been previously submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- parts of this work have been published as deliverables for the Instant Knowledge project, and as Barnard and Prügel-Bennett (2011a) *Experiments in Bayesian Recommendation* and Barnard and Prügel-Bennett (2011b) *Privacy-Preserving Profiling.*

Signed:.....

Dated:.....

Acknowledgements

Thanks to my supervisor, Adam Prügel-Bennett, for his help throughout my PhD, and for stepping in when my original supervisor left. Thanks to my Mum, Dad, and Gran for their support over the last 27 years.

The work reported in this thesis has formed part of the Instant Knowledge Research Programme of Mobile VCE, (the Virtual Centre of Excellence in Mobile & Personal Communications), www.mobilevce.com. The programme is co-funded by the UK Technology Strategy Boards Collaborative Research and Development programme.

Chapter 1

Introduction

In recent years the amount of information available to us all has increased dramatically. The World Wide Web puts a wealth of information at our disposal, and social networking sites allow us to share with our friends and colleagues. It can be difficult to make use of this information effectively, and techniques need to be developed to help sort through this data.

The rise of smart phones means that for many users their primary means of accessing information is on a mobile device. In mobile situations users may not be able to spend the time searching for information they want, so information should be recommended automatically, taking into account the user's situation. Useful information can also be obtained from a user's social network. Current information filtering systems do not make good use of social and contextual information.

1.1 Instant Knowledge

The *Instant Knowledge* (IK) project was intended to help solve the problem of finding and sharing information, providing personalised results in mobile situations by making use of contextual information and users' social networks and connections. The system was intended to minimise the effort required by the users, and the risk of disclosing private and confidential information.

More specifically, the IK project aimed to find a solution to the problem of finding experts within an organisation (Helmhout et al., 2009), motivated by the observation that it is often difficult to keep track of "who knows what". In academia researchers often find out too late that someone was working on a similar problem in the same department, with each unaware of the other's work. Each user's professional interests are automatically collected to build profiles of expertise, which are used to recommend experts to users of the system automatically, based on context.

The IK project was overseen by the *Mobile Virtual Centre of Excellence* (MobileVCE), a not-for-profit organisation whose members include a number of mobile phone network operators, handset manufacturers, and government groups, in collaboration with academia. The universities working on the project were the University of Southampton, the University of Strathclyde, and Royal Holloway University of London (RHUL). At Southampton we were responsible for developing new ways of extracting and processing information, and making recommendations using machine learning techniques. Context, social networking information, and user interfaces were to be provided by Strathclyde, while security and privacy were to be considered at RHUL.

The original requirements for the IK system involved machine learning using social and contextual information in a distributed environment. This scenario presents two main challenges: dealing with data sparsity, and preserving user privacy. Sparsity arises in a distributed environment as the information available from connected peers will be a limited subset of the total information available in the system. Making the best use of this limited information is a challenging task. Privacy is a problem in a distributed environment as when sharing information with peers it may fall into the hands of a third party. When generating profiles automatically there is a risk of incorporating private information accidentally.

1.2 Overview

This thesis is organised into two main parts. The first half looks at recommender systems. The second half of this thesis explores the problem of automatically generating profiles of experts' interests while preserving privacy. The main contributions of this thesis are a Bayesian method of recommendation which copes with conditions of sparsity and can be easily adapted to make use of contextual information, a framework for evaluating privacy-preserving profiling, and several methods of filtering private information.

1.2.1 Recommender Systems

Recommender systems are information filtering systems which recommend items based on models of user preferences. They form models of user preferences based on past behaviour, and recommend unseen items based on this model. For instance a recommender system may suggest films for users to watch based on films they have seen and rated highly in the past. In the context of the IK system recommender systems were chosen to provide recommendations of users, items, and content.

This thesis looks at three problems with recommender systems. First that recommendations are usually made in isolation, without reference to the user's context. Secondly that recommender systems are usually centralised, monolithic systems, that may not

be dynamic enough to react to users' immediate needs. Finally, that the most popular collaborative filtering algorithm is rather ad-hoc, and not grounded in mathematical theory.

Using a principled mathematical approach allows you to pick a model, a probability distribution for example, that matches the features of the domain. Extra information and uncertainty can be dealt with in a optimal way. It allows you to choose between competing models by measuring which models best fits the data, and can provide estimates or bound on errors.

The reasons for rating an item highly can be complex, and may be related to the environment in which this decision is made. Current recommender systems do not, however, make use of this contextual information in making recommendations, and so the recommendations made may be inappropriate or untimely. By adding context to the recommendation process it is hoped that the usefulness of recommendations made can be improved.

Although context-aware recommender systems have been studied before, context-aware recommender systems are relatively new. Context information has been applied to the recommendation of restaurants (Horozov et al., 2006), events (de Spindler et al., 2007), films (Adomavicius et al., 2005), news (Choi et al., 2007), and mobile applications (Woerndl and Groh, 2007); these items are usually sensitive to context. Typically the context information used is physical such as the user's location and the current time, but social context has also been explored in the form of the user's friends (Woerndl and Groh, 2007), or the people around them (de Spindler et al., 2007).

Most recommender systems in use today are centralised, monolithic, recommender systems capable of providing recommendations for millions of users. The sheer number of users of a recommender system such as Amazon's product recommender system requires the use of vast computing resources. Even then, these systems typically update infrequently, meaning that changes to users' ratings are not immediately reflected in the items recommended.

Some work has already been done in the area of distributed recommender systems to try to overcome some of these limitations. By distributing the recommendation process over multiple devices, scalability can be improved, and the system can respond dynamically to the requests of users. The ability to respond more quickly allows the system to react to changing context information in real time.

Collaborative filtering recommender systems often make use of ad-hoc algorithms with little grounding in mathematical theory. Techniques such as memory-based collaborative filtering can achieve good performance, but require cases where there is little information to make predictions to be dealt with explicitly, for example when two users have few ratings in common. By using a probabilistic framework and treating recommendation

as a machine learning problem it should be possible to deal with these cases implicitly.

Previous work on Bayesian recommender systems has focused on simple methods of estimating probabilities, such as normalised rating counts with Laplace smoothing. I investigate two different methods of estimating probabilities, the first uses a multinomial distribution to model the generation of co-ratings, which is augmented with a Dirichlet prior. The second uses a Gaussian distribution to model differences in rating behaviour, to which a Gaussian-gamma prior is added.

1.2.2 Privacy

Increasing amounts of users' time is spent using services which are profile-driven, such as social networks, recommender systems, e-commerce, and content-sharing websites. In the case of social networking sites such as Facebook, users provide large amounts of personal information in the form of status updates, contact information, relationships, interests, addresses, dates of birth, and other details. The information provided helps facilitate communication and the sharing of information between users, and pays for these services through targeted advertising (Chai et al., 2007; Enders et al., 2008; Leitner and Grechenig, 2008).

While users of these services provide their information freely, they may not be aware of how widely this information is made available, and to whom. Companies may be legally obliged to protect users' data, but this cannot directly prevent a privacy breach through negligence, or the actions of a third party. High profile attacks such as the release of private information by the LulzSec hacking group have shown that companies cannot guarantee the security of their users' information (Mansfield-Devine, 2011).

Users may not be aware how the information that they make available online may be combined and misused; criminals could find the address of an individual from a social networking site, determine their location from status updates, and use this information to pick the best time to break into their home. Private information could also be used to perform identity theft.

There have been a number of cases where user privacy has been compromised through user profiling. In 2006 AOL released "anonymised" search queries from over half a million users, but was forced to remove this data as many users were identifiable through their queries (Arrington, 2006). Similar concerns and legal action led to the second Netflix recommendation prize being cancelled, and the dataset for the first prize being made unavailable (Singel, 2010).

Facebook has come under repeated criticism for their treatment of users' private information. In 2007 they launched the Beacon advertising system, which tracked user activity across a number of websites. The system tracked purchases on these websites,

automatically posting this information to their profiles without their consent (Story and Stone, 2007). They eventually added the option to opt-out from broadcasting information, but not the collection of the data itself (Debatin et al., 2009). They have also been criticised for their use of profiling in providing personalised advertisements, which may allow advertisers or other users to discover private information such as sexual orientation (Guha et al., 2010).

These services usually provide the option to tighten privacy settings, the default settings can be too permissive, for example providing the option to "opt-out" rather than "opt-in" as with Facebook's Beacon. The only way to guarantee user privacy is to put the control of their private information in their hands, so that they can decide which information to share, and with whom.

Unfortunately, confusing privacy settings, user agreements, lapses in judgement, or human error may lead to too much information being shared; Generating and maintaining a profile, and ensuring that it contains no private information is a time consuming task. This may lead to profiles becoming infrequently updated, leading to a gap between a user's current interests and their profile.

If users cannot tightly control their private information, these companies nevertheless have legal obligations to protect their users' information, such as the *European Union Data Protection Directive* and its implementations. If information is passed to third parties, for example to facilitate advertising, it should not contain personally identifiable information. These sites have millions of users and so it would be impossible to do this manually.

In this thesis I describe attempts to solve the problem of automatically generating profiles in the context of the IK project, by filtering private data using machine learning techniques. I introduce a set of experiments which can be used to determine the effect of private information on performance and privacy, and the efficacy of privacy-preserving profiling algorithms.

In this thesis I consider privacy to be the safeguarding of private information, and a loss of privacy is a disclosure of private information. Private information is information that does not belong in a public profile of an expert's professional interests. This includes information usually considered private such as medical conditions or sexual preferences, and interests which are irrelevant to an an expert's professional profile such as their hobbies or favourite films.

1.3 Publications

The bulk of this thesis was built on deliverables written for the IK project which have been extended and modified. Chapter 2, Chapter 3, and Chapter 4 originally formed

part of an IK deliverable *D-K3.1 Context-Aware Recommender Systems*, co-authored with Mustansar Ali Ghazanfar, but little remains of the original deliverable. Parts of Chapter 3 were taken from an IK deliverable entitled *D-K3.3 Sharing Mechanisms for Information*.

Parts of the first half of this thesis were published at the 2011 Atlantic Web Intelligence Conference in Fribourg, Switzerland, in a paper entitled "Experiments in Bayesian Recommendation". Parts of the second half of this thesis have been published in a preliminary form at the 2011 WORLDCOMP Data Mining (DMIN'11) conference in Las Vegas, USA, in a paper entitled "Privacy-Preserving Profiling", and have been submitted for consideration to IEEE Transactions on Knowledge and Data Engineering (TKDE).

Chapter 2

Recommender Systems

Life presents us with a bewildering array of choices, from deciding which film to see at the local cinema, and where to eat afterwards, to deciding which song to listen to next, and which attractions to visit while on holiday. On the World Wide Web the number of pages is constantly growing, making it increasingly difficult to find relevant information. In business, it is often difficult to find the right person or company for the job.

It isn't possible to sample everything ourselves to find things we like as this takes time, money, and effort. Instead, it would be better to receive recommendations for items we will probably like. We may receive some recommendations from other people through word-of-mouth, but these recommendations may be inaccurate. Recommender systems are a tool to improve and automate this process (Resnick and Varian, 1997), that aim to solve the problem of information overload: finding relevant information amongst vast amounts of irrelevant information.

Recommender systems are information filtering systems that collect records of user behaviour and utilise machine learning and information retrieval techniques to build models of user preferences, to recommend items based on these models. For instance a recommender system may suggest films for users to watch based on films they have seen and rated highly in the past. They have been successfully deployed on e-commerce websites such as Amazon to recommend products to users based on past purchases (Linden et al., 2003).

The rest of this chapter is organised as follows, in Section 2.1 recommender systems are introduced, in Section 2.2 content-based recommender systems are described, in Section 2.3 collaborative filtering recommender systems are introduced, and in Section 2.4 I talk about hybrid recommender systems.

TABLE 2.1: User-Item Matrix for a Film Recommender system

	Blade Runner	2001: A Space Odyssey	THX1138	Brazil
Rick	4	3	5	5
Rachel	4	3	⊘	4
Dave	5	⊘	5	3
Frank	4	1	5	5
Sam	5	3	⊘	⊘

2.1 Overview of Recommender Systems

An attempt to formalise the process of recommendation was made by Adomavicius and Tuzhilin (2005), who see recommendation as a process of picking items such that they maximise the utility to the user. Most often this utility is a rating on an item, indicating a preference for that item. Formally, let u be a utility function that measures the utility of item n to user m,

$$u : M \times N \to R, \tag{2.1}$$

where R is an ordered set. For each user m $\in$ M, choose that subset of items $n' \in N$ that maximise its utility,

$$\forall_{m \in M}, N'_m = \underset{n \in N}{\operatorname{argmax}}\, u(m, n). \tag{2.2}$$

Recommender systems are needed because this utility function is not completely defined; only utilities (ratings) for certain user-item pairs are present. In order to present the user with items which maximise their utility, ratings have to be predicted for items which they have not yet rated. Ratings made by users on items can be stored in a user-item matrix. An incomplete user-item matrix is shown in Table 2.1, where ratings are on a scale of 1 to 5, and missing ratings are indicated by ⊘.

The main tasks for a collaborating filtering system are as follows (Herlocker et al., 2004),

Annotation in context The system is used to analyse structured discussion postings and determine which ones are worth reading. For example Tapestry (Goldberg et al., 1992), and GroupLens (Resnick et al., 1994), both help find relevant newsgroup postings.

Finding good items The system presents users with a ranked list of recommended items and a corresponding prediction for how much the user would like them. Many systems focus on this task, Ringo for example (Shardanand and Maes, 1995).

Other tasks Other tasks for collaborative filtering recommender systems include the recommendation of a sequence of items, finding all good items, and finding credible recommendations.

The application of a recommender system has a large influence on the method in which the results are presented, as an ordered list of items, or a single good item for instance, and this influences how its performance is evaluated.

Recommender systems use a wide number of techniques to make ratings predictions, but fall into three main classes: content-based systems, collaborative filtering systems, and hybrid systems which combine content-based and collaborative filtering techniques. These approaches will be described in the following sections.

2.2 Content-Based Recommender Systems

Content-based recommender systems recommend items to users based on their content, their properties, or associated labels and metadata. For example, websites may be recommended to users based on their textual content, or films may be recommended to a user based on keywords, a plot synopsis, or genre.

Using the utility notation, the utility $u(m, n)$ of item n for user m depends on the utilities $u(m, n_i)$ assigned by the user m to item $n_i \in N_n$ which are similar to the item n. That is the rating for an item n is predicted based on the ratings user m gave to items $n_i \in N_n$ with similar content.

There are four main steps in content-based recommendation:

1. Information is collected about the items of interest. For example, in film recommendation the titles, genres, actors, and directors of films may be collected.
2. Item ratings are collected from users. These ratings may be on a binary scale (i.e. like or dislike), or a numeric scale, the integers from 1 to 5, for example.
3. User profiles or models are built using the information gathered in the first two steps. Machine learning or information retrieval techniques may be applied in this process.
4. Unrated items are compared with the user profiles, assigning a predicted score to each item based on the strength of the match. The predicted ratings of unrated items are sorted into descending order, and the best items are presented to the user.

For example, in a film recommender system, the system finds similar films to the ones the user has rated highly in the past, and then recommends the most similar unrated films to the user. When presented a user who likes the films Blade Runner and Star Wars, such a system may recommend the Indiana Jones films because Harrison Ford appears in them too, or Alien because it is also a science fiction film.

Content-based recommender systems draw on techniques from the fields of information retrieval and information filtering. The process begins with converting each item or document into a feature vector which contains information about the content and properties of that item. In the case of text documents, the components of this vector correspond to terms which appear in the collection of documents. In the next section the operation of a simple text-based content-based retrieval system will be described.

2.2.1 A Simple Content-Based Recommender System

For a text-based recommender system, the first stage is converting the documents into feature vectors. The documents are first tokenised, producing an unordered list of tokens or terms, by splitting the document along punctuation and whitespace. This is referred to as a "bag of words" representation. So called "stop words" are removed, which are common words such "and", "a", and "the". These words would otherwise have high weight, but carry little information. Rare words are also removed as these also have little discriminating power.

Stemming is performed, which groups different words by their root, removing endings. This increases data density. For example *compute*, *computing* and *computer* are all mapped to the same stem *comput*. Finally the frequency of each word is counted to produce a list of terms and their frequency within the document. This can be represented as a term vector, where each component of the vector corresponds to the weight of a given term in the document. This can be written as,

$$D = (t_0, Wd_0; t_1, Wd_1; ...; t_i, Wd_i), \tag{2.3}$$

where D is a term vector, t_i is a term and Wd_i is the weight associated with that term (Salton and Buckley, 1987).

Term frequency (TF) can be used as a term weight, and is simply a count of how many times a particular term, i, appears in a document, d, denoted by $TF(d, i)$. This can be used to search for documents on a given subject, as important terms are likely to be used more frequently. TF weighting may not be efficient when a term occurs frequently in a collection of documents; the term will have little discriminating power and will reduce the precision of the query as less relevant documents are returned. For example, searching for documents on "computer vision" in a collection of documents on a number of computer related topics, may yield documents on "computer science", or "computer aided design".

An improvement on TF weighting is TF-IDF weighting (Salton and Buckley, 1987). The inverse document frequency (IDF) is used to improve precision. The IDF has the highest value when a term occurs infrequently in a collection of documents. This is defined as

follows for term i,

$$IDF(i) = \log \frac{N}{n_i}, \tag{2.4}$$

where N is the number of documents in the collection, and n_i is the number of documents in which the term i occurs.

The TF-IDF weighting scheme combines TF and IDF weighting to weight terms which are used frequently in individual documents, but rarely in the collection of documents as a whole, the highest. These terms should have the highest discriminating power. The TF-IDF weight for a given term i in a document d is calculated simply by taking the product of the TF and IDF weights,

$$w(d,i) = TF(d,i) \times IDF(i). \tag{2.5}$$

After feature vectors have been created for each document, ratings are collected, which are assumed to be "relevant" or "irrelevant", "like" or "dislike", corresponding to positive and negative training examples respectively. Once ratings have been collected they are used to build a profile. For this purpose a simple heuristic algorithm, based on the Rocchio Algorithm can be used (Joachims, 1997).

Documents classified by the user as positive or negative are used to create prototype vectors,

$$\vec{c} = \sum_{\vec{d} \in C} \vec{d}, \tag{2.6}$$

where C is the collection of documents belonging to a user-defined class.

Given a document $\vec{d'}$ and the prototype vectors corresponding to relevant and irrelevant documents C, a document can be classified by measuring the cosine similarity with the prototype vectors as follows,

$$\begin{aligned} H_{TFIDF}(\vec{d'}) &= \underset{\vec{c} \in C}{\operatorname{argmax}} \cos(\vec{c}, \vec{d'}) \\ &= \underset{\vec{c} \in C}{\operatorname{argmax}} \frac{\vec{c}}{||\vec{c}||} \cdot \frac{\vec{d'}}{||\vec{d'}||}, \end{aligned} \tag{2.7}$$

which picks the prototype vector belonging to the class which has the smallest angle to d', pointing in the most similar direction. Note that this simple model based on TF-IDF vectors is likely to suffer from poor performance because of the different vocabularies used by authors of different documents. This will be addressed in the second half of this thesis.

2.2.2 Problems with Content-Based Recommender Systems

Adomavicius and Tuzhilin (2005) identify several shortcomings with content-based rec-

ommender systems:

Limited content analysis These systems depend on features associated with items, and so there must be a sufficient number of features to differentiate them. The range of features available is limited by the ability to automatically extract them. It may be difficult to distinguish between two different items defined by the same set of attributes; it is difficult to distinguish between a poorly written and a well-written article if both are defined by the same features (Shardanand and Maes, 1995).

Over specialisation These systems only recommend items that are the most similar to the user's profile; the user cannot find any recommendation that is substantially different from ones they have already rated or seen. This problem can be solved by introducing an element of randomness or novelty. The system should not recommend those items to users that are too similar to previously recommended items, such as different news articles describing the same event.

New user problem In order to build models of user preferences, a content-based system requires users to rate a large number of items. No ratings exist for new users, so the system cannot make any recommendations for them. This problem is also called the cold-start problem (Maltz and Ehrlich, 1995).

2.3 Collaborative Filtering

Collaborative filtering recommender systems attempt to overcome some of the limitations of content-based recommender systems by utilising ratings information from multiple users. They build user models based on item preferences, finds users that have similar preferences, and recommend items based on the ratings of those similar users. They work based on the assumption that people who have agreed in the past, will agree in the future (Resnick et al., 1994). For collaborative filtering recommender systems, the utility $u(m, n)$ of item n for user m is based on the utilities $u(m_j, n)$ assigned to item n by those users $m_j \in M$, who are similar to user m.

The first collaborative filtering recommender was Tapestry (Goldberg et al., 1992). Tapestry allowed users to find interesting newsgroup conversations by using explicit annotations such as "interesting" or "uninteresting" or implicit ratings, such as the fact that a given user has contributed to a conversation. Tapestry still required the user to do some work in specifying queries, similar users are not identified automatically, and recommendations are not personalised.

The GroupLens project, like Tapestry allowed users to receive newsgroup recommendations, however it added a number of improvements; similar users are found and used in

making recommendations automatically, and the system has a distributed architecture, allowing for multiple ratings servers (Resnick et al., 1994). Another early collaborative filtering recommender systems was Ringo (Shardanand and Maes, 1995). Ringo was a web and email based system for recommending artists and albums based on music users have previously rated highly.

Collaborative filtering recommender systems have several advantages over content-based recommender systems (Shardanand and Maes, 1995). Collaborative filtering recommender systems work with items whose content is not readily processed by computers, for example films, games, and videos. They may suggest items which are quite different in terms of content to those rated highly by a user. Finally there is a measure of quality built in, recommended items are more likely to be of a high quality as well as matching preferences. The operation of collaborative filtering recommendation systems will be explored in more detail in the coming chapters.

2.3.1 Problems with Collaborative Filtering

While collaborative filtering recommender systems have a number of advantages over content-based recommender systems, they have their own disadvantages,

New user problem Collaborative filtering works by finding similar users based on their ratings, but it is not possible to obtain ratings for a new user with no ratings. Several techniques have been developed to attempt to alleviate this problem including hybrid systems (Burke, 2002), ontologies (Middleton et al., 2001), item entropy, item popularity, and personalisation.

New item problem Collaborative filtering relies on the preferences of users for making recommendation. When a new item is added to the system, it isn't possible to recommend this item to users based on rating behaviour. Along with the new user problem, this is referred to as the cold start problem (Maltz and Ehrlich, 1995) and similar methods can be used to overcome it.

Sparsity In most recommender systems, the percentage of ratings made by users compared to the number of possible ratings is very low. If an item is rated by few users, the system will rarely recommend that item. When sparsity is high there is likely to be less overlap between ratings, and so finding similar users or items becomes more difficult.

2.4 Hybrid Recommender Systems

Hybrid recommender systems combine elements of content based systems and collaborating filtering based systems to avoid the limitations of both. Recommender Systems

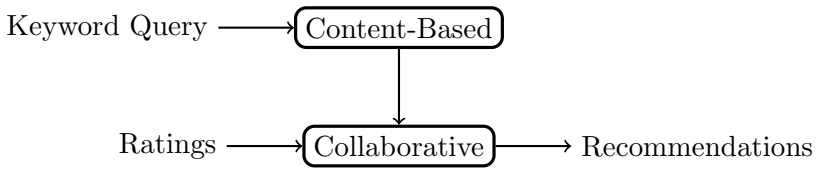


FIGURE 2.1: Hybrid Recommendation

falling into this category include Hayes and Cunningham's *Smart Radio* system which uses the user's current playlist as context to order a list of playlist recommendations created by a conventional collaborative filtering recommender system (Hayes and Cunningham, 2004). This is also the approach taken in Woerndl and Groh (2007) where hybrid recommender systems are used to recommend mobile applications based on physical context.

Content-based and collaborative filtering systems can be combined in the followings ways,

Combining their predictions of both systems The results of each recommender system are either combined, or the results from the system which provides the best results are chosen. Fab is an example of a system which uses this approach (Balabanovic, 1997).

Add content features to a collaborative filtering system This approach uses a content-based profile in addition to collaborative filtering. An example of this approach is the filter-bot approach taken by Good et al. (1999) which uses agents to make ratings based on item content in lieu of actual user ratings.

Implementing a general model that uses both techniques Content and collaborative features are combined to make recommendations. Techniques such as case-based reasoning, probabilistic methods, or ontologies can be used.

A simple type of hybrid recommender system is a cascading hybrid recommender where the output from one recommender system is refined by passing through another. An illustration of this concept is given in Figure 2.1. In this case the content-based recommender is used as a filter to cut down the set of items considered by the collaborative filtering recommender.

CF recommender systems usually operate on two-dimensional data: users and items. Adding additional information such as context to the recommender system adds an extra dimension (or multiple dimensions) to the item-user matrix. This also increases

the complexity of the recommendation, as well as increasing the sparsity problem as ratings are spread thinly over multiple dimensions (Woerndl and Groh, 2007).

Chapter 3

Collaborative Filtering

Collaborative filtering recommender systems produce recommendations of items independent of their content, and are therefore applicable in situations were it is easy to collect ratings, but difficult to process and interpret item content, for example when recommending films, pictures, or computer games. They are also useful for recommending objects which may have no digital representation, such as people or places.

The IK project required a recommender system which could provide recommendations of people and places based on contextual information and social networks. This system would be implemented as a hybrid recommender system, with the result produced by a content-based recommender being augmented by ratings, queries, and social networking information.

Collaborative filtering algorithms can be divided into two main classes, memory-based and model-based. Memory-based algorithms use the user database or ratings matrix directly to make predictions. Model-based algorithms use the user database to learn a model of rating behaviour and make predictions based on this model. Breese et al. (1998) found that predictive accuracy in model-based methods was higher than in memory-based methods, additionally they use less memory and run more quickly once the model has been produced. However the models can take a significant amount of time to set-up and update. In this chapter I will focus on memory-based algorithms, though a type of model-based recommender will be looked at in the next chapter.

3.1 Memory-Based Collaborative Filtering

Memory-based collaborative filtering algorithms directly use the ratings matrix to make predictions. Shardanand and Maes (1995) identifies three main steps in the recommendation process,

- Users rate items, these ratings make up their profile. These ratings are typically integers on a scale from 1 to 5 or 1 to 10, but may be binary.

- The system finds similar users to this user by comparing their profiles. There are a number of different ways similarity can be measured.

- The information in these similar users' profiles are used to make predictions and generate recommendations for the user.

In the next sections each of these steps will be described in detail.

3.1.1 Similarity Calculation

After a set of profiles have been built up the next step is to find similar users to the active user for whom predictions are to be generated. The two main ways of determining similarity in a collaborative filtering recommender systems are correlation and vector-based similarity.

Correlation is typically calculated using the Pearson correlation coefficient between users' ratings, which is calculated as described by Breese et al. (1998),

$$w(a,i) = \frac{\sum_j (v_{a,j} - \bar{v}_a)(v_{i,j} - \bar{v}_i)}{\sqrt{\sum_j (v_{a,j} - \bar{v}_a)^2 \sum_j (v_{i,j} - \bar{v}_i)^2}}, \tag{3.1}$$

where $w(a,i)$ is the correlation between users a and i, $v_{a,j}$ is the rating given for item j by user a, $\bar{v_a}$ is user a's mean rating and $\sum_j$ is a sum over items for which users a and i have both provided ratings. The values produced by this algorithm are between -1 and 1, where -1 denotes negative correlation, 1 positive correlation and 0 no correlation.

Vector similarity is determined by taking vector cosines. It measures the degree to which two vectors point in the same direction. As there are no negative ratings, values produced by this algorithm are restricted to 0 and 1, where 0 similarity indicates the two vectors are orthogonal and 1 that they are pointing in the same direction. Cosine similarity can be calculated as follows,

$$w(a,i) = \frac{\sum_j v_{a,j} v_{i,j}}{\sqrt{\sum_{k \in I_a} v_{a,k}^2 \times \sum_{k \in I_i} v_{i,k}^2}}, \tag{3.2}$$

where I_a is the set of items rated by user a. Breese et al. (1998) found that cosine similarity performed slightly worse than Pearson correlation.

3.1.2 Neighbourhood Calculation

Before calculating ratings predictions, a "neighbourhood" of similar users must be calculated. For a given user-item pair, the neighbourhood consists of users who have rated the item and have non-zero similarity with the active user. Usually this neighbourhood does not contain all of the users who have rated the item of interest, but a subset is selected to speed up calculation and ensure accurate predictions.

The two main ways to pick a neighbourhood are to pick the top-n most similar users to the active user who have rated the item, or to pick all users whose similarity is above some threshold. It is also possible to combine these methods, by picking up to n users who are above some similarity threshold. For example, picking up to 100 users whose similarity to the active user is greater than zero. This is the scheme used in the experiments described later in this thesis, as including users with negative similarities significantly degrades performance.

Rather than storing the similarity for every user-user pair, the similarities for the top-n most similar users for each user can be pre-calculated and stored, where n is the model size (Sarwar et al., 2001). Picking a smaller number of similarities to store reduces the storage requirements for the model and speeds up the calculation of the user neighbourhood. The disadvantage of picking a smaller model size is that coverage and accuracy may be reduced.

3.1.3 Prediction

Once a set of similar users has been found the next step is to make predictions on items using their profiles. This is calculated by using a weighted sum of other users' ratings as follows (Breese et al., 1998),

$$p_{a,j} = \bar{v}_a + k \sum_{i=1}^{n} w(a,i)(v_{i,j} - \bar{v}_i), \tag{3.3}$$

where $p_{a,j}$ is the predicted rating for item j by user a, n is the number of users, and w(a, i) is the similarity weighting described earlier. The constant k is a normalising factor, which is calculated as follows,

$$k = \frac{1}{\sum_{i=1}^{n} w(a,i)}. \tag{3.4}$$

3.1.4 Refinements

Although MBCF performs well and produces recommendations reasonably quickly and efficiently (Breese et al., 1998), it requires a number of modifications in cases where data is limited to achieve this performance. One of the most popular similarity measures used

is based on Pearson correlation. In cases where there are few points to fit, it tends to extreme ends of the rating scale, so that similarity scores tend to be overestimated.

Cases where there are few items rated by two users can be removed or given a low weighting, a process called significance weighting (Herlocker et al., 1999). This works by modifying the measure of similarity given by Pearson correlation to,

$$w(a,i) = \frac{min(|I_{a,i}|, n)}{n} \frac{\sum_j (v_{a,j} - \bar{v_a})(v_{i,j} - \bar{v_i})}{\sqrt{\sum_j (v_{a,j} - \bar{v_a})^2 \sum_j (v_{i,j} - \bar{v_i})^2}}, \tag{3.5}$$

where $I_{a,i}$ is the set of items rated by both user a and user i, n is a parameter picked such that when $|I_{a,i}| \geq n$ the significance weighted similarity measure becomes equal to the Pearson correlation. One problem with significance weighting is that n is picked arbitrarily.

Another technique used to improve recommendations when users have few items in common is *Default Voting* (Breese et al., 1998). Default voting inserts a neutral or negative rating where a user has not rated an item, when calculating similarity. For example, using a scale of 1 to 5, 2 or 3 may be used as a neutral rating. Rather than calculating similarities over the intersection of items rated by two users, they are calculated over the union of items rated by two users.

Another version of default voting assumes that there is an additional set of items, unrated by either user, that they would agree upon. In this case the similarity calculation becomes,

$$w(a,i) = \frac{(n+k)(\sum_j v_{a,j} v_{i,j} + kd^2) - (\sum_j v_{a,j} + kd)(\sum_j v_{i,j} + kd)}{\sqrt{\begin{array}{c}((n+k)(\sum_j v_{a,j}^2 + kd^2) - (\sum_j v_{a,j} + kd)^2) \\ \times((n+k)(\sum_j v_{i,j}^2 + kd^2) - (\sum_j v_{i,j} + kd)^2)\end{array}}}, \tag{3.6}$$

where n is the number of unique items rated by either user, k is an arbitrary number indicating the number of items agreed upon, and d is the default vote.

3.2 Item-Based Collaborative Filtering

One problem with memory-based collaborative filtering is that in order to find similar users to make recommendations, it requires comparing every user with every other user, which scales as $O(n^2)$ with respect to computation time, and the space needed to hold these similarity values. The similarities of users with relatively few items in common is likely to change rapidly as new ratings are made, and so these similarity values should be recomputed regularly in order to provide the best results.

Item-based collaborative filtering is a variant of conventional memory-based collaborative filtering which aims to solve some of these problems (Sarwar et al., 2001). Rather than looking at the similarities between users it looks at the similarity between items. These similarities should be more stable than those between users, as they should be computed over more ratings, and are less likely to change over time meaning they can be recalculated less frequently. In a large system there is likely to be far more users than items, so the amount of similarities which need to be computed and store should be lower. Additionally, in conventional memory-based collaborative filtering predicting ratings requires a neighbourhood of similar users to be calculated for each user-item pair, in item-based collaborative filtering the system simply retrieves the similarities of similar items.

In item-based collaborative filtering, Sarwar et al. (2001) suggest making ratings predictions using a weighted sum of the active user's rating on items similar to the item of interest,

$$p_{a,j} = \frac{\sum\limits_{k \in I_a} w(j,k) \times v_{a,k}}{\sum\limits_{k' \in I_a} w(j,k')}, \tag{3.7}$$

where I_a is the set of items rated by user a and $w(j,k)$ is the similarity of item j and item k. I found, however, that better results could be achieved using a modified version of Equation 3.3,

$$p_{a,j} = \bar{v}_j + k \sum_{i \in I_a} w(i,j)(v_{a,i} - \bar{v}_i), \tag{3.8}$$

where $\bar{v}_i$ and $\bar{v}_j$ are now the average ratings for items i and j respectively, $w(i,j)$ is the similarity between items i and j, $v_{a,i}$ is the active user's rating for item i and k is the reciprocal of the sum of similarities. Equation 3.8 is derived from Equation 3.3 by swapping user average ratings and similarities for item average ratings and similarities.

The authors suggest three different similarity measures, the first two being the familiar cosine similarity and Pearson correlation coefficient used in conventional collaborative filtering, except they use item ratings and means rather than user ratings and means. The third similarity measure, referred to as *Adjusted Cosine* similarity, seeks to adjust for user's mean ratings when calculating item similarities. It is calculated as follows,

$$w(i,j) = \frac{\sum\limits_{u \in U_i \cap U_j} (v_{u,i} - \bar{v}_u)(v_{u,j} - \bar{v}_u)}{\sqrt{\sum\limits_{u \in U_i \cap U_j} (v_{u,i} - \bar{v}_u)^2} \sqrt{\sum\limits_{u \in U_i \cap U_j} (v_{u,j} - \bar{v}_u)^2}}, \tag{3.9}$$

where U_i and U_j are the sets of users who have rated item i and j respectively, and $\bar{v}_u$ is the average rating given by user u. In my experiments I found that item-based collaborative filtering using Equation 3.8 for prediction and the adjusted cosine similarity measure outperform basic memory-based collaborative filtering techniques.

3.3 Context-based Recommender Systems

The appropriateness of recommendations may be highly dependent on context. Most collaborative filtering algorithms are insensitive to the user's immediate interests and needs, and any recommendations made may not be appropriate for the current context (Hayes and Cunningham, 2004). Most recommender systems do not capture the context in which a rating is made, nor the reasons for making that rating. Contextual recommendation is an important part of the IK project, which will be explored in this section.

3.3.1 Context

Context can been used to describe both a particular environmental value, for example the weather, and a set of values which contribute to a particular situation, for example being at work. Schmidt et al. (1999) provide this definition "A context describes a situation and the environment a device or a user is in." For each context a set of features is relevant, and for each relevant feature a range of values is determined by the context.

They divide context feature space into two broad categories, each of which can be further divided into three subcategories: human factors including user, social environment, and task, and physical environment including conditions, infrastructure and location. Human factors covers information directly related to the user such their preferences, their social context (friends and colleagues), their mood, their current task or goals, and the activities they are engaged in. Physical environment covers things such as the weather, time, location, light levels and properties of the device.

Brown, P. and Jones, G. (2001) provide another way to categorise context, distinguishing between context features entered manually, and those automatically determined from sensory information. Chen (2005) notes than context can come from multiple sources, either sensors on the device or external services, so availability of all features at all times cannot be relied upon.

3.3.2 Incorporating Context into Recommendation

Few have attempted to incorporate context into recommender systems, and most that have used a single context feature in conjunction with a more conventional collaborative filtering or content-based recommender system. Spatiotemporal information is typically used in these systems to filter lists of location based items such as restaurants or tourist attractions produced by the main recommender system.

Other systems such as *COMPASS* have used location as a "hard criterion" to filter out services not within a certain distance (van Setten et al., 2004). Horozov et al. (2006) uses location in a similar manner, but to filter the ratings of users outside a certain distance of the active user. Apart from spatiotemporal context, others have used emotional context to tailor the presentation of recommendations (González et al., 2007).

There have been a small number of attempts to incorporate multiple context features into recommender systems. CF recommender systems usually operate on two-dimensional data: users and items. Adding context to the recommender system adds an extra dimension (or multiple dimensions) to the item-user matrix. This also increases the complexity of the recommendation, as well as increasing the sparsity problem as ratings are spread thinly over multiple contexts (Woerndl and Groh, 2007).

In the next two sections we will look in detail at two methods for adding context to recommender systems.

3.3.2.1 Context Similarity

Chen (2005) proposes an update to the conventional memory-based collaborative filtering algorithm to incorporate context. When item ratings are entered, a "snapshot" of the context at that moment is associated with the rating. In this implementation an object associated with each available context feature is linked to the rating, for example one for time and one for location.

Ratings prediction is similar to conventional memory-based collaborative filtering, except that ratings are weighted based on the similarity between the context in which they were made and the context in which a recommendation is requested. This rating is described by the following equation,

$$v_{a,j,c} = k \sum_{x \in C} \sum_{t=1}^{z} v_{a,j,x} \, \mathrm{sim}_t(c, x), \tag{3.10}$$

where $v_{a,j,c}$ is the rating of item j by user a weighted by the current context c. k is a normalising factor, the sum of similarities. The outer sum loops over contexts (combinations of context features, being at work for example), and the inner sum loops over context features (time for example). The actual rating made by user a on item j in context x is given by $v_{a,j,x}$. The function $\mathrm{sim}_t(c, x)$ compares the similarity of two values of a given context feature t.

The prediction formula is given the following equation,

$$p_{a,j,c} = \bar{v_a} + k \sum_{i=1}^{n} w(a, u)(v_{i,j,c} - \bar{v_i}), \tag{3.11}$$

where $p_{a,j,c}$ is the predicted rated item for the active user a for item j given the current context c. The main differences between this and Equation 3.3 is that the context-weighted rating $v_{u,j,c}$ is used rather than the normal user rating $v_{u,i}$.

3.3.2.2 Reduction-Based Contextual Recommendation

Adomavicius et al. (2005) suggest a reduction-based approach based on work in OLAP (Online analytic processing) databases. The main idea is that only ratings made in the same context as the active user's current context are used to generate predictions. The technique does not prescribe any particular recommendation algorithm and so may be used with any recommendation algorithm.

The algorithm works by the reduction of a multidimensional ratings matrix to a two dimensional ratings matrix where only items matching the given context are retained. This may leave few ratings in the matrix, and a more general approach is to reduce by contextual segments; time may be segmented into weekdays or weekends rather than filtering by the exact day.

Given the exaggerated sparsity problem in context-aware collaborative filtering, in some cases it is desirable to make use of as much information as possible, and use the ratings made in all contexts not just those matching the current one. This combined approach uses reduction-based CF when it would outperform conventional CF and vice versa. It does, however, require some offline preprocessing to work. Ratings are first segmented into the largest groups possible based on context dimensions, before measuring the performance on each segment. Segments are kept where no more general partition performs better. Where the reduction-based method performs better than the conventional method it is used, and vice versa.

3.3.2.3 Challenges

The two approaches looked at earlier in this section suffer from an exacerbated sparsity problem when compared to conventional collaborative filtering. The addition of an extra dimension of context increases the sparsity by a factor of the number of divisions of the context dimension. For non-context-aware recommender systems it is possible to pre-calculate much of the information used to make recommendations, as it is relatively static. Context-aware recommendations are more dynamic, and many of the calculations have to be done on-the-fly to incorporate the current context.

Context similarity can be measured by correlating ratings, which suffers from the same sparsity problem as measuring similarity between users, and is computationally expensive. The reduction based approach should work well when there is lots of data, but it is more likely that the user-item-context matrix will be very sparse, and filtering it will

create a very sparse user-item matrix. Picking how to segment context dimensions is a problem, and picking the best combinations of features is computationally expensive.

These approaches also fail to take into account unavailable or unreliable context information. For example, if weather context is obtained from a server, that server, or the connection to it, may fail. In the second case, assume that an item is not rated in the context in which it was consumed, but at a later date in a different context. Techniques need to be developed for dealing with situations where context is missing, perhaps by trying to fill in the missing context information. Erroneous context also needs to be dealt with, perhaps by omitting context features that are likely to be incorrect.

3.4 Distributed Recommender Systems

Mobile and distributed scenarios are important for the *Instant Knowledge* project. Most recommender systems use a central server responsible for maintaining user profiles and generating recommendations. Distributed recommender systems are simply recommender systems where all or parts of the recommendation process is spread over multiple devices. They consist of a number of connected devices, each of which contains a personal recommender system responsible for generating recommendations for a single user. A number of advantages and disadvantages of distributed recommender systems compared with centralised recommender systems are listed below:

Scalability As each device in a distributed recommender systems only has to deal with a small subset of the total users and items in the system, adding more users and items will not have much effect on individual recommenders in the system, and so the system will scale better.

Adaptability As the recommender system is running on a user's device it will have access to more information from that user, such as contextual information. The system can be updated dynamically as it only has to store a small amount of information.

Mobility In cases where a distributed recommender system has a personal recommender running on each device, the user does not need a connection to a central server, or to other users to receive a recommendation.

Privacy Distributed recommender systems allow users to decide what information is shared with other users.

No single point of failure If a single device becomes inaccessible in a distributed recommender system, then the rest of the system can function. If a single device is compromised, then only the information contained on that device is affected.

Limited information In distributed recommender systems each device will have access to only a limited subset of the total information in the system. Recommender system algorithms work better with more information and so the quality of recommendations made may be lower than with a centralised recommender system.

Limited Resources Mobile devices have access to much lower resources than servers; computational power, storage, and bandwidth are all reduced. This could result in lower quality recommendations, as algorithms have to be used which will produce results in an acceptable time period, rather than those which will produce the best results.

3.4.1 Related Work

Distributed recommender systems are almost as old as recommender systems themselves. GroupLens was an early recommender system for newsgroup articles (Resnick et al., 1994). This system used the distributed nature of usenet to automatically propagates messages between servers. Special newsgroups were created to store ratings, which were propagated across the system, and recommendations were generated using a collection of servers.

Tveit (2001) described a more sophisticated distributed recommender system based on a peer-to-peer architecture. Requests for recommendations are broadcast across this network of peers as ratings vectors. At each node the similarities between this vector and cached vectors are calculated, and if these vectors are similar enough they are returned, otherwise the vector is broadcast on. Once these vectors have been returned to the querying peer, recommendations can be calculated. This system partially distributes computation, but involves sending ratings vectors across the network, which is a significant privacy issue.

PocketLens is a personal recommender system designed for mobile devices which has much in common with the *Instant Knowledge* project (Miller et al., 2004). The PocketLens project's two main goals are mobility and privacy. PocketLens uses an incremental item-based collaborative filtering algorithm which does not require complete recalculation when more information becomes available, making it more suitable for mobile deployment.

Tribler is a peer-to-peer recommender system for television programmes which uses "zapping behaviour" as an implicit item rating (Pouwelse et al., 2006; Wang et al., 2008a). Tribler uses an epidemic protocol called *BuddyCast*, which uses exploitation and exploration to find similar users. Each user either exchange social networks with a known user (exploitation), or exchanges information with a random user (exploration). In this way the system converges to a state where users are connected to similar users without the use of a centralised server.

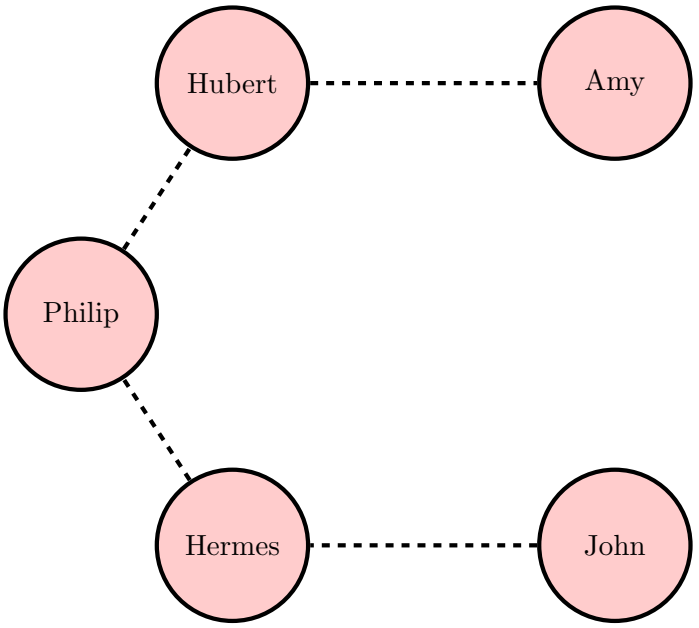


FIGURE 3.1: Peer-to-peer Network

TABLE 3.1: User-Item Matrix for a Film Recommender system

	Blade Runner	Total Recall	Star Wars
Philip	5	⊘	4
Hubert	4	4	5
Hermes	2	⊘	2
John	4	5	3
Amy	5	3	⊘

MobHinter is a mobile ad-hoc distributed recommender system which uses an epidemic protocol to form self-organising communities (Schifanella et al., 2008). Similarity networks are built up through the exchange of lists of neighbours and peers, and their ratings, with other users. After the exchange of these lists, users add similar users to their list of neighbours and may discard the rest.

3.4.2 Example

The operation of distributed recommender systems can best be demonstrated with an example. In the following example Tveit's peer-to-peer recommendation method will be used. Here we have a number of users in a distributed film recommendation system, shown in Figure 3.1. The interests of the users in the system are shown in Table 3.1. This is a ratings-matrix as it would appear in a centralised recommender system, but in this case each user will only have access to their own ratings.

Philip wants a recommendation, so he broadcasts his ratings vector through the system to his peers. At each peer the similarity of this vector is compared to locally stored vectors, if a match is found a ratings vectors is returned, otherwise the vector is passed

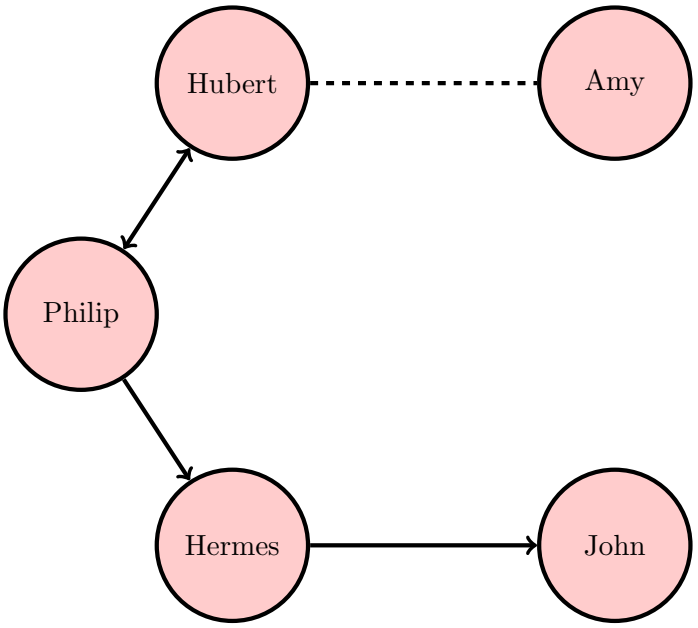


FIGURE 3.2: Making a Request

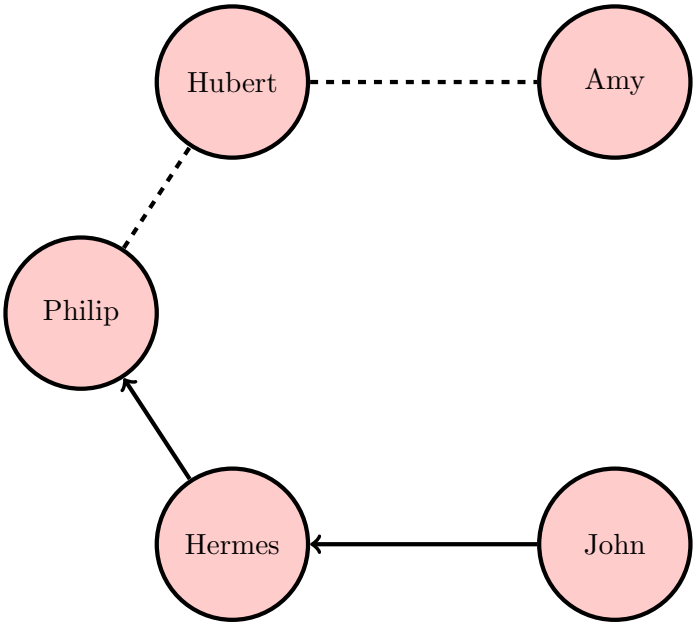


FIGURE 3.3: Sending a Response

on. Hermes finds no match so passes this vector on to John. Hubert on the other hand finds a match, and sends his vector back to Philip. This situation is illustrated in Figure 3.2.

When Philip's ratings vector gets passed to John, a match is found, and so his ratings vector is passed back to Hermes, and then back to Philip. This is shown in Figure 3.3. Now Philip has Hubert and John's ratings vectors, and a recommendation can be produced. Using standard collaborative filtering the recommendation produced is that Philip should see the film "Total Recall".

3.4.3 Discussion

Distributed recommender systems can be placed into two broad categories. Those in which each user provides recommendations for other users, and those in which the information required to generate recommendations is provided to other users.

In the first case recommendations, or similarities are calculated for other users after they have provided their profiles. The advantages of this technique is that it involves sharing less personal information with other users, and reduces network traffic as other users only respond if they have information to send. The disadvantages of this technique are that it involves using processing power, and therefore battery power for other users, and as less information is exchanged this means that each user's personal recommender gathers information more slowly. The user also has to wait for responses to arrive so recommendations take longer to generate.

In the second case user profiles are swapped, and each user's device is responsible for providing recommendations for that user alone. The advantages of this method is that the user's personal recommender builds up enough information to make recommendations when other users cannot be contacted, and recommendations can be generated more quickly as the information exists on the user's device. The disadvantages of this technique are that it involves higher network traffic and so may not be appropriate where bandwidth is limited, it also involves more exchange of data, which may not be acceptable from a privacy standpoint.

Simple distributed recommender systems can be explored for the purposes of evaluation by setting a fixed neighbourhood for calculations, representing users who are connected either directly or through a peer-to-peer network to a given user.

The focus for the IK project moved from a distributed scenario to a client-server architecture. The emphasis on context-based recommendation was also reduced. For these reasons I did not explore these techniques in greater detail, and they are not covered in the rest of this thesis.

Chapter 4

Evaluating Recommender Systems

To improve and develop better recommender systems algorithms, meaningful experiments and performance metrics are needed. These experiments must also be run using appropriate datasets in order to draw meaningful conclusions.

4.1 Evaluation Metrics

Herlocker et al. (1999) divides recommender systems evaluation metrics into three main categories: coverage, statistical accuracy, and decision-support metrics. Breese et al. (1998) suggest that there are two main types of recommender system, those which recommend items one at a time, and those which recommended ordered lists of items. The first class should be evaluated by looking at the accuracy of individual predictions, whilst the second should be evaluated by considering the ordering of lists of recommendations. In the next sections each category of evaluation metric will be examined in more detail.

4.1.1 Coverage

Coverage is the percentage of items for which a recommender system can provide predictions (Herlocker et al., 1999). In a memory-based collaborative filtering recommender system reasons for not being able to provide a prediction for a rating include having no ratings for an item, or having no ratings for a user. Poor coverage is often caused by data sparsity.

High coverage does not indicate high quality recommendations, and similarly low coverage does not indicate poor recommendation performance. Ratings which cannot be

predicted may be for items in which the user has no interest. Coverage may be a useful measure of performance when used with other evaluation methods.

4.1.2 Statistical Accuracy

These metrics work by comparing predicted ratings with ratings which are known and have been withheld from the recommender system. The most commonly used statistical accuracy metric is known as the *Mean Absolute Error* (MAE), and is calculated as follows,

$$|\bar{E}| = \frac{\sum_{i=1}^{N} |p_i - r_i|}{N}, \tag{4.1}$$

where N is the number of predictions made, p_i is the ith predicted rating, and r_i is the corresponding true rating (Shardanand and Maes, 1995). MAE produces scores which are on the same scale as the values being tested, so are easy to understand. For example a MAE of 1 on a 5 point scale corresponds to an error of 1 point on that scale.

Root Mean Squared Error (RMSE) is a metric which gives large errors more significance than small errors, and is calculated as follows,

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (p_i - r_i)^2}{N}}. \tag{4.2}$$

MAE is the most commonly used method of evaluating recommender systems, so using it makes it easier to compare performance with previous recommender systems. However, experiments have shown that when using MAE to evaluate collaborative filtering algorithms they do not perform much better than basic recommender systems which use average item ratings (Herlocker et al., 1999). They do not take into account the main purpose of recommender systems: helping users make decisions (Sarwar et al., 1998).

4.1.3 Decision-Support Metrics

These metrics evaluate the ability of a recommender system to aid users in selecting high quality items. These metrics are based on the premise that recommendations are based on a binary decision, to follow up a recommendation or to ignore it. In order to apply these metrics to predictions made by recommender systems it may be necessary to convert these recommendations to a binary scale. For example a 5-point scale could have a threshold applied where ratings greater than or equal to 4 are considered to be positive (Herlocker et al., 1999).

The simplest of these metrics are precision and recall. Precision is the ratio of true positives to positive results,

$$Precision = \frac{tp}{tp + fp}. \tag{4.3}$$

It measures how well a system retrieves positive results, while minimising false positives. For a recommender systems this is maximised by recommending items a user is known to like, while not recommending items a user is known not to like.

Recall is the ratio of true positives, to true positives and false negatives,

$$Recall = \frac{tp}{tp + fn}. \tag{4.4}$$

It measures how well a system retrieves positive results. For a recommender system it is maximised by recommending items a user is known to like, and not falsely rejecting items a user is known to like.

A more complex metric which combines precision and recall is the F-measure, calculated by,

$$F_\beta = \frac{(\beta^2 + 1) \times \text{Precision} \cdot \text{Recall}}{\beta^2 \cdot \text{Precision} + \text{Recall}}, \tag{4.5}$$

where β is a parameter controlling the weighting of recall versus precision, for instance setting $\beta = 2$ makes recall twice as important as precision. We make use of the F1 measure which gives precision and recall equal importance.

4.1.4 Ranking

Breese et al. (1998) suggest a technique for evaluating recommender systems based on the ordering of a list of recommendations. The utility of a list is the probability of viewing a recommended item times its utility. The utility of an item is taken to be the difference between its rating and the default rating. The list of predictions is sorted in descending order of predicted rating. The utility of a list is calculated as follows,

$$R_a = \sum_j \frac{\max(v_{a,j} - d, 0)}{2^{(j-1)/(\alpha-1)}}, \tag{4.6}$$

where d is the neutral vote and α is the viewing half-life. The half-life is the number of the item on the list such that there is a 50% chance that the user will view that item. They chose a value of 5.

To evaluate a list for one user, actual ratings were used where available, and default ratings were used otherwise. To score a recommender system using this method, the following equation can be used,

$$R = 100 \frac{\sum_a R_a}{\sum_a R_a^{\max}}, \tag{4.7}$$

where $R_a^{\max}$ is the maximum achievable utility for a user a, using their known rating order.

4.2 Datasets

To evaluate recommender systems algorithms requires ratings datasets. When selecting a dataset several key issues have to be considered (Herlocker et al., 2004):

Live user experiments vs. Offline analysis Algorithms can be evaluated through live user experiments, or through offline analysis of datasets. Live user experiments may produce more accurate results than offline analysis, but they are slower and more expensive to perform.

Synthetic vs. Natural datasets Either an existing dataset that matches the properties of the target domain can be used, or one can be synthesised. Synthetic datasets may not match the nature of real data. Natural datasets are readily available and are being used in a large number of projects.

When choosing a dataset, it is important to look at the properties of those datasets. These include domain features, inherent features, and sample features

Domain features This describes the nature of the content being recommended, and the task for which a system is designed. Algorithms should be evaluated with data that matches the domain to ensure that the results obtained are accurate.

Inherent features This describes the nature of ratings including the source of ratings, the scale of ratings, and user and item information.

Sample features This describes the statistical properties of the dataset.

4.2.1 Popular Datasets

The use of well-known recommender systems datasets makes comparison with existing algorithms easier, and avoids the task of creating a new dataset which may be time consuming and expensive. There are a number of published datasets which have become standard datasets, and have been used by many different researchers to evaluate their algorithms. While the task for the IK project was not film recommendation, using these popular datasets made it easier to gauge performance and progress.

The Netflix prize dataset was a large dataset of film ratings containing 100 million ratings on around 18,000 films made by almost half a million users. It was released as part of the Netflix Prize, which was awarded for improving recommendation performance by 10% when compared with the standard Netflix algorithm. The size of the dataset meant that it tested scalability as well as recommendation performance. The dataset was unfortunately made unavailable because of privacy concerns.

TABLE 4.1: MovieLens 100,000 Ratings Dataset Details.

Users	943
Items	1682
Mean Rating	3.5
Median Rating	3
Mode Rating	4
Avg. Ratings/User	106
Avg. Ratings/Item	59
Avg. Items in Common	174
Avg. Users in Common	105

The MovieLens dataset[1] is a popular collaborative filtering recommendation dataset, which contains a collection of ratings made by users on films. Ratings are made on a 5-star integer scale. A number of versions of this dataset are available, the smallest of which contains 100,000 ratings for 1682 films by around 943 users, and is 94% sparse. The dataset also contains demographic information about users, and basic film metadata such as titles and years of release.

4.2.1.1 MovieLens Dataset

I decided to use the MovieLens dataset for my experiments. Table 4.1 gives some details of the 100,000 ratings MovieLens dataset. Figure 4.1 shows the number of ratings made at each point of the ratings scale for the MovieLens 100,000 ratings dataset. The mode of the ratings is 4, with the next most common ratings being 3 then 5, with comparatively few ratings at the bottom end of the scale; Users tend to consume and rate items that they like.

Figure 4.2.1 and Figure 4.2.2 show histograms of ratings per user and ratings per item respectively. Here we see that most users and items have between 0 and 50 ratings each, with a few users and ratings having a much larger amount. Figure 4.3.1 plots the number of ratings for each item, in order of the number of ratings. Figure 4.3.2 plots the same information cumulatively. These graphs show that a small number of popular items account for the majority of ratings, with a far greater number of items in the "long-tail". An eighth of the items account for one half of the total ratings.

4.3 Experiments

The basic experimental methodology for evaluating recommendation algorithms involves splitting the dataset into portions for training and testing, before performing *K-Fold Cross-Validation* (KFCV) on each of the splits. The datasets are split randomly by

[1] http://www.grouplens.org/

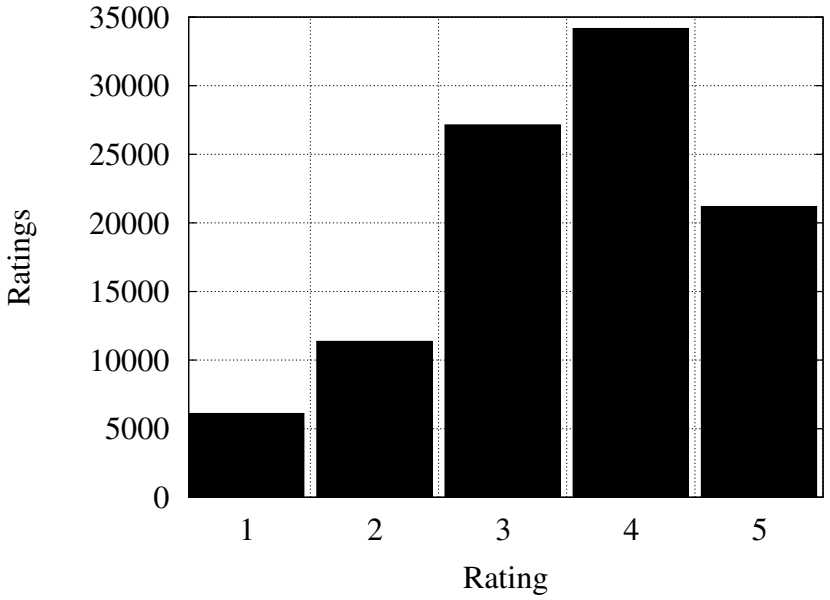
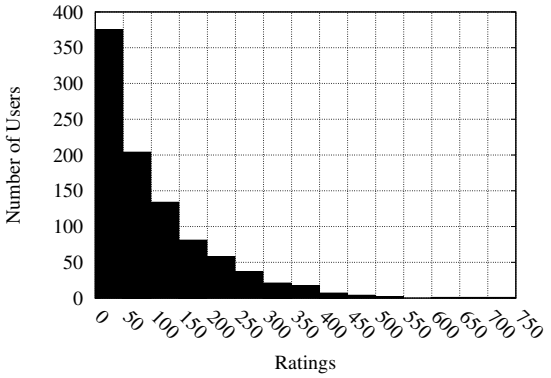
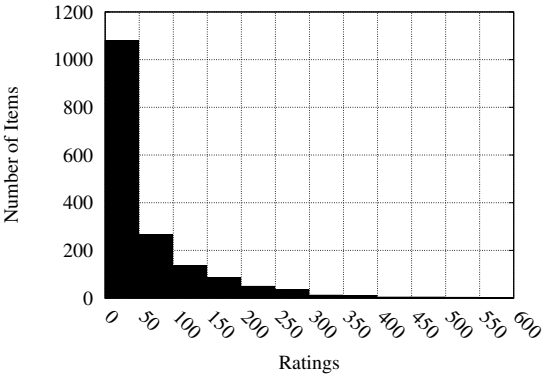


FIGURE 4.1: Ratings Histogram

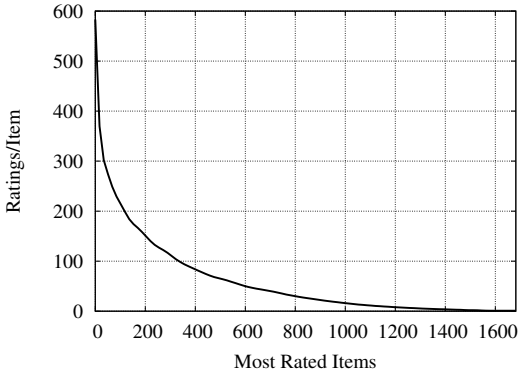


4.2.1: Users

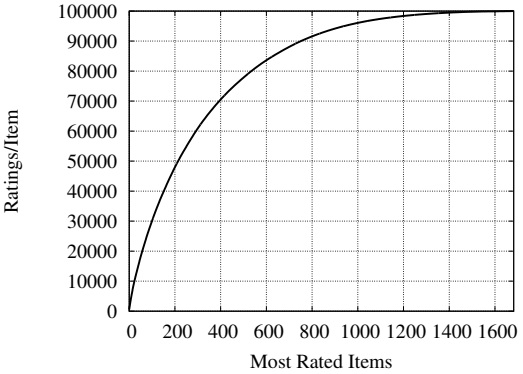


4.2.2: Items

FIGURE 4.2: Ratings Histograms



4.3.1: Ratings for the Top Items



4.3.2: Ratings for the Top Items Cumulatively

FIGURE 4.3: Ratings for the Top Items

rating, that is without splitting each user's ratings directly, and the results from each run are averaged or combined, depending on the evaluation metric used. In these experiments I use 5-fold cross validation.

It is also possible to use *Leave-One-Out Cross-Validation* (LOOCV) which involves using all of the ratings but one to train the system, and using a single hold out rating to test it. This process is repeated withholding each rating once. This may produce better results than KFCV, but is far more time-consuming, even if a small sample of ratings are used to test the system.

As distributed situations are important to the IK project I also decided to look at how techniques perform under conditions of sparsity. After splitting each dataset for k-fold cross validation, a varying proportion of the ratings in the training split are removed randomly.

Chapter 5

Probabilistic Recommendation

Although memory-based collaborative filtering performs reasonably well and produces recommendations quickly and efficiently, as we saw in Chapter 3 it requires a number of modifications in cases where data is limited to achieve this performance. These modifications, though effective, are rather ad-hoc and use arbitrary parameter values to function. It is difficult to make principled improvements to these techniques because they are ad-hoc and not grounded in mathematical theory.

Billsus and Pazzani (1998) note that although machine learning techniques have often been applied by content-based recommender systems, most collaborative filtering recommender systems are based on ad-hoc techniques. They suggest that better collaborative filtering recommender systems can be built by treating recommendation as a classification problem, and making use of tried-and-tested machine learning techniques.

Motivated by the goal of producing recommendations in a more rigorous and robust fashion, we decided to take a probabilistic approach. In this chapter I look at two variations of a Bayesian model for collaborative filtering. After looking at related work in Section 5.1, I look at making recommendations using Bayes' theorem in Section 5.3. I then present two models for modelling user ratings, the first based on a multinomial distribution in Section 5.4, and the second based on a Gaussian distribution in Section 5.5. Results are given in the next chapter.

5.1 Related Work

Breese et al. (1998) present two different models for probabilistic recommendation: a cluster model using a naïve Bayes classifier, and a Bayesian network model. The cluster model groups users based on their rating habits, before predicting ratings given cluster membership using a naïve Bayes model. The Bayesian network is built with one node for each item. In both cases a global model is learnt for all users, and absent ratings

are considered to be significant. Their Bayesian network model was shown to perform comparably to the Pearson-correlation based collaborative filtering algorithm.

Miyahara and Pazzani (2000) present a recommender system based on a naïve Bayes model that makes binary rating predictions (i.e. like or dislike). This approach differs from Breese's as they create a separate model for each user. Ratings are predicted by considering to which class an item belongs based on its features. The classes correspond to the active user's rating, and the features correspond to the ratings made by other users on this item.

The author considers two different methods of treating the data: a sparse model where absent ratings are ignored, and a transformed model where like and dislike are different features, and their absence is informative. They find that their method significantly outperforms memory-based collaborative filtering, with the sparse model performing better than the transformed model.

Robles et al. (2003) present a semi-naïve Bayes approach that takes into account the uncertainty in estimates of conditional probability by making use of confidence intervals. After calculating confidence intervals for each variable they search for the best combination of values within those intervals. They found that their technique outperformed that described in Miyahara and Pazzani (2000) but required a significantly longer training phase.

Wang et al. (2008b) present a probabilistic relevance ranking method that is similar to the item-based collaborative filtering algorithm. Their technique calculates the log-odds of relevance, using a Beta prior to estimate probabilities. Implicit ratings are used to determine relevance though items are divided into just two groups depending on whether or not they are present in a user's profile. To remove the effect of infrequently seen items a threshold is used. This is similar to the heuristic used in memory-based collaborative filtering, although a Bayesian approach is listed as future work.

My approach builds on the simple technique used in Miyahara and Pazzani (2000), but I apply the technique to ratings on a numerical rather than binary scale. The naïve Bayes approach is often overlooked in favour of more complex models. In addition, I incorporate prior knowledge into the probability estimates, as Wang et al. (2008b) do with binary ratings.

5.2 Notation

Before presenting Bayesian recommender it is necessary to define the notation used in the rest of this chapter. The set of all users in the recommender system is given by U, and the set of all items by I. The rating made by user u on item i is given by $r_{u,i} = k$, where $k \in \boldsymbol{K}$, the set of possible rating values, in this case $K = \{x | 1 \leq x \leq 5, x \in \mathbb{N}\}$.

The set of items for which user u has made a rating is given by I_u, and the set of users who have made a rating on item i is given by U_i. Finally, the set of items for which two users u and u' have both provided a rating is $I_{u,u'} = I_u \cap I_{u'}$.

5.3 Bayesian Recommendation

Bayes' theorem allows us to update our beliefs about the likelihood of an event occurring given the evidence. Naïve Bayes models can achieve good performance, despite the strong independence assumptions that are made (Hand and Yu, 2001). The model is said to be naïve in that each feature, each rating for an item in our model, is assumed to be conditionally independent even though this is unlikely to be the case. In the case of recommendation, our beliefs are the probability that a user will make a particular rating on an item, and our evidence is the ratings made by other users.

The assumption that users' ratings are conditionally independent of each other does not seem unreasonable, so it could be argued that this technique is fully Bayesian, and not naïvely Bayesian. This is in contrast to document classification, for example, where the assumption that words in a document are independent of each other is truly naïve .

Putting user ratings into Bayes' theorem gives the following equation,

$$P(r_{u,i} = k|D) = \frac{P(r_u = k) \prod_{u' \in U_i} P(r_{u'} = k'|r_u = k)}{\sum_{k''} P(r_u = k'') \prod_{u' \in U_i} P(r_{u'} = k'|r_u = k'')}, \tag{5.1}$$

where $D = \{r_{u',i}|u' \in U_i\}$, is the set of users' ratings for item i.

We can further simplify the equation if we consider each rating separately, and incrementally update the posterior probability. Each rating is considered one at a time, and the posterior probability given that rating becomes the prior for the next step,

$$P(r_{u,i} = k|r_{u',i} = k') = \frac{P(r_u = k)P(r_{u'} = k'|r_u = k)}{\sum_{k''} P(r_u = k'')P(r_{u'} = k'|r_u = k'')}. \tag{5.2}$$

Once we have calculated the posterior probability of each rating class, we can either pick the most likely value, as in a naïve Bayes classifier, or combine them to find the expected value of the rating $E(r_{u,i})$ as suggested by Breese et al. (1998),

$$E(r_{u,i}) = \sum_{k \in \boldsymbol{K}} P(r_{u,i} = k)k. \tag{5.3}$$

In our experiments we used the most likely class as we found it gives slightly better results, although it makes it harder to use certain evaluation metrics, such as those which require a ranked list.

5.4 Multinomial Model

To make predictions using Bayes rule, we need to estimate both the prior probability of a user making a particular rating $P(r_u = k)$, and the likelihood of a particular rating given another user's rating $P(r_u = k|r_u = k')$.

The multinomial distribution describes a probability distribution over a series of N trials, each of which can have one of a number of independent outcomes, producing a list of counts $\boldsymbol{x} = \{x_1, \ldots, x_k\}$. The parameters of this distribution are the number of trials N, and the probability of each outcome p_k, and has the probability mass function given in Equation 5.4. Here the counts are the number of ratings for each user, or user pair, that fall into a given rating class.

$$f(\boldsymbol{X}|\boldsymbol{p}, N) = \frac{N!}{\prod_k x_i!} \prod_k p_i^{x_i}. \tag{5.4}$$

The problem seems like a good fit to a multinomial distribution. Our evidence is in the form of ratings. In the case of our priors we have a vector of length $|K|$ of the ratings made by a user. We also have a $|K|^2$ matrix of the co-rating counts between each pair of users.

To obtain probabilities from the multinomial model, we can take maximum likelihood estimates by normalising the rating counts,

$$P(r_u = k) = \frac{n_{u,k}}{\sum_{k'} n_{u,k'}}, \tag{5.5}$$

$$P(r_u = k, r_{u'} = k') = \frac{n_{u,u',k,k'}}{\sum_{k''} \sum_{k'''} n_{u,u',k'',k'''}}, \tag{5.6}$$

where $n_{u,k}$ is the number of times user u has given an item a rating k, and $n_{u,u',k,k'}$ is the number of times user u has rated k, when user u' rated k'. This allows us to calculate the conditional probability,

$$P(r_u = k'|r_u = k) = \frac{P(r_u = k, ru' = k')}{\sum_{k''} P(r_u = k, r_{u'} = k'')}. \tag{5.7}$$

As these counts tend to infinity, these probabilities become exact. We have, however, rather fewer than an infinite number of ratings, and we expect that these rating counts will be zero in many cases, particularly for co-ratings. Users are also more likely to rate items that they like (Herlocker et al., 2004), leading to small or zero rating counts at the lower end of the rating scale. This can lead to zero probabilities, and even if only one other user's co-rating probability is zero, the overall posterior probability will be zero.

To remove these zeros we can use a simple form of smoothing called Laplace smoothing.

To do this we simply add one to each count in the preceding equations to obtain the following,

$$P(r_u = k) = \frac{n_{u,k} + 1}{\sum_{k'} n_{u,k'} + |\boldsymbol{K}|}, \tag{5.8}$$

$$P(r_u = k, r_{u'} = k') = \frac{n_{u,u',k,k'} + 1}{\sum_{k''} \sum_{k'''} n_{u,u',k'',k'''} + |\boldsymbol{K}|^2}. \tag{5.9}$$

This technique has been used by Miyahara and Pazzani (2000) achieving better results than MBCF in a binary recommendation problem. Using Laplace smoothing we obtain a MAE of about 0.76, following the same method as in that paper, except with numerical rather than binary ratings. This result is comparable to memory-based collaborative filtering using Pearson correlation. This model is rather simplistic, and although these results are reasonably good it should be possible to do better.

5.4.1 Dirichlet Prior

In the previous section we looked at Laplace smoothing as a simple way of filling in gaps in our knowledge, but we can fill those gaps in what would seem a more intelligent way by using prior knowledge of how users make ratings as a whole.

A way to incorporate this prior knowledge is to make use of a prior over the parameters of our generative distribution. The prior used for the multinomial distribution, Mult(X), is the Dirichlet distribution Dir(α), such that $X \sim \text{Dir}(\alpha)$. It is parameterised by $\boldsymbol{\alpha} = \alpha_i, \ldots, \alpha_k > 0$, which correspond to the number of times a particular outcome k has been observed. Laplace smoothing is a simple case of this where each hyperparameter is set to one.

To use a Dirichlet prior, we first have to learn prior values for the hyperparameters of a Dirichlet distribution, we can then perform a Bayesian update on the hyperparameters given the information available in each individual case of user ratings, or user pair co-ratings. Estimating the parameters of the Dirichlet distribution is made slightly more difficult because ratings are more accurately modelled as being generated by a Dirichlet Compound Multinomial, or Pólya distribution. In this model multinomial distributions are generated by a Dirichlet distribution, and these multinomial distributions in turn generate ratings. This reason for this is that ratings counts are visible, while the parameters of the hidden multinomial distributions are not.

Unfortunately a closed-form solution for estimating the parameters of a Dirichlet distribution given word counts is not available, so we make use of a fixed-point iteration algorithm described in Section 3 of Minka (2003). My implementation was ported from

Minka's Matlab Fastfit[1] library function, "polya_fit_simple", to Python. Given initial estimates of the hyperparameters, the fixed-point iteration is given by,

$$a_k^{new} = a_k \frac{\sum_i \psi(n_{ik} + \alpha_k) - \psi(\alpha_k)}{\sum_i \psi(n_i + \sum_k \alpha_k) - \psi(\sum_k \alpha_k)}, \tag{5.10}$$

where ψ is the digamma function, the first derivative of the logarithm of the gamma function, $\Gamma(n+1) = n!$, and n_i is the total counts for a given observation. Using this equation we can obtain prior parameters using all users rating counts, ignoring users which have a small number of ratings. This iteration is run until the estimates for the hyperparameters converge.

The parameters of the Dirichlet distribution must be greater than zero, but running this algorithm on a sparse dataset where certain outcomes do not occur at all, will lead to many of the parameters being zero. To solve this problem, any zero parameters are set to,

$$\begin{aligned} \alpha_{\text{zero}} &= 0.01 \times \alpha_x, \\ x &= \underset{w}{\operatorname{argmin}}(\alpha_w). \end{aligned} \tag{5.11}$$

The next step is to update these parameters for each user and user pair by using Bayesian inference. For our user priors, we want to infer a probability density function,

$$f(\boldsymbol{p}_u|\boldsymbol{n}_u) = \frac{P(\boldsymbol{n}_u|\boldsymbol{p}_u) f(\boldsymbol{p}_u)}{P(\boldsymbol{n}_u)}, \tag{5.12}$$

where $f(\boldsymbol{p})$ is the Dirichlet distribution $\text{Dir}(\boldsymbol{p}_u|\boldsymbol{\alpha}_u)$. As the Dirichlet and multinomial distributions are conjugate, the posterior is simply a Dirichlet distribution, with parameters $\boldsymbol{\alpha}_u + \boldsymbol{n}_u$. The equations are similar for updating the parameters for co-rating likelihoods.

We can then obtain parameters for our multinomial distribution, by taking the expected value of the Dirichlet distribution,

$$E(\boldsymbol{p}) = \frac{\boldsymbol{\alpha}}{\sum_k \alpha_k}. \tag{5.13}$$

Using this model we obtain a MAE of about 0.86, much worse than even just predicting the user's average rating for each item. By taking the mean we lose information about the Dirichlet distribution. The precision of a Dirichlet distribution is defined be $s = \sum_k \alpha_k$. The higher the precision, the tighter the distribution about its mean. This gives us a measure of how certain we are about the distribution that is lost by taking the expected value.

[1] `http://research.microsoft.com/en-us/um/people/minka/software/fastfit/`

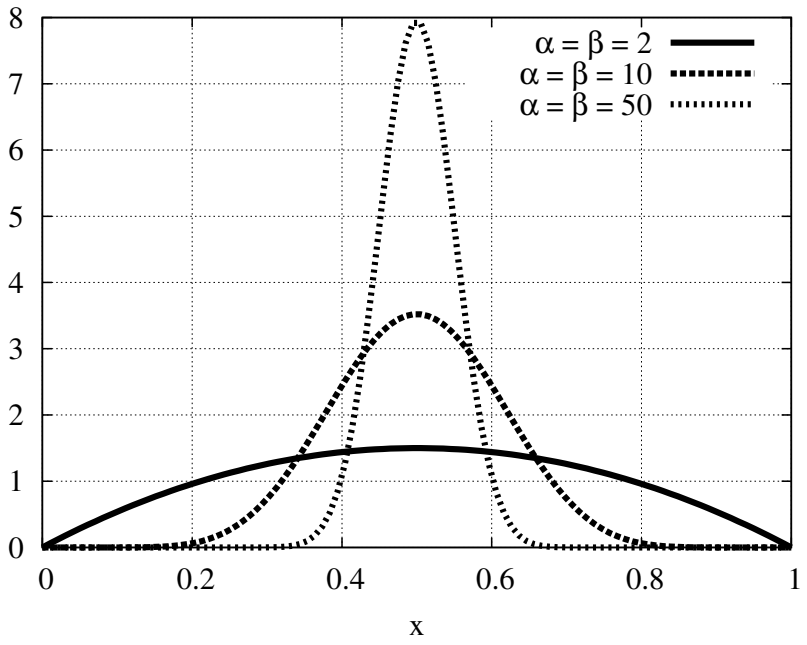
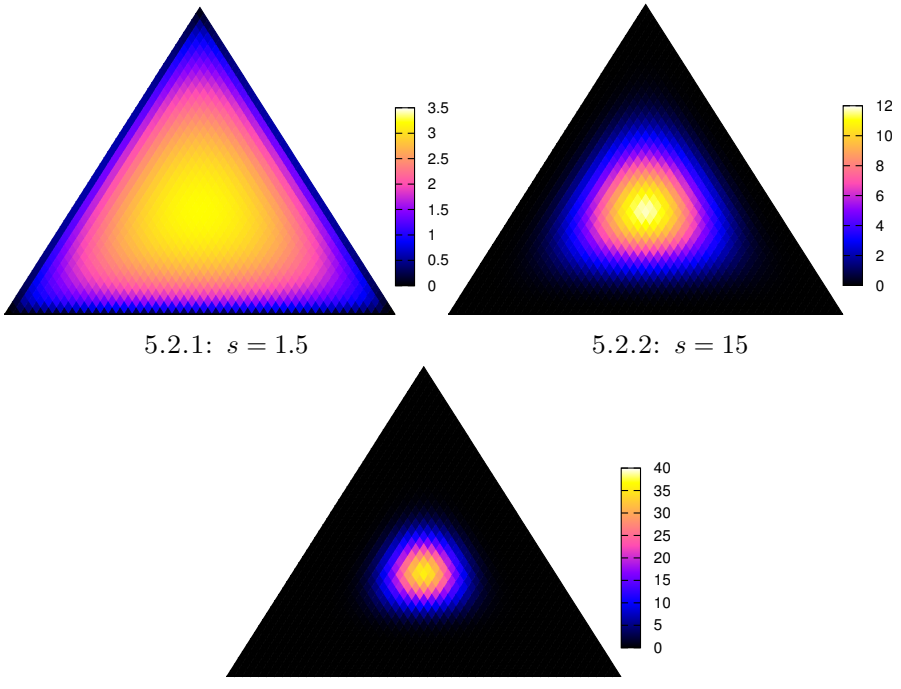


FIGURE 5.1: Different 2D Dirichlet (Beta) distributions with the same mean.

This is illustrated in Figure 5.1, which shows different 2D Dirichlet distributions (Beta distributions), with the same mean but different precisions. Figure 5.2 shows the 2-simplex of 3D Dirichlet distributions with increasing precision showing the same effect.



5.2.1: $s = 1.5$ 5.2.2: $s = 15$

5.2.3: $s = 45$

FIGURE 5.2: Different Dirichlet distributions with the same mean.

5.4.2 Dealing with Uncertainty

Rather than simply taking the mean, we want to integrate out our uncertainty in the parameters,

$$P(r|d) = \int \frac{\boldsymbol{\pi}(r)M(r,d)}{\sum_{r'} \boldsymbol{\pi}(r')M(r',d)} f(M)dM, \tag{5.14}$$

here we change the notation slightly so that $\boldsymbol{\pi}$ is the prior, and M is the likelihood. We also omit the user subscripts, substituting $r_u = k$ for r, and $r_{u'} = k'$ for d. We do not integrate over the prior, as this is taken over a single distribution, where the likelihood involves $|K|$ separate Dirichlet distributions. In the case of the prior the mean of the Dirichlet distribution is simply $E(\boldsymbol{p})$.

No closed form solution exists for this integral, so instead we calculate the integral using a stochastic expansion about the mean of M,

$$P(r|d) = E\left(\frac{\boldsymbol{\pi}(r)(\overline{M}(r,d) + \Delta(r,d))}{\sum_{r'} \boldsymbol{\pi}(r')(\overline{M}(r',d) + \Delta(r,d))}\right), \tag{5.15}$$

where $\overline{M}$ is the expected value of the Dirichlet distribution corresponding to the likelihood, and Δ is a deviation about that mean. Below we derive correction terms for the likelihood by expanding the stochastic expansion. First we add substitutions to simplify the equation,

$$P(r|d) = E\left(\frac{A(r) + D(r)}{A + D}\right), \tag{5.16}$$

$$A(r) = \pi(r)\overline{M}(r,d) \qquad A = \sum_{r'} A(r'),$$

$$D(r) = \pi(r)\Delta(r,d) \qquad D = \sum_{r'} D(r').$$

We rearrange the denominator,

$$P(r|d) = E\left(\frac{A(r) + D(r)}{A\left(1 + \frac{D}{A}\right)}\right), \tag{5.17}$$

and using the series,

$$\frac{1}{1+x} = 1 - x + x^2 - x^3 \ldots, \tag{5.18}$$

we obtain

$$P(r|d) = E\left(\left(\frac{A(r) + D(r)}{A}\right)\left(1 - \frac{D}{A} + \left(\frac{D}{A}\right)^2 \ldots\right)\right), \tag{5.19}$$

$$E(D(r)) = 0 \qquad E(D) = 0.$$

Ignoring higher-order terms, and removing independent terms leaves us with

$$P(r|d) = \frac{A(r)}{A} - \frac{E(D(r)D)}{A^2} + \frac{A(r)E(D^2)}{A^3}, \tag{5.20}$$

$$\begin{aligned} E(D(r)D) &= E\left((\pi(r)\Delta(r,d))\left(\sum_{r'}\pi(r')\Delta(r',d)\right)\right) \\ &= \pi(r)^2\Delta^2(r,d) \\ &= \pi(r)^2\left(\frac{\alpha_d\left(\alpha_0-\alpha_d\right)}{\alpha_0^2\left(\alpha_0+1\right)}\right) \end{aligned} \tag{5.21}$$

$$E(D^2) = \sum_r \pi(r)^2\left(\frac{\alpha_d\left(\alpha_0-\alpha_d\right)}{\alpha_0^2\left(\alpha_0+1\right)}\right). \tag{5.22}$$

After adding these correction terms and running the experiments again we obtain a MAE of around 0.86, little change from the uncorrected results.

The failure of this technique is likely because of several reasons. First, we treat each rating as a separate class, and not as set of integers. The information we have about 2-star ratings should have influence on our knowledge of 1-star and 3-star ratings. The problem is that we learn $|K|$ independent probability distributions, with $|K|^2$ parameters in total for each user prior, and each user pair. In many cases the information is likely to be rather limited, and by spreading the information over these distributions and parameters we exacerbate the problem we wanted to solve. Ratings are smoothed across the conditional distributions in one dimension, where they should be smoothed in two dimensions across the joint probability distributions.

The second problem is that we assume too much information in the case where there are no co-ratings between two users. The Dirichlet distribution is learnt over ratings from users who have ratings in common, and can be used to smooth their ratings. Their ratings may not agree, but the fact that they have rated the same items mean that they have something in common. We cannot say the same about two users who have no commonly rated items. They may share interests but lack common ratings through chance, or they may not have similar interests. If we take that view, and use the Dirichlet parameters learnt for the prior probabilities for the likelihoods as well, we obtain a MAE of about 0.736.

5.5 Gaussian Model

Motivated by the failure of the multinomial model we sought an alternative probability distribution with fewer parameters. Comparing the multinomial model with conventional collaborative filtering, we see that we effectively have $|K|$ independent similarity

measures, compared to one for the MBCF case. Significance weighting effectively adds a second parameter which gives you a measure of confidence in the similarity value.

In order to use a similar concept but in a probabilistic way we can look at the difference between users' ratings over the set of commonly rated items. We model these differences $r_u - r_{u'}$ as being drawn from a Gaussian, or Normal distribution. The distribution is centred around the mean difference between two user's ratings, with a precision parameter which tells us how tightly the distribution is concentrated about this mean. It also gives us an indication of our uncertainty. This model captures the fact that ratings are made on an ordered scale and should not be treated as independent categories.

Our model is not strictly Gaussian, as we make some simplifications, and our likelihoods are discrete, rather than continuous. The formula is

$$P(r_{u'} = k' | r_u = k) = \frac{e^{-\frac{\tau_{u,u'}}{2}\left(k - k' - \mu_{u,u'}\right)^2}}{\sum_{k''} e^{-\frac{\tau_{u,u'}}{2}\left(k'' - k' - \mu_{u,u'}\right)^2}}, \tag{5.23}$$

where $\mu_{u,u'}$ is the mean difference between the two user's ratings, and $\tau_{u,u'}$ is the precision of the Gaussian distribution, or σ^{-2} the reciprocal of the variance. We obtain values for the mean and precision through maximum likelihood estimates,

$$\hat{\mu}_{u,u'} = \frac{1}{|I_{u,u'}|} \sum_i \left(r_{u,i} - r_{u',i}\right), \tag{5.24}$$

$$\hat{\sigma}^2_{u,u'} = \frac{1}{|I_{u,u'}|} \sum_i (r_{u,i} - r_{u',i} - \mu_{u,u'})^2, \tag{5.25}$$

$$\hat{\tau}_{u,u'} = \frac{1}{\hat{\sigma}^2_{u,u'}}. \tag{5.26}$$

In cases where there are few ratings, such that $\sigma^2 = 0$, we set the precision to zero. We would not expect this model to work very well. Where there are few ratings you would expect this model to overestimate the precision.

To prevent this we introduce prior knowledge in the form of a conjugate prior. For a Gaussian distribution this prior is a Gaussian-gamma, or Normal-Gamma distribution. We treat mean and variance as unknown, with mean modelled by a Gaussian distribution, and variance modelled by a Gamma distribution. The Gaussian-Gamma distribution has the following probability density function,

$$GG(\mu, \tau | \mu, \kappa, a, b) \sim N(\mu | \mu, \kappa\tau^{-1}) \Gamma(\tau | a, b). \tag{5.27}$$

As with our multinomial model we must first find initial estimates of our hyperparameters, μ_0, κ_0, a_0, and b_0. We calculate these using our ML estimates of μ and τ in Equation 5.24 and Equation 5.26. Note that we use the parameterisation of the Gamma

distribution where a is the shape parameter, and b is the rate parameter. The alternative parameterisation uses a scale parameter in place of the rate parameter, which is simply $\beta = b^{-1}$.

We obtain formulae to calculate these parameters by looking at the marginal distributions of μ and τ,

$$P(\tau) = \Gamma(a, b), \tag{5.28}$$

$$P(\mu) = T_{2\alpha}(\mu, \frac{a\kappa}{b}), \tag{5.29}$$

where T is a Student's T distribution (DeGroot, 1970).

We have maximum likelihood estimates for the mean and variance of τ, and using the properties of the gamma distribution, we can derive these estimates for its parameters,

$$a_0 = \frac{\hat{\mu}_\tau^2}{\hat{\sigma}_\tau^2}, \tag{5.30}$$

$$b_0 = \frac{\hat{\mu}_\tau}{\hat{\sigma}_\tau^2}. \tag{5.31}$$

We use a similar procedure using the properties of Student's T distribution to obtain an estimate for κ_0,

$$\kappa_0 = \frac{b_0(a_0 - 1)}{\hat{\sigma}_\mu^2}. \tag{5.32}$$

Finally we set μ to zero as our matrix of differences contains $r_{u,i} - r_{u',i}$ as well as $r_{u',i} - r_{u,i}$, these differences cancel out.

Once we have prior values for our hyperparameters, we can perform a Bayesian update using the following equations, which can be found in DeGroot (1970),

$$\mu_n = \frac{\kappa\mu + n\hat{\mu}_{u,u'}}{\kappa_0 + n}, \tag{5.33}$$

$$\kappa_n = \kappa_0 + n, \tag{5.34}$$

$$a_n = a_0 + \frac{n}{2}, \tag{5.35}$$

$$b_n = b_0 + \frac{1}{2}\sum_{i=1}^{n}(r_u - r_{u'} - \hat{\mu}_{u,u'})^2 + \frac{\kappa_0 n(\hat{\mu}_{u,u'} - \mu_0)^2}{2(\kappa_0 + n)}. \tag{5.36}$$

We obtain parameters for our Gaussian distribution by taking the expected values of the mean and precision from our posterior distribution $GG(\mu, \tau|\mu_n, \kappa_n, a_n, b_n)$,

$$E(\mu) = \mu_n, \tag{5.37}$$

$$E(\tau) = \frac{a_n}{b_n}. \tag{5.38}$$

5.6 Item-Based Probabilistic Collaborative Filtering

As memory-based collaborative filtering can be performed using an item-based rather than a user-based approach, so can probabilistic recommendation. Probabilities calculated over users are calculated over items, and vice versa. Here Equation 5.2 becomes

$$P(r_{u,i} = k | r_{u',i} = k') = \frac{P(r_i = k)P(r_{i'} = k' | r_i = k)}{\sum_{k''} P(r_i = k'')P(r_{i'} = k' | r_i = k'')}. \tag{5.39}$$

In my implementation this is done by transposing the ratings matrix, and swapping the item and user indices when requesting recommendations.

Chapter 6

Recommendation Experiments

To compare the performance of the techniques described in this thesis, I implemented several basic recommender system algorithms in Python[1] using Numpy[2] and Cython[3]. The simplest CF recommender systems are those that simply predict the user's average rating for unseen items, and the item's average rating for unseen items. Although simple, these techniques can be surprisingly effective. I also implemented a variety of memory-based collaborative filtering techniques.

In the following sections I use mean-absolute-error, root-mean-squared-error, and the F1 score to evaluate collaborative filtering recommender systems algorithms. For MAE and RMSE lower scores indicate better performance, while a higher F1 score indicates better performance. For these metrics standard errors over the results of 5-fold cross-validation are given.

6.1 Memory-Based Collaborative Filtering

Before comparing MBCF techniques against our probabilistic recommender system we must first find the optimal values of the parameters used to build a model and generate recommendations. Figure 6.1 shows the MAE of the user and item-based MBCF techniques over different values of n used for significance weighting. As the significance weighting increases the MAE decreases, this is because high similarity values based on few common are penalised. At all values of n the item-based technique beats the user-based technique. For both techniques there is a point beyond which increasing the value of n does not improve results. For the user-based technique the optimum value is around $n = 150$, and for the item-based technique $n = 250$. These values are used in the subsequent experiments.

[1] http::/www.python.org
[2] http://numpy.scipy.org
[3] http://www.cython.org

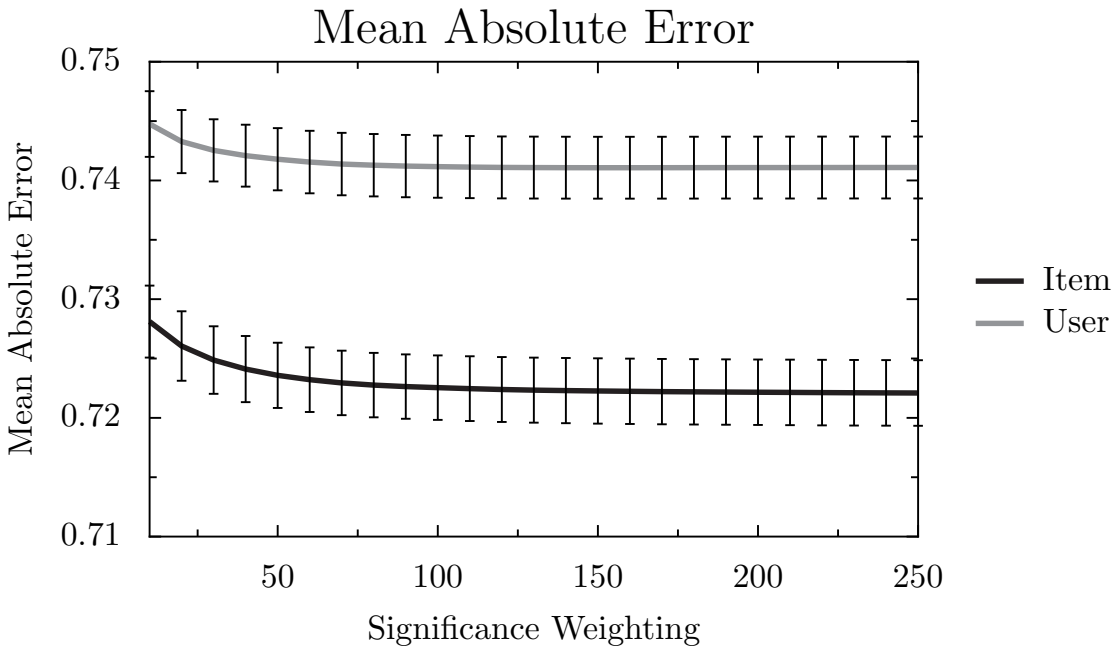


FIGURE 6.1: MAE using different significance weightings.

Figure 6.2 shows the MAE for various MBCF techniques using different neighbourhood sizes. "User" and "Item" are the user and item-based techniques using Pearson correlation and adjusted cosine similarity respectively, with "User (Sig)" and "Item (Sig)" being their significance weighted counterparts. "User (Cos)" is the user-based technique with cosine similarity and no significance weighting. As the size of the neighbourhood is increased the MAE for all techniques decreases. The best performance is provided by the item-based significance weighted technique. As the neighbourhood increases beyond $k = 30$ there is little change in the performance, and so I use this size of neighbourhood in the subsequent experiments. Using a smaller neighbourhood size should speed up ratings prediction.

Figure 6.3 shows the MAE using different model sizes. For both user and item-based techniques reasonable results can be obtained with a model size as small as 100 similar users or items. In the subsequent experiments the similarity between all users and items is retained in order to maximise performance.

6.2 General Results

In this section the results of experiments using the whole user-ratings matrix are presented. The results of each experiment are presented in descending order, so that the best score is at the top. Most of the method names are self-explanatory. The letters given in brackets for each probabilistic method indicate the method used for calculating prior probabilities. "D" indicates a Dirichlet prior, "G" a Gaussian prior, and "GG" a Gaussian-Gamma prior. The presence of "I" indicates that this method is an item-

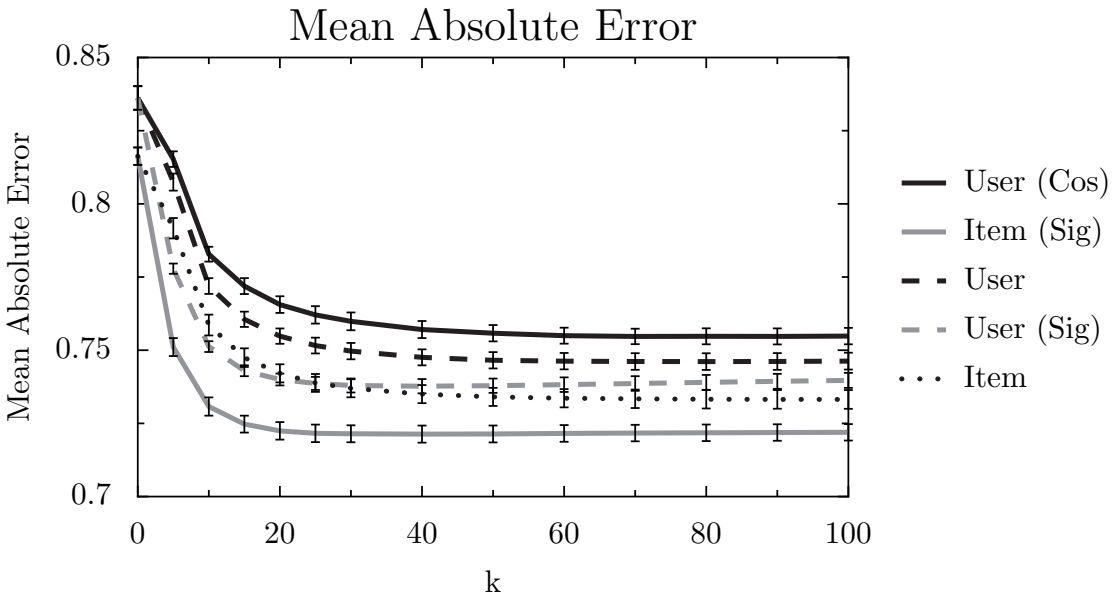


FIGURE 6.2: MAE using different neighbourhood sizes.

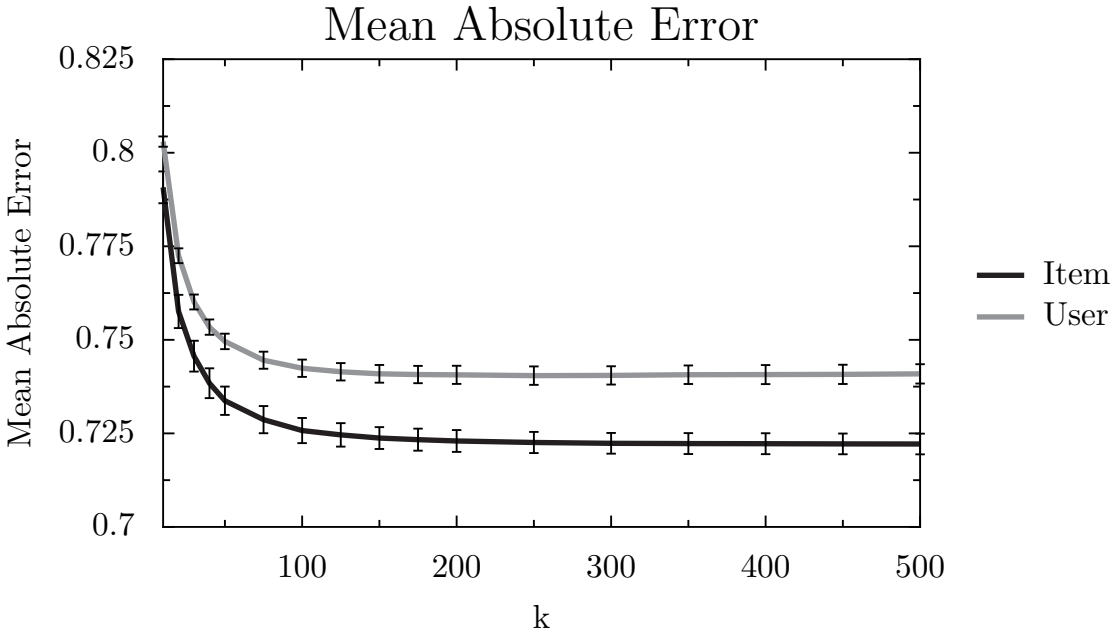


FIGURE 6.3: MAE using different model sizes.

based variant. The version of MBCF with a default vote uses a value of 3 for the default vote. For MAE significance testing is performed using a paired t-test at a 95% level of significance between each technique and the technique below it. Techniques which are significantly different from the technique below are shown in bold. SE indicates standard error.

Table 6.1 shows MAE along with standard error for each of the techniques. The Gaussian methods perform the best, with little difference between each prior used, but the item-based methods perform slightly worse indicating that this model doesn't capture the

Method	MAE	SE
Gaussian (G)	0.7018	0.001466
Gaussian (D)	0.7022	0.001736
Gaussian (GG)	0.7028	0.001603
Gaussian-Gamma (G)	0.7033	0.0008990
Gaussian-Gamma (D)	0.7037	0.001618
Gaussian-Gamma (GG)	**0.7042**	**0.0007326**
Gaussian-Gamma (GI)	0.7069	0.002666
Gaussian-Gamma (GGI)	**0.7072**	**0.002681**
Gaussian-Gamma (DI)	**0.7079**	**0.002709**
Gaussian (GI)	0.7098	0.002567
Gaussian (GGI)	**0.7123**	**0.002527**
Gaussian (DI)	**0.7143**	**0.002703**
Item MBCF (Adjusted Cos)	**0.7217**	**0.002822**
Item MBCF (PCC)	0.7366	0.002341
PCC MBCF (Weighted)	**0.7380**	**0.002387**
Item MBCF (Cos)	**0.7407**	**0.001935**
PCC MBCF	**0.7497**	**0.002798**
PCC MBCF (Default Vote)	**0.7557**	**0.003102**
Laplace	**0.7644**	**0.002156**
Laplace (I)	**0.7965**	**0.003210**
Item Average	**0.8163**	**0.002973**
User Average	**0.8362**	**0.004029**
Dirichlet	**0.8658**	**0.006308**
Dirichlet (Corrected)	**0.8693**	**0.007463**
Dirichlet (I)	1.059	0.02003

TABLE 6.1: Mean Absolute Error. Lines in bold are statistically significant compared with the line directly below at the 95% level.

problem quite as well. Below the Gaussian techniques are the MBCF techniques, of which the item-based versions perform best. The multinomial techniques are next, slightly better than the simple averages, and finally the Dirichlet methods come last. It is interesting that the Dirichlet methods come last, below the simple averages. This issue will be explored at the end of this chapter.

Table 6.2 shows the RMSE results for each technique. Here the item-based MBCF techniques perform best, followed by the user-based MBCF techniques, then the Gaussian techniques. As with the MAE the multinomial techniques perform worst. It is strange that the RMSE results do not quite match the MAE results. As the RMSE exaggerates the effect of large errors, the Gaussian techniques may have a tendency to produce large errors on a number of ratings, despite providing more accurate results on average.

The F1-score results are shown in Table 6.3. The Gaussian models perform the best as with the MAE, but in contrast to the MAE and RMSE results, the MBCF methods are outperformed by the Dirichlet techniques. It is interesting that the Dirichlet models which produced large errors did so well using the F1 metric, losing statistical accuracy,

Method	RMSE	SE
Item MBCF (Adjusted Cos)	0.9209	0.003309
Item MBCF (PCC)	0.9386	0.002608
Item MBCF (Cos)	0.9416	0.002651
PCC MBCF (Weighted)	0.9467	0.002758
PCC MBCF	0.9551	0.003653
PCC MBCF (Default Vote)	0.9721	0.003470
Gaussian (G)	1.010	0.001968
Gaussian (GG)	1.010	0.001878
Gaussian-Gamma (G)	1.011	0.001264
Gaussian-Gamma (GG)	1.012	0.001263
Gaussian (D)	1.015	0.001547
Gaussian-Gamma (GI)	1.016	0.003776
Gaussian-Gamma (D)	1.016	0.001132
Gaussian-Gamma (GGI)	1.017	0.003762
Gaussian (GI)	1.017	0.003633
Gaussian-Gamma (DI)	1.018	0.003783
Gaussian (GGI)	1.020	0.003513
Item Average	1.023	0.002972
Gaussian (DI)	1.025	0.003851
User Average	1.044	0.005326
Laplace	1.123	0.002823
Laplace (I)	1.155	0.004846
Dirichlet	1.242	0.007122
Dirichlet (Corrected)	1.246	0.008463
Dirichlet (I)	1.468	0.02651

TABLE 6.2: Root Mean Square Error

but still outperforming MBCF in classification accuracy.

Overall the Gaussian models performed the best on the metrics we tried, closely followed by the item-based memory-based collaborative filtering algorithm. The RMSE results indicate that the Gaussian techniques may be more likely to produce extreme errors than the MBCF techniques, though the MAE shows that on average the predictions it produces are more accurate. The Gaussian probabilistic techniques also have higher classification accuracy, as shown by the F1 results.

6.3 Sparsity Results

For the sparsity experiment we looked at a subset of the techniques which performed well in the general experiments. We first looked at a selection of the probabilistic techniques, before comparing the best of these techniques against the MBCF and averages.

Figure 6.4, Figure 6.5, and Figure 6.6 show the sparsity results using MAE, RMSE, and F1 Score respectively for the probabilistic techniques. As sparsity increases there is a

Method	F1 Score	SE
Gaussian-Gamma (D)	0.7585	0.004236
Gaussian-Gamma (G)	0.7584	0.004159
Gaussian (D)	0.7583	0.003607
Gaussian (G)	0.7582	0.003599
Gaussian-Gamma (GG)	0.7580	0.004123
Gaussian (GG)	0.7578	0.003600
Gaussian (GI)	0.7551	0.003614
Gaussian-Gamma (DI)	0.7551	0.003374
Gaussian-Gamma (GI)	0.7550	0.003360
Gaussian-Gamma (GGI)	0.7546	0.003358
Gaussian (GGI)	0.7543	0.003632
Gaussian (DI)	0.7540	0.003607
Laplace	0.7442	0.003662
Dirichlet	0.7315	0.005976
Dirichlet (Corrected)	0.7308	0.005656
Laplace (I)	0.7283	0.003765
PCC MBCF (Default Vote)	0.6084	0.002277
Dirichlet (I)	0.5979	0.01406
PCC MBCF (Weighted)	0.5960	0.002912
Item MBCF (Adjusted Cos)	0.5585	0.006217
PCC MBCF	0.5448	0.005674
Item MBCF (PCC)	0.5329	0.004998
Item MBCF (Cos)	0.5190	0.006201
Item Average	0.3896	0.004065
User Average	0.3084	0.01633

Table 6.3: F1 Score

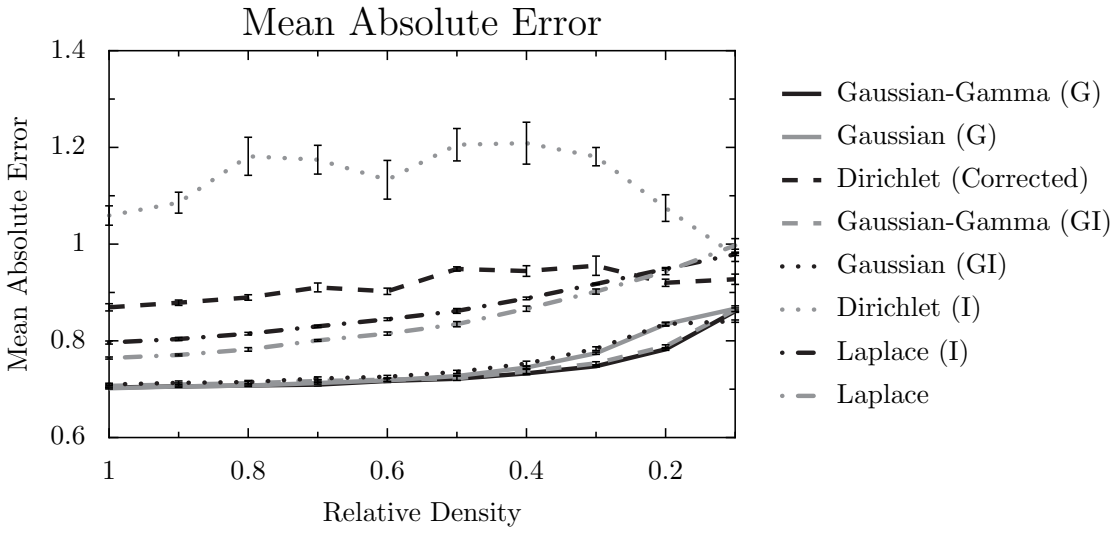


Figure 6.4: MAE under conditions of varying sparsity for probabilistic techniques.

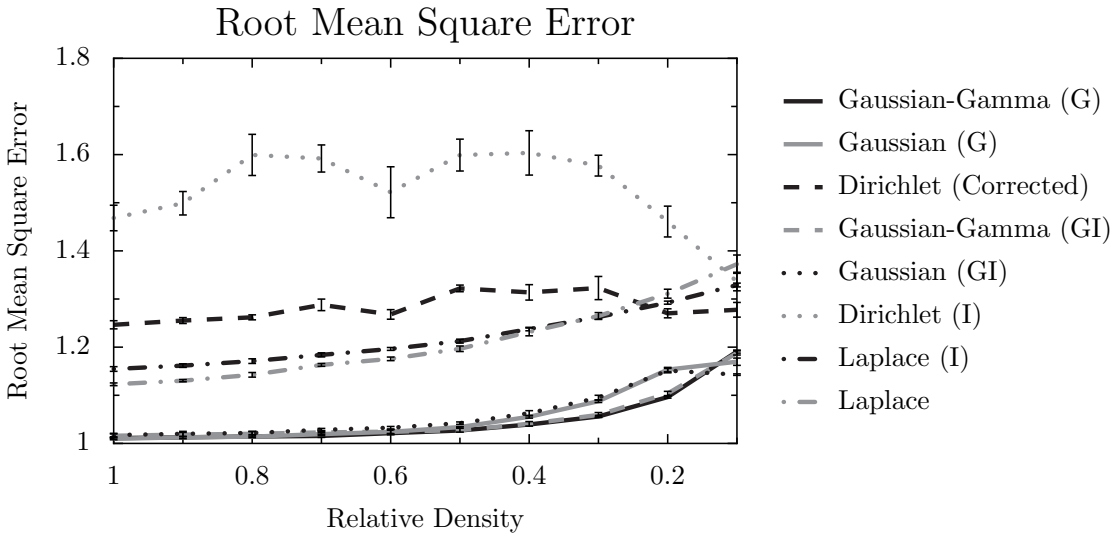


FIGURE 6.5: RMSE under conditions of varying sparsity for probabilistic techniques.

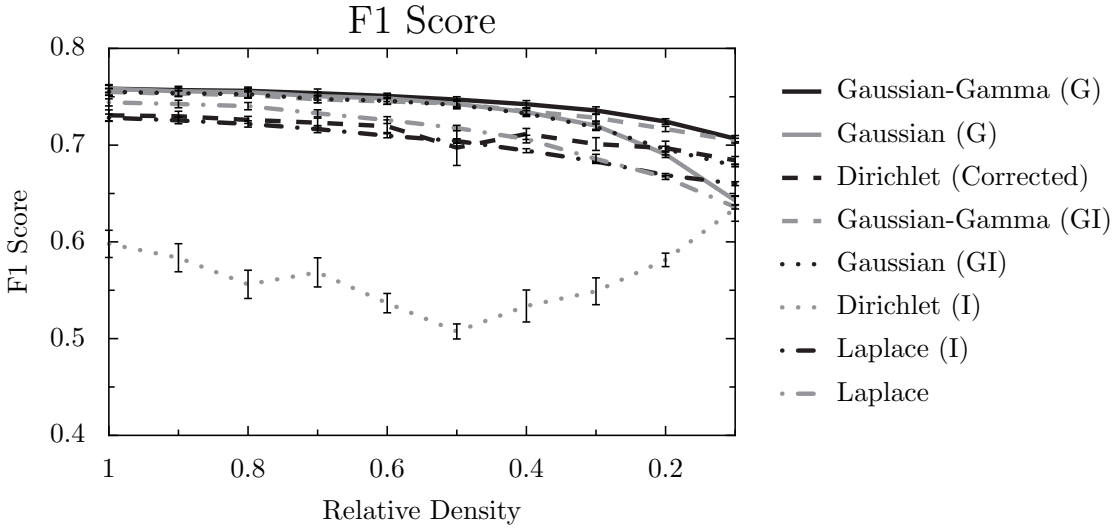


FIGURE 6.6: F1 score under conditions of varying sparsity for probabilistic techniques.

slow increase in MAE, but the relative performance of each technique stays largely the same as in the general experiments. The only anomaly is that the error of the Dirichlet techniques actually decreases as sparsity increases. This may be because as the there is less information the system may only have the users prior rating probabilities to make a recommendation. This is similar to the user average method, which performed better than the Dirichlet technique in the general experiment.

The picture is almost the same for RMSE, except at conditions of very high sparsity the Gaussian techniques without a Bayesian prior have slightly lower error than those that do. The difference is not statistically significant.

The F1 scores show a slow decrease in performance as sparsity increases, with the techniques with Bayesian priors showing a slightly slower decline in performance. It is interesting that the Dirichlet technique performs almost as well as the Laplace tech-

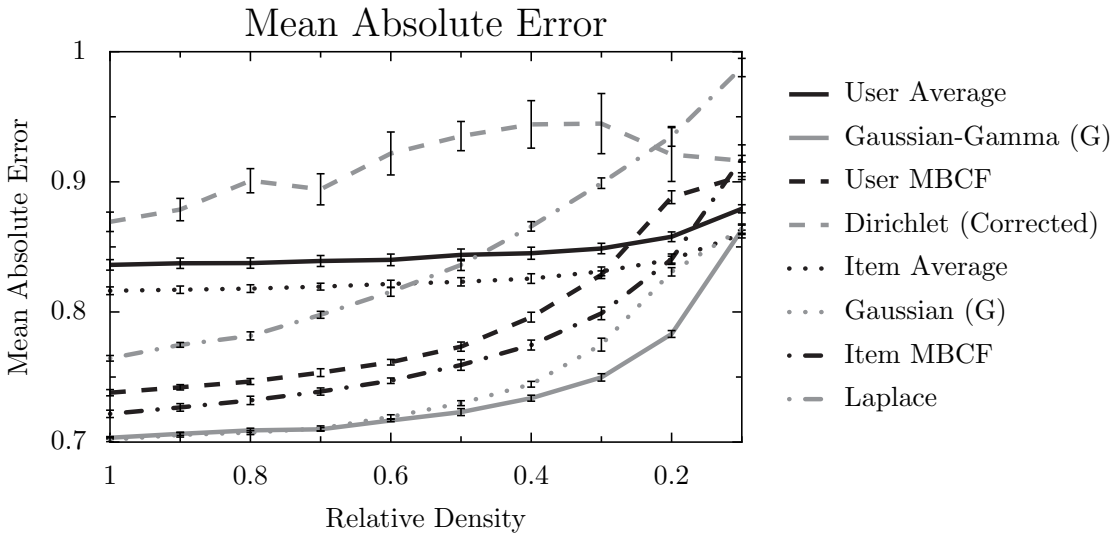


FIGURE 6.7: MAE under conditions of varying sparsity for selected techniques.

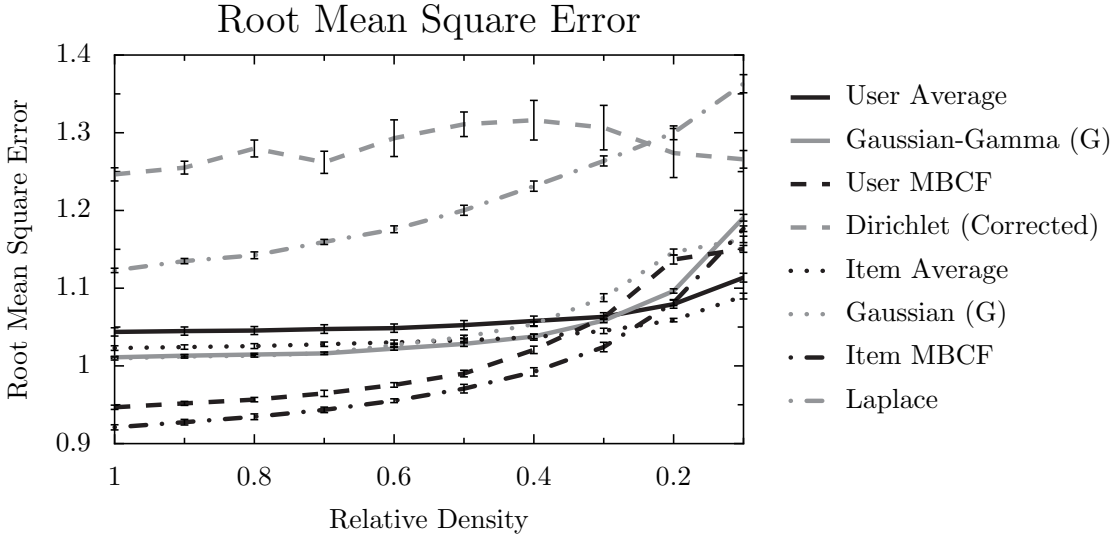


FIGURE 6.8: RMSE under conditions of varying sparsity for selected techniques.

nique as sparsity increases. The performance of the item-based Dirichlet technique is rather erratic and it isn't clear why this should be the case.

Out of these techniques we chose to keep the Gaussian-Gamma, Gaussian, corrected Dirichlet, and Laplace recommenders to compare with the MBCF and average recommenders. Figure 6.7, Figure 6.8, and Figure 6.9 show the sparsity results using MAE, RMSE, and F1 Score respectively.

As sparsity increases, the MAE increases fairly slowly until half of the original ratings remain. As sparsity increases to one fifth, the difference in performance is pronounced, and this difference accelerates as the data is reduced to a tenth of the original ratings. The multinomial recommender with Laplace smoothing is hit hardest by the increase in sparsity. Although the Dirichlet results are worse, they are more erratic, and finish with

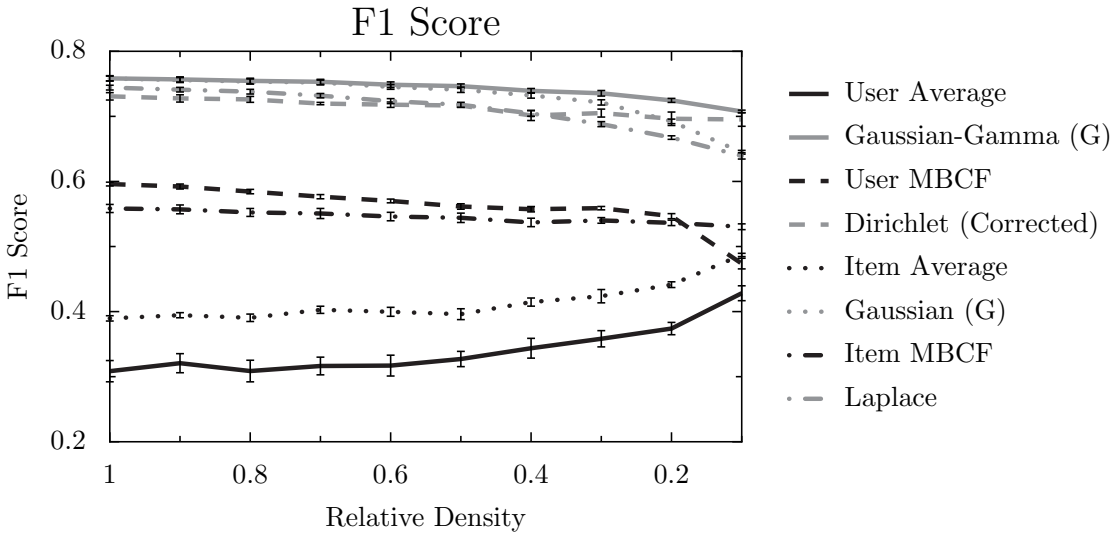


FIGURE 6.9: F1 score under conditions of varying sparsity for selected techniques.

lower error at the highest level of sparsity. The average rating techniques cope fairly well with increased sparsity, as does the item-based MBCF technique. Under conditions of moderate sparsity the Gaussian-Gamma technique performs better than the Gaussian technique, but there is no significant difference at a relative dataset density of 0.1.

As with the general experiments the MBCF techniques perform better than the probabilistic ones using RMSE. As sparsity increases this stays the same. Looking at the F1 score there is a large gap between the probabilistic techniques and the other techniques. In this case the Gaussian-Gamma technique retains excellent levels of performance even when using the most sparse dataset. It is also interesting that increasing data density actually degrades the performance of the average methods using the F1 score. Users tend to rate items they like, so removing ratings may remove some of the lower ratings for items and make predictions more accurate.

6.4 Analysis

In this chapter I have shown that Bayesian recommender systems are capable of producing results which are significantly better, at a 95% level using a paired t-test, than those gained from using memory-based collaborative filtering. In particular I find that the Gaussian model produces the best results across most of the tests.

In the case of the multinomial model, the addition of a conjugate prior is found to be harmful to the model, at least where some of our metrics are concerned. The MAE is significantly increased, although it performs better on some of our decision support metrics, such as the F1-score. I believe this is because the model is not a good fit to the problem despite initial appearances. It suffers from a surfeit of parameters, and a

paucity of data with which to learn them. Finally, it makes independence assumptions on top of those imposed by a naïve Bayes model.

The Gaussian model performs well using simple maximum likelihood estimates, outperforming the other models tested on MAE, and on most of the decision support metrics. The addition of prior knowledge to the system in the form of a conjugate prior did not significantly affect the results obtained. The results would seem to suggest that for co-rating behaviour the Gaussian distribution is a good model.

It is interesting that the technique improved results when the data set was made more sparse. The problem with applying prior knowledge to recommender systems is that in many cases there is little or no information. In order to deal with these cases correctly we have to use these to train the priors. In our Gaussian-Gamma model this means we assume less prior knowledge, which helps with low information cases, but doesn't do much for the higher information cases. The reverse is also true when excluding lower information cases. As the Gaussian model has an inbuilt measure of uncertainty, this could explain why adding prior knowledge of uncertainty does not help much, and also why it outperforms the Dirichlet model.

Overall our experiments have shown that a Gaussian model for our prior probabilities with a Gaussian-Gamma model for our likelihoods works best. It is important to incorporate a measure of uncertainty in probabilistic models, so that results are not skewed by overconfident estimates from limited data. The use of a principled probabilistic framework allowed us quickly move on from the relative failure of the Dirichlet model in a logical way by analysis our results, rather than through haphazard experimentation.

In this chapter I have concentrated on comparing the predictive performance of the probabilistic model with conventional collaborative filtering. However Probabilistic models provide more information which may be of value in some applications. For example, it provides a probability associated with each rank which can be used to predict the expected error, and may be used to minimise some loss function, thus providing more accurate decisions. The Bayesian framework may even be used for selecting between different possible models.

6.5 Conclusion

In the first half of this thesis I looked at producing collaborative filtering recommendations in a principled way. This gave statistically significant improvements on some metrics, but decreased performance on others, compared to simple ad-hoc memory-based collaborative filtering techniques. In cases where the improvement was statistically significant it may not be noticeable to a user of the system or lead to an increase in utility. It may not be possible, or worthwhile, to reduce error far beyond this point.

The first model we tried performed poorly even when using complex correction factors to deal with uncertainty. The second technique was much simpler but produced far superior results even without using a Bayesian prior. The simple techniques based on average ratings produced reasonable results even at high levels of sparsity, and may be good enough for many applications. The most important thing is to pick the correct model. While these techniques did not perform spectacularly, they make it relatively easy to add additional information, such as context, to the model. The improvements made in this chapter will help recommenders produce better results in mobile and distributed enviroments where data is sparse.

Chapter 7

User Profiling

The final incarnation of the IK system uses a client-server architecture. The server handles queries using a text-based information retrieval system, while the clients, running on users' devices, produce profiles of expertise and generate automatic and manual queries. The system was intended to generate profiles automatically to minimise user effort, which raises the issue that information not related to a user's professional interests will be incorporated into their profile.

In this chapter I explore the problem of privacy-preserving profiling within the context of the IK system. In Section 7.1 the final IK system is described in detail, before briefly looking at profile generation in Section 7.2. In Section 7.3 privacy-preserving profiling is defined, and related work is summarised in Section 7.4. Finally, a simple information retrieval system is described in Section 7.5 before looking at how we may evaluate privacy-preserving algorithms in Section 7.6.

7.1 The IK System

An overview of the architecture of the IK system is shown in Figure 7.1, with the system being composed of a single central server, and many different clients. In this version of the system we assume that each user and their corresponding profile is associated with only one client. The use of multiple devices would require a way to combine several sub-profiles into a single profile on the server, which is not explored in this thesis.

The server is supplied with a corpus of documents which are analysed to generate a document model. The parameters of this model are transmitted to the clients in order to convert profiles into the appropriate format, and to train classifiers for document filtering. The algorithms used will be discussed in more detail later in this thesis.

The server is also responsible for handling queries for expert recommendations, which are executed against a store of profiles received from expert's devices. Profile recom-

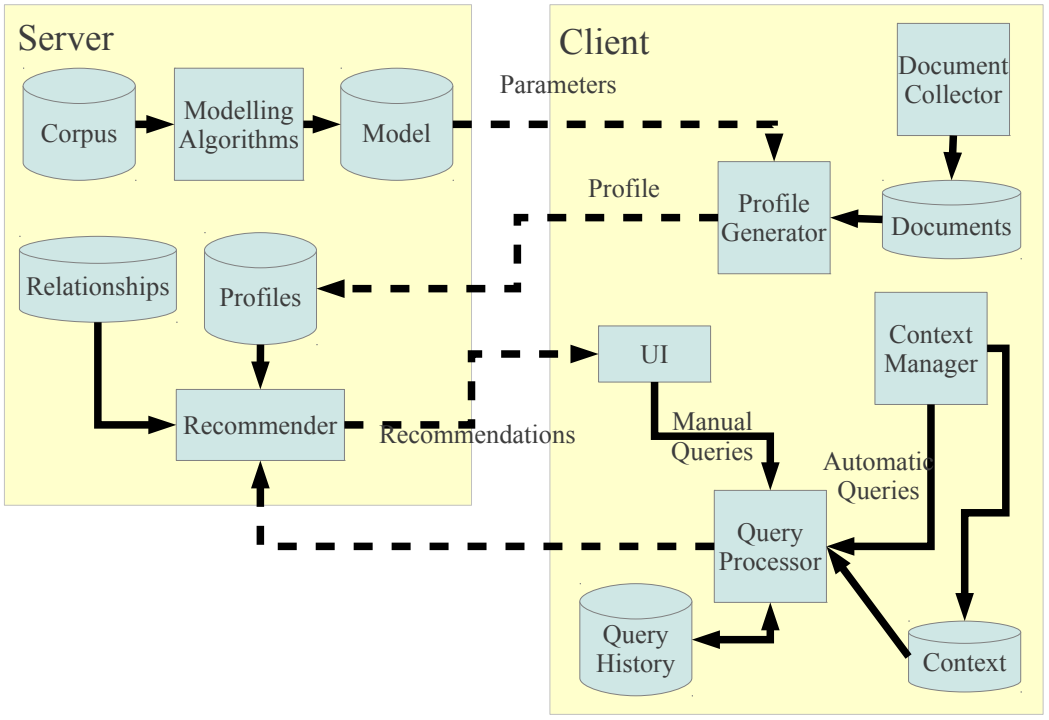


FIGURE 7.1: Architecture of IK System

mendations are generated through a text-based information retrieval system described later in this thesis. The IK system also modifies the raw recommendations by weighting experts who have a "stronger" relationship to the user more highly. The strength of a relationship could be based on the user's distance in a network of co-authorship, or the level of communication between authors (Banford et al., 2010a,b).

The client is responsible for generating profiles and queries. The first step in profile generation is the collection of documents. Documents are collected automatically from a user's device from a variety of sources, which may include: academic publications, technical reports, draft documents, emails, social networking updates, instant messages, web browsing history, and bookmarks. In this thesis the focus will be on documents such as academic publications, which will be assumed to have been found automatically on a user's device. I will not explore the details of obtaining documents to form a profile. Once documents have been collected they are processed into a suitable format, using the model parameters provided by the server, and transmitted to the server to be stored.

The second job of the client is to generate queries, which is done both automatically and manually. Automatic queries are generated based on task context, e.g. what a user is typing on their device, what applications are running, and environmental context such as the time of day or day of the week. Context is provided by a context manager, and stored on the device. Context is also combined with query history, and used to augment both manual and automatic queries before sending them to the server.

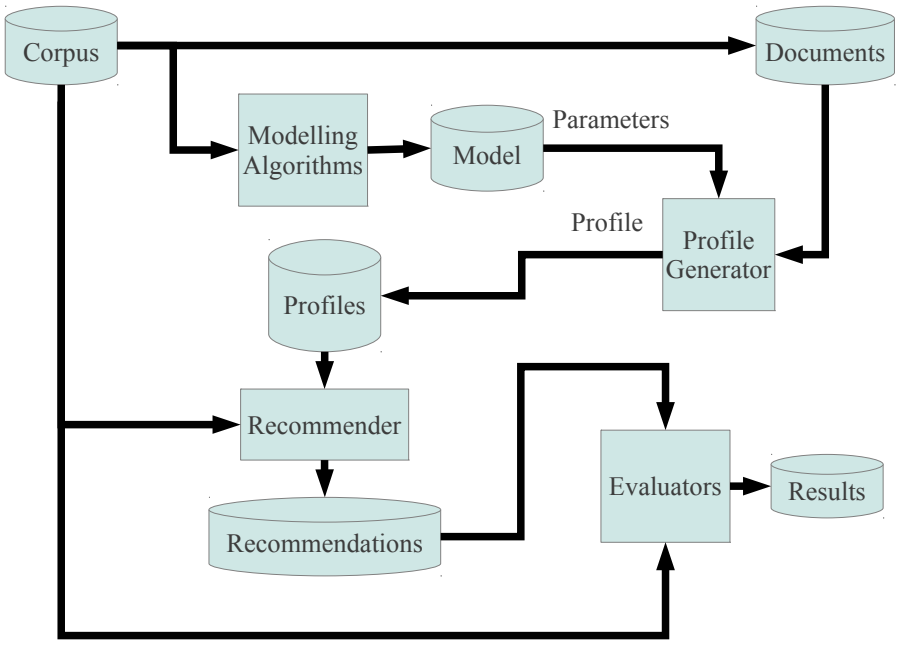


FIGURE 7.2: Cut-down IK System

While a full demonstrator of the IK system detailed in Figure 7.1 was successfully implemented, a cut-down version of this system was built for experimentation and evaluation. The portions of the IK system implemented in this version are shown in Figure 7.2.

The system was intended to run on Nokia N810 internet tablets running a Linux-based operating system, intended to be representative of the capabilities of average smart phones, but more open and easier to develop for. These devices have rather modest capabilities: a 400 MHz CPU, 128MB RAM and 2GB of storage space. The use of these devices had some influence on the choice of algorithms used; most work has to be done on the server. In the time since this device was chosen, smart phone technology has advanced considerably. As an example the latest iPhone contains an 800 MHz processor, 512MB RAM, 16-64GB of storage, and a dual-core GPU. If this system was implemented on modern devices more computationally demanding algorithms could be used.

7.2 Profile Generation

One goal of the IK system is to produce profiles and generate queries with a minimum of effort from the user. Outside of initial training and configuring of the system, and occasional correction of misclassified documents, the system should automatically find and process documents to create a profile.

The alternative would be for users to manually create profiles, either through entering free text, selecting from a limited list of tags, or manually adding documents to the system. Entering free text may lead to poor profiles, as experts may not pick the best terms to describe their interests. Selecting from a limited list of terms relies on the list being updated frequently enough to cover new topics of interest.

An alternative would be to incorporate only documents manually added to a self-archiving repository such as an ePrints[1] archive, or arXiv.org[2]. While this would ensure that private documents were not added, publications may be infrequent, and not every line of investigation will lead to formal publication.

In the case that profile generation requires any significant exertion on the part of the users, there is a high likelihood of the users losing interest in the task, leading to poorly maintained profiles of little utility. By collecting documents automatically we hope to generate profiles which are more up-to-date, have greater depth, and give a better picture of an expert's interests.

This approach does, however, present some challenges; some of the information collected may be irrelevant or private. In the case of irrelevant data, recommendation performance may be reduced, in the case of private information disclosure may have serious negative consequences. In the following sections I will look more closely at the issues involved with automatically generating profiles while maintaining user privacy.

7.3 Privacy-Preserving Profiling

Privacy in data mining has a number of different meanings, from anonymity to uncertainty in the values of attributes. As the function of the IK system is to recommend users, anonymity is not an option. Researchers at RHUL worked on a different notion of privacy in an IK system, requiring permission to share each piece of information and coordinating collaboration using pseudonyms (Yau and Tomlinson, 2011). Here I consider a different approach using machine learning techniques to automatically filter information, and assume that any unfiltered information is shared freely within an organisation.

While the contents of profiles in the IK system are considered to be hidden, they are indirectly visible through the results returned from queries. A query relating to a controversial topic may expose users who are interested in it, and it may be possible to reconstruct a user's profile by making a series of carefully constructed queries. By repeating this process over time it may be possible to infer the contents of individual documents, or a user's activities. In this chapter I ignore security issues, and assume

[1] `http://eprints.org`
[2] `http://arxiv.org`

that third parties cannot peek at the profile, for example by observing it in transit to the server.

We must ensure that private information is removed from profiles, without affecting the public information. These goals are in opposition with each other: as we remove private information we risk removing useful information which will reduce performance; as more public information is retained we risk including private information.

There are three main types of private information we want to remove from a profile:

Private Terms Individual words such as usernames, passwords, banking details, names, or email addresses.

Private Interests Private interests are larger pieces of information which are clearly not professional interests, but constitute a private interest or hobby. These may be irrelevant, embarrassing, or incriminating.

Out-of-Context Interests Out-of-context interests are interests which may form a legitimate part of a users' profile, but appear out-of-context in a particular user's profile.

Ideally private information should be removed from profiles without user input, but this is a difficult task as each user will consider different topics to be private. While some subjects may be private for most users, such as sexual preferences, these topics may constitute an integral part of certain users professional interests. Public documents may be vastly outnumbered by private documents on a user's device making automatic determination of a user's professional interests difficult.

In this thesis I present a framework for evaluating the performance of privacy-preserving profiling algorithms. Our first attempts at privacy-preserving profiling used term filtering and corpus projection to remove private information without user input. These techniques failed to remove private information adequately and so we moved on to look at techniques which filter whole documents.

In this thesis I look at two document filtering techniques, the first is based on filtering all profiles based on a single global model, and the second uses a classifier per-user to filter documents. We will see later in this thesis that each of these approaches has certain advantages and disadvantages, and there is a trade-off between the power of a single global classifier trained on large amounts of data, and the flexibility of multiple classifiers trained using less information.

7.4 Related Work

While privacy in data mining is an important and active area of research, there has been little work done on the problems described earlier in this chapter. Profiles are usually treated as objects which are either wholly private or public, where in our scenario public and private data is not trivially separable, and profiles are linked to named individuals through necessity.

In Reichling and Wulf (2009) the authors describe an expert recommender system which is quite similar to the IK system, though more limited in scope. Their "ExpertFinding" system is designed to help find experts to deal with customer requests using profiles of expert interests coupled with a simple keyword-based search engine. User profiles are built using a combination of processed user documents, and keywords manually selected by the user. These keyword profiles are finally processed using *Latent Semantic Analysis*. The documents used to generate a profile are picked from folders manually selected by users, and stored centrally. While our system is quite similar to the one described in their paper, in that our goals both include partially-automatic generation of profiles, and we use similar information retrieval techniques, they make little attempt to deal with privacy, and instead only use documents which are manually added to the profile.

Privacy preserving data mining (PPDM) is a growing area of research which aims to ensure that data mining activities can be conducted while safeguarding user privacy (Verykios et al., 2004). PPDM encompasses data modification techniques which seek to remove explicit and implicit private information in a way that resists discovery through data mining techniques. Verykios et al. (2004) describes three broad classes of PPDM algorithms: heuristic, cryptographic, and reconstruction-based.

Heuristic PPDM algorithms work by blocking or filtering values in an attempt to sanitise records as in Sweeney (2002). Cryptographic PPDM algorithms deal with the *Secure Multiparty Computation* (SMC) problem to perform data mining without giving away information. Reconstruction-based techniques use data perturbation to modify records, and look at distributions of information rather than individual records to perform data mining (Agrawal and Aggarwal, 2001).

Evaluating PPDM algorithms can be done in a number of ways. In Agrawal and Aggarwal (2001) the authors use information theory to quantify privacy-preservation and information-loss using differential entropy. While these metrics are mathematically rigorous they are difficult to apply to real PPDM systems. In Bertino et al. (2005), the authors develop a broad framework for evaluating privacy-preserving data mining algorithms.

They state that the main goals of PPDM algorithm should be: prevention of private information discovery, resistance to multiple data mining techniques, scalability, and

preserving data mining utility. The experiments described in this thesis build upon their evaluation dimensions, adapting them to privacy-preserving profiling.

In Schneider (2005) the author applies semi-supervised text classification techniques to the problem of filtering junk email. Spam emails are considered positive, and legitimate emails are considered negative. Under the assumption that there is a supply of junk emails for training, as well as larger collection of "mixed" messages for training supplied by the incoming stream of emails, the author uses a variant of naïve Bayes which uses only positive and unlabelled examples. The motivation is that this technique requires little or no user input. I use an improved version of this technique in my experiments, although rather than training a single classifier using a large amount of information I train multiple classifiers with little information.

7.5 A Simple Information Retrieval System

As the focus of this work is automatic privacy-preserving profiling, and not information retrieval techniques, I make use of a simple information retrieval system. While more complex techniques may produce better results, they may also make it more difficult to analyse the effects of filtering algorithms on profiles. Some of the material in this section was covered in Section 2.2 but is restated here for the sake of clarity.

The system as well as the algorithms that run upon it are implemented using the scripting language Python[3], with most of the work done using the numerical libraries Numpy[4] and Scipy[5]. Some text processing is done with the *Natural Language Toolkit* (NLTK)[6], and some with the Gensim vector space modelling package[7]. The system runs largely in a single process, with some simple parallelization to speed running experiments.

7.5.1 Text Processing

I will quickly recap the subject of text processing. The first step in generating profiles is to convert each user's documents into a standard form. Documents are reduced to pure text, converted to lower-case ASCII, and turned into a list of tokens. Common words with little discriminative power, called stop words, are removed using a list provided by Fox (1989). Each word in the list of tokens is reduced to its root form using the Porter stemming algorithm; For example, "computer" and "computation" may be reduced to the stem "comput". Finally these words are counted to produce a term-frequency representation of the original document. While this bag-of-words representation removes

[3] `http://python.org`
[4] `http://numpy.org`
[5] `http://scipy.org`
[6] `http://nltk.org`
[7] `http://radirehurek.com/gensim/`

some information from the documents, such as the context of words, and may result in reduced performance, it should also remove some private information.

Profiles are produced by summing term frequency representations of their constituent documents, normalised by their length. This prevents the profile from being dominated by larger documents. The equation is given below,

$$\mathrm{TW}_{w,p} = \sum_{j \in D_p} \frac{n_{w,j}}{\sum\limits_{w' \in W} n_{w',j}} \tag{7.1}$$

where $\mathrm{TW}_{w,p}$ is the weight of term w in profile p, D_p is the set of documents that profile p contains, $n_{w,j}$ is the number of times term w occurs in document j, and W is the set of terms.

Each profile is represented as a multidimensional vector in a vector-space model (Salton et al., 1975), where each dimension corresponds to the weight of a particular term in the profile. Term frequency-inverse document frequency (TF-IDF) weighting is applied to each vector, which normalises the term frequency weights by the profile length, and multiplied by the inverse document frequency (IDF), giving a higher importance to terms which occur in fewer documents (Manning et al., 2008). The TF-IDF weighting equations are given below,

$$TF_{w,p} = \frac{TW_{w,p}}{\sum\limits_{w' \in W} TW_{w,p}} \tag{7.2}$$

$$IDF_w = \log \frac{|D|}{|\{d : t_i \in d\}|}, \tag{7.3}$$

$$TFIDF_{w,p} = TF_{w,p} \times IDF_w, \tag{7.4}$$

where $TF_{w,p}$ is the term frequency weighting of term w in profile p, IDF_w is the inverse document frequency of term w, D is the collection of documents, and $TFIDF_{w,p}$ is the TF-IDF weighting of term w in profile p.

Applying this process to a collection of documents produces a matrix which contains a row for every term which appears in the document collection. As each document will only contain a small fraction of all terms which appear in a large set of documents, this matrix will be very sparse. Differences in users' vocabularies means that documents which refer to similar concepts may have few terms in common. High-dimensional vectors and matrices require more computational resources to manipulate.

7.5.2 Latent Semantic Analysis

Using TF-IDF weights for comparing documents in a vector-space model has several drawbacks. The first is that different words can describe the same concept, i.e. words can have many synonyms. If different authors use different words to describe the same

concepts, their documents may be orthogonal in vector-space even though they are obviously similar. The second problem is the so-called "curse of dimensionality"; typically document collections contain many thousands of words, and possibly millions of documents. Dealing with such large and sparse document matrices may be time-consuming and expensive at best, and at worst, intractable.

Latent Semantic Analysis (LSA) or *Latent Semantic Indexing* (LSI) is a data processing technique that attempts to uncover hidden meaning in term-document matrices. LSA works by projecting documents into a lower-dimensional concept-space where documents which are similar are represented by vectors which point in a similar direction. This solves the problem with synonymy, dimensionality, and serves to remove noise from the dataset. LSA is implemented using the *Singular Value Decomposition* (SVD), which is described in detail in the next section.

7.5.2.1 Singular Value Decomposition

The singular value decomposition can be used to factorise any matrix, square or rectangular, into the product of three matrices,

$$A = U\Sigma V^T, \tag{7.5}$$

where A is an m-by-n matrix composed of n document vectors each with m terms, U is an m-by-m orthogonal matrix, V is an n-by-n orthogonal matrix, and Σ is an m-by-n diagonal matrix.

The decomposition is found by considering the eigenvectors and eigenvalues of the symmetric matrices A^TA and AA^T which are

$$\begin{aligned} A^TA &= (U\Sigma V^T)^T U\Sigma V^T \\ &= V\Sigma^T U^T U\Sigma V^T \\ &= V\Sigma^2 V^T, \end{aligned} \tag{7.6}$$

$$\begin{aligned} AA^T &= U\Sigma V^T (U\Sigma V^T)^T \\ &= U\Sigma V^T V\Sigma^T U^T \\ &= U\Sigma^2 U^T. \end{aligned} \tag{7.7}$$

Square matrices may be factorised as $A = Q\Lambda Q^T$, where Q is a matrix of orthogonal eigenvectors, and Λ is a diagonal matrix containing the corresponding eigenvalues. These correspond to the definitions of A^TA and AA^T given above, where U are the eigenvectors of AA^T, V are the eigenvectors of A^TA, and Σ contains the square root of the eigenvalues

of both A^TA and AA^T. If we're not using the documents used to build the SVD, we can save time by only calculating the eigensystem of AA^T, that is U and Σ^2.

To project documents into concept space we use

$$\hat{v} = \Sigma^{-1}U^T v, \tag{7.8}$$

or to project them back,

$$v = U\Sigma\hat{v}^T, \tag{7.9}$$

where v is a document column vector.

Similarly, new terms can be added to the decomposition using

$$\hat{u} = uV\Sigma^{-1}, \tag{7.10}$$

for a term vector u.

To compare pairs of projected document vectors we use

$$\begin{aligned} \mathrm{Sim}(\hat{v}_a, \hat{v}_b) &= (\Sigma\hat{v}_a)^T(\Sigma\hat{v}_b) \\ &= \hat{v}_a^T\Sigma^2\hat{v}_b, \end{aligned} \tag{7.11}$$

which gives the cosine similarity between two projected document vectors $\hat{v}_a$ and $\hat{v}_b$. To compare matrices of documents we use

$$M = V^T\Sigma^2 V, \tag{7.12}$$

which produces a symmetric matrix of cosine similarities between each document in the matrix of projected document vectors V. Note that the projected document vectors should be normalised first. If we define document vectors to be $\hat{v} = U^T v$ we can avoid multiplying by Σ^2 and instead use,

$$\mathrm{Sim}(\hat{v}_a, \hat{v}_b) = \hat{v}_a^T\hat{v}_b, \tag{7.13}$$

and

$$M = V^T V, \tag{7.14}$$

as the factors from the singular values Σ cancel out.

Profiles are formed from concatenated documents normalised by length. They are updated simply by adding more tf-idf weightings to the profile, $P_{\text{updated}} = P_{\text{old}} + P_{\text{new}}$, where P is a profile. It does not matter in which order documents are projected and added together to form a profile because of the distributive multiplication law for ma-

trices,

$$\begin{aligned}\hat{P}_{\text{updated}} &= P^T_{\text{updated}} U\Sigma^{-1}, \\ &= (P_{\text{old}} + P_{\text{new}})^T U\Sigma^{-1}, \\ &= P^T_{\text{old}} U\Sigma^{-1} + P^T_{\text{new}} U\Sigma^{-1}.\end{aligned}$$

7.5.2.2 Dimensionality Reduction

Singular value decomposition reduces the original document vectors into n-dimensional vectors, however some of these dimensions will contain little information and can be safely removed. The low-rank matrix approximation is given by $A_k \approx U_k \Sigma_k V_k^T$, and the properties of the SVD guarantee that each rank-k approximation has the minimum error of any rank-k approximation.

The low-rank matrix approximation is produced from a singular value decomposition, where the rows of U, entries of Σ, and columns of V^T are put in descending order of the size of the singular values. Then an approximation can be formed by retaining the top-k rows of U, columns of V^T and the k-by-k top-left portion of Σ.

The rank of a matrix is determined by the number of linearly independent columns it contains, corresponding to document vectors in this case, hence the closest approximation possible has rank $|D|$ at most, where D is the set of document vectors contained in the term-document matrix.

7.5.2.3 Interpretation

Latent semantic analysis through singular value decomposition is closely related to another technique called *Principal Component Analysis* (PCA). PCA also makes use of SVD, but rather than working with a document or feature matrix directly a covariance matrix is analysed. The singular values and eigenvectors then represent principal components, or the directions of maximal variance within the dataset. The computation of a covariance matrix involves the creation of a dense n-by-n matrix, where n is the number of terms and features making it intractable in this case.

In LSA the eigenvectors do not correspond directly to principal components, but nonetheless represent important components of the information contained within the data. The larger eigenvalues and eigenvectors in the decomposition represent larger components of the data, and are sometimes said to have more "energy". Alternatively the eigenvectors can be seen as a transformation of the data into a high-dimensional hyperellipsoid. Here each eigenvector represents an axis of the hyperellipsoid, with the eigenvalues representing the length along that axis.

7.5.2.4 Implementation

Computing a singular value decomposition is really the problem of computing the eigenvectors and eigenvalues of AA^T. We make use of an LSI implementation called Gensim which is a Python package. This implementation uses a stochastic singular value decomposition to efficiently compute the SVD of blocks of documents which are subsequently merged into a single model. This makes it better suited for working on large document collections, although it may not be quite as accurate as deterministic algorithms.

In order for the clients to project profiles and documents into concept-space the client requires the matrices U and Σ, the inverse document frequency scores, and a map between words and ids. If we estimate that the public corpus contains around 20000 words, using 64-bit precision for the IDF gives us a vector size of under 160 kilobytes. The size of U and Σ is dependent on the rank of these matrices and the number of terms, using $k = 100$ with 64-bit (double) precision gives a combined size of around 15 megabytes. More accurate results could be obtained by retaining more dimensions, but the amount of data transferred, and the time to run calculations would be increased. These figures do not account for overheads or compression.

These model sizes are fairly modest, and if one considers a system being used by many thousands of users, this model should change only very slowly, and it should be acceptable to only update the model every few weeks or months. It is likely the initial transfer of data will be performed on a fast wireless or local area network within an organisation, and so this may not be too much of a problem. Additionally updates could be sent as the difference from the current model.

7.5.3 Shortcomings of the System

The techniques I have used to implement this version of the IK system are fairly simple and have some limitations. The bag-of-words representation of documents removes quite a lot of information such as document structure including sentences and paragraphs, and the order and context in which words appear. An alternative would be to use n-grams: combinations of words appearing next to each other in a document. This would help preserve the context of each word, and may help with polysemy, but would vastly increase the dimensionality of the model and its sparseness.

Latent semantic analysis has been criticised for not dealing with polysemy, as each meaning of a word will be mapped to the same place in concept-space. LSA also uses the Frobenius norm to find an optimal decomposition, which makes an assumption of Gaussian noise which may not hold (Hofmann, 2001). Probabilistic Latent Semantic Analysis(Hofmann, 1999), and its successor Latent Dirichlet Allocation(Blei et al., 2001), are two related techniques which attempt to deal with these shortcomings. Our system uses LSA because it is relatively fast, and relatively simple.

7.6 Evaluating Privacy-Preservation

The two main questions we wish to answer by evaluating the privacy-preserving techniques described in this chapter: how well do these techniques preserve performance compared to "clean" and "contaminated" profiles; and how well do these techniques preserve privacy?

Bertino et al. (2005) describe five criteria with which to evaluate PPDM algorithms:

- Efficiency
- Scalability
- Data Quality
- Hiding Failure
- Privacy Level

Of these criteria the most applicable to our problem are data quality and hiding failure. Data quality describes the effect that the privacy preserving process has on the original data. They suggest that this can be tested by the change in data mining performance when using the processed data versus the original dataset. Hiding failure relates to the amount of private data that can be recovered from the sanitised data.

To answer these questions I have devised a series of experiments, using a source of public profiles, and a source of other documents which are added to the profiles to act as private documents. Private documents are added randomly to each profile in a given proportion relative to the number of public documents that profile already contains. Each private document may appear in multiple profiles, but will appear in a particular profile at most once. Each filtering technique is then applied to the contaminated profiles and the results compared with the unfiltered contaminated and uncontaminated results.

Each experiment is also repeated with a set of documents which are similar to those included in the public dataset. Each profile in the system belongs to one of two categories, and are contaminated with documents belonging to the other category. In this way we test the ability of each privacy-preserving technique to remove out-of-context information.

7.6.1 Performance

To test the performance impact of the techniques I tested the ability of the system to return relevant profiles when they are contaminated with varying amounts of private documents. A portion of public documents are withheld as queries for the information

retrieval system, and the results are evaluated using the relevance of each profile that is returned.

In this chapter I consider two different situations, the first considers users to be relevant to a query if they are listed as an author of document used to generate that query. This tests the ability of the system to return the exact authors of a document, which can lead to low performance scores, as many documents have a small amount of authors, but other experts working in the same field may be relevant to a query.

The second version considers users to be relevant to a query if they belong to the same group as the authors of the paper. This tests the ability of the system to classify users. Not all users in a group will be similar to every document written by people in that group, which may lead to inflated performance scores. By taking these experiments together we hope to build a better picture of the effect of contamination and filtering on performance, in the absence of true relevance information.

7.6.2 Privacy Preservation

To test how well each filtering technique preserves privacy we consider a malicious user using the query interface to search for profiles which are similar to a given topic. For instance, the attacker may be attempting to find medical researchers who are secretly involved in vivisection.

A number of profiles are contaminated with private documents, and the system is queried using a set of withheld private documents. For each query the set of relevant profiles is the set of contaminated profiles. Techniques which produce higher scores on this task are worse at preserving privacy.

When looking at the out-of-context dataset, we contaminate two sets of profiles, and use two sets of queries with the corresponding relevant profiles. The experiments are constructed so that a proportion of profiles are relevant to a query, so that when using the out-of-context dataset with a proportion of half of all profiles, all of the profiles in a given group are contaminated.

7.6.3 Classification

I tested the document-filtering techniques by looking at their classification performance. Public and private documents were split into K-folds and used for training and testing. The global classifier was trained and tested using approximately equal proportions of public and private documents. The per-user filter was trained on profiles contaminated using varying amounts of private documents, but tested on approximately equal numbers

of public and private documents, with the public documents being taken from the users' profiles.

This experiment shows how well the document-filtering techniques work outside of the information retrieval system. We expect these results to be similar to those in the performance and privacy-preservation experiments, but by studying these results independently from the information retrieval system, we should gain greater insight into their performance.

7.6.4 Datasets

In order to carry out experiments in user profiling we require both corpora containing documents representative of user profiles, and private information with which to contaminate them. These are also used to train models used to filter private information. We did not have the time and resources to obtain real user profiles contaminated with real private data. Instead we created profiles using academic publications and documents representing private information from different sources.

7.6.4.1 Initial Datasets

In my work for our first paper on privacy-preserving profiling (Barnard and Prügel-Bennett, 2011b), the corpus of public documents was obtained from a semantic web service[8] providing documents automatically harvested from a collection of ePrints publications archives (Glaser et al., 2009). The RKBExplorer website which is part of the ReSIST project at the University of Southampton provides a semantic web database containing information from a number of institutions where authors of academic papers have self-archived their publications in ePrints repositories. This dataset has information on authors and their publications, including titles and abstracts, but unfortunately not full document texts.

To create a profile dataset from the ePrints data, I downloaded all of the RDF files from the RKBExplorer website. These files were used to populate a 4Store[9] RDF database. Abstracts, titles, and authorship information were extracted from this dataset using SPARQL queries, and saved into an SQLite database for convenience.

I sampled this database to create a dataset with around 750 profiles and a total of around 14,000 documents. I decided to create a dataset of private documents from another source; a collection of text files obtained from BBS (Bulletin Board Systems)[10], grouped broadly by topic. Amongst these groups were collections of files categorised as

[8] `http://www.rkbexplorer.com`
[9] `http://www.4store.org`
[10] `http://www.textfiles.com`

"Anarchy" and "Drugs". I processed these documents in the same way as the profile data to create datasets with around 1500 and 500 documents respectively.

As this dataset was generated semi-automatically it suffered from a number of problems, such as papers with no abstracts, duplicated authors, abstracts containing markup, abstracts with few words, and wrongly attributed documents. Categories for authors or documents were not available. The profiles were rather sparse and the private datasets were small.

I split this dataset into multiple datasets manually, each representing documents and users from different categories. I looked for networks of users through co-authorship information, using the Python module NetworkX[11], as well as by manually searching for keywords to select sub-graphs of users. While I did succeed in creating a set of datasets from domains including computer science, medicine, astrophysics, and chemistry, these datasets were rather noisy, and heavily biased towards electronics and computer science researchers.

7.6.4.2 Improved Datasets

Having conducted the initial investigation using the dataset above, it was concluded that to obtain better results we required an improved dataset. Therefore, I collected a second set of datasets that are used in the rest of this thesis.

The dataset used for the experiments in this thesis was built using the academic publications archive arXiv.org. This is larger than the ePrints dataset and contains categories for documents. In this dataset most of each user's documents belong to a single category, and so it is possible to categorise users. I picked two sets of users from the largest categories in this dataset, one working on astrophysics, and the other condensed-matter physics.

To create the dataset I downloaded publications metadata from arXiv.org using the *Open Access Initiative Protocol for Metadata Harvesting* (OAI-PMH) interface. I kept the information on document authorship, and document abstracts in place of the full text of the documents. The abstracts were then processed into a bag-of-words representation, and words removed which occurred in fewer than 5 documents, or more than one third of documents. I removed documents which belonged to more than one category, and discarded documents not in the categories listed earlier. I created two different datasets, one large and one small, using the top 250 and top 50 most prolific authors in the two main categories respectively.

I have created three private datasets, the first is a collection of plain text files taken from the same source as above, but under the heading of "Erotica". This dataset was

[11] `http://networkx.lanl.gov`

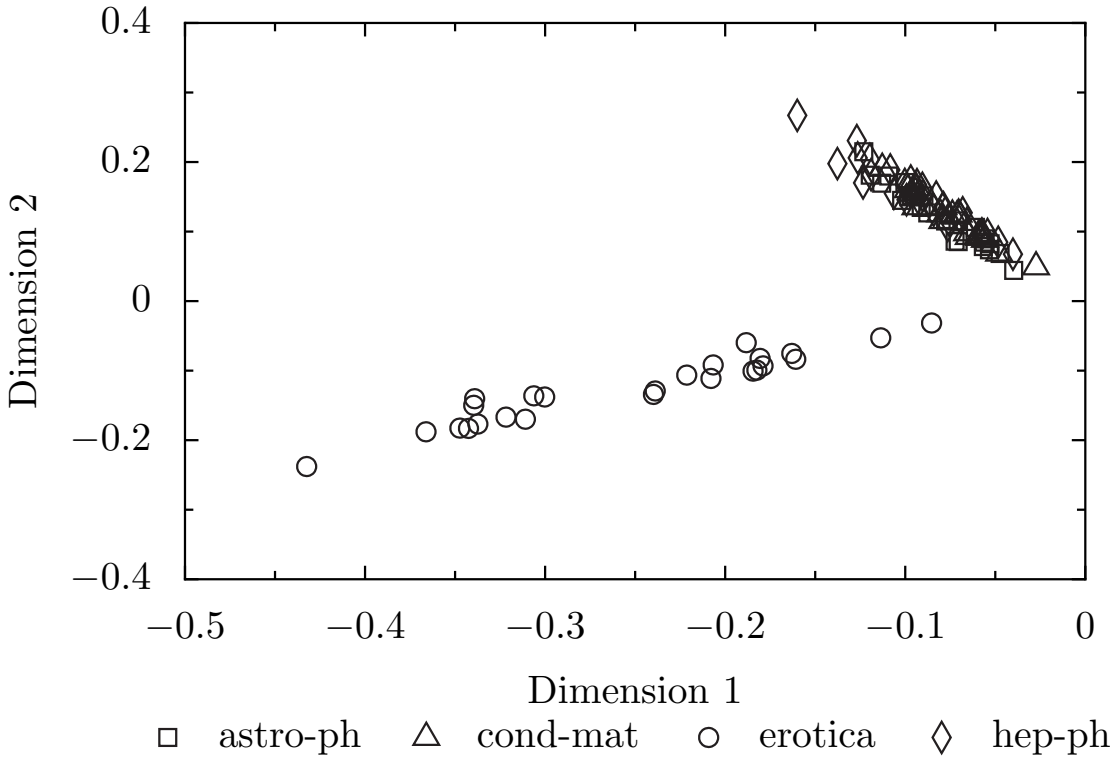


FIGURE 7.3: Dimensions 1 and 2 of LSA Projection of Datasets

much larger than the "Anarchy" and "Drugs" datasets, and slightly less problematic to work with. These documents represent material substantially different from the public documents, and representative of information that users would like to keep out of their public profiles.

The second private dataset was formed from the arXiv.org dataset, but using documents not included in the profiles. The third largest set of documents belonged to the high-energy physics category, of which 5000 documents were selected to form a private dataset. To form the out-of-context information dataset I took around a third of the documents from the profile dataset to form separate astrophysics and condensed-matter physics datasets.

To visualise the differences between each of the datasets, I built an LSA model using a sample of documents, and plotted the first 3 dimensions in Figure 7.3 and 7.4. From these you can see that the erotica dataset is markedly different from those derived from arXiv.org, but there is some overlap in the physics datasets.

In Table 7.1, and 7.2 the top-25 words for the large and small datasets are listed. There is some overlap in the physics datasets, but there is little overlap between the physics dataset and the erotica dataset. It is interesting to note that the most common words in the erotica dataset are all fairly innocuous taken individually.

The number of unique words, documents, and authors in each dataset are given in Table 7.3, and 7.4 for the large and small datasets respectively. The datasets contain

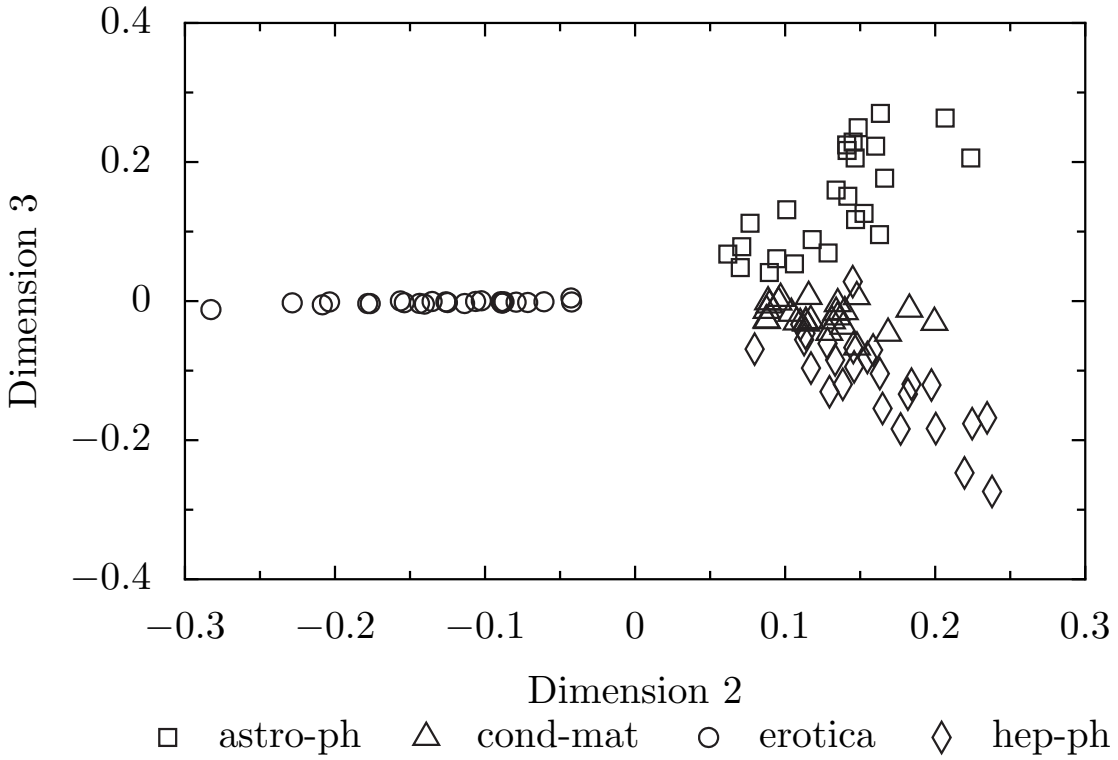


FIGURE 7.4: Dimensions 2 and 3 of LSA Projection of Datasets

62679 unique words overall. It is interesting that although the erotica dataset contains the fewest number of documents amongst the contamination datasets, it contains the greatest number of different words. This may reflect the more controlled and consistent use of language in academic publications. The average number of words per document in the erotica dataset is also much higher, as it contains short stories rather than abstracts.

TABLE 7.1: Top-25 words in large datasets.

	astro-ph	cond-mat	hep-ph	erotica
1	observation	temperature	model	time
2	galaxies	magnet	mass	hand
3	star	electron	effect	look
4	result	studied	quark	head
5	model	field	result	pull
6	mass	spin	decay	feel
7	data	result	energies	hard
8	detect	effect	parameter	leg
9	ray	model	studied	little
10	emission	phase	standard	start
11	source	depend	couple	eye
12	redshift	system	scale	bodies
13	time	transit	value	move
14	studied	observation	lead	trick
15	optic	energies	calculation	mouth
16	low	low	predict	reach
17	measure	quantum	discuss	stop
18	consist	measure	data	watch
19	sample	densities	observation	close
20	luminosity	interact	contribute	don't
21	line	structure	higgs	didn't
22	survey	calculation	product	hair
23	densities	function	obtain	finger
24	suggest	couple	qcd	slowly
25	field	superconducting	theories	left

TABLE 7.2: Top-25 words in small datasets.

	astro-ph	cond-mat	hep-ph	erotica
1	observation	temperature	model	time
2	galaxies	magnet	mass	hand
3	ray	electron	effect	look
4	result	field	quark	head
5	model	spin	result	pull
6	detect	effect	decay	feel
7	emission	result	energies	hard
8	star	studied	parameter	leg
9	data	measure	studied	little
10	source	depend	standard	start
11	mass	observation	couple	eye
12	redshift	system	scale	bodies
13	optic	transit	value	move
14	sample	phase	lead	trick
15	survey	energies	calculation	mouth
16	time	low	predict	reach
17	line	model	discuss	stop
18	studied	densities	data	watch
19	measure	superconducting	observation	close
20	consist	structure	contribute	don't
21	luminosity	interact	higgs	didn't
22	distribute	calculation	product	hair
23	low	quantum	obtain	finger
24	densities	couple	qcd	slowly
25	energies	single	theories	left

TABLE 7.3: Large dataset statistics.

	Corpus	Profiles	AP	CM	HE	Erotica
Docs	12360	12360	6672	5688	5000	4886
Words	15376	15516	10974	8049	7814	17688
Words/Doc	57.1	57.3	66.3	46.9	65.8	600.4
Authors	500	500				
Authors/Doc	1.5	1.5				
Docs/Author	38.3	38.0				

TABLE 7.4: Small dataset statistics.

	Corpus	Profiles	AP	CM	HE	Erotica
Docs	4581	4580	2528	2051	5000	4886
Words	9823	9959	7098	5114	7814	17688
Words/Doc	57.8	57.7	66.1	46.3	65.8	600.4
Authors	100	100				
Authors/Doc	1.3	1.2				
Docs/Author	57.3	57.2				

Chapter 8

Preserving Privacy

In this section I describe a number of solutions for implementing privacy-preserving profiling. Starting with passive filtering techniques, I look at term filtering in Section 8.1 and corpus projection in Section 8.2. I then move on to look at document filtering techniques in Section 8.3 and Section 8.4.

8.1 Term Filtering

This technique involves removing terms from documents which do not occur in a dictionary derived from a corpus of public documents. This removes private information such as usernames, passwords, misspelled words, slang, and noise such as text used for formatting, which should not be present in the corpus.

It will not work on private words which also appear in the corpus; grass and weed are slang terms for cannabis, but they may also appear legitimately in publications on agriculture. Term-filtering will also fail to remove words which appear out-of-context, for example the words gonorrhoea, syphilis, or herpes, may appear in medical publications, and so wouldn't be removed from the profiles of users working in other fields.

I found that using term-filtering did little to preserve privacy; the results obtained were not significantly different from those obtained without applying any filtering technique. Most words which occur in private documents also occur in public documents. Even if explicit terms are censored enough it is still possible to determine the subject of the document.

8.2 Projection

As latent semantic analysis can be used to filter noise from a dataset, we wondered if it could also be used to filter private information from a dataset. By training the SVD on a corpus of public documents and reducing the number of dimensions we hoped that private information would not be well represented by the largest singular values, and would be filtered or attenuated by the process of projecting them into concept-space. Ideally the private information would be orthogonal to the projection, and would be projected onto the zero vector.

To test this theory I looked at the similarity between the original TF-IDF document vectors from each dataset, and the same vectors which had been projected into concept-space and back. I trained a LSA model using our corpus at 100 dimensions. The average cosine similarity for each dataset between the original and reconstructed document vectors is shown in Figure 8.1. I found that the highest similarity was found in the datasets which were most similar to the corpus, and as we hoped the documents in the datasets of private vectors were changed most by the projection. We would not expect the vectors to be orthogonal as there will be some similarities between public and private documents.

The projection of data onto a lower-dimensional concept space provides some blurring of information, as the dimensions representing the directions of least variability are removed. By removing this information innocuous documents and terms will become more similar to private documents and terms, providing a measure of plausible deniability, at the expense of loss of fine detail.

In practice I found that projection using a corpus of public documents did little to remove private information from profiles. The decomposition built from public documents must be sufficiently rich to capture private information, because of the variability of natural language and the presence of innocuous information within the private dataset.

8.3 Global Document Filtering

Given the failure of term filtering to preserve privacy, I moved on to look at techniques which remove whole documents from a profile. This requires training a system to remove documents classified as private from profiles.

The first approach I took uses a single global classifier trained on a labelled collection of public and private documents, to filter out documents classified as private. As a single classifier is used, if a document appears in multiple profiles, it will be removed from all of them. I used a naïve Bayesian classifier with a multinomial probability distribution.

The Bayesian technique is based on a multinomial distribution with Laplacian smoothing (Manning et al., 2008). The probability that a document belongs to a given class is given

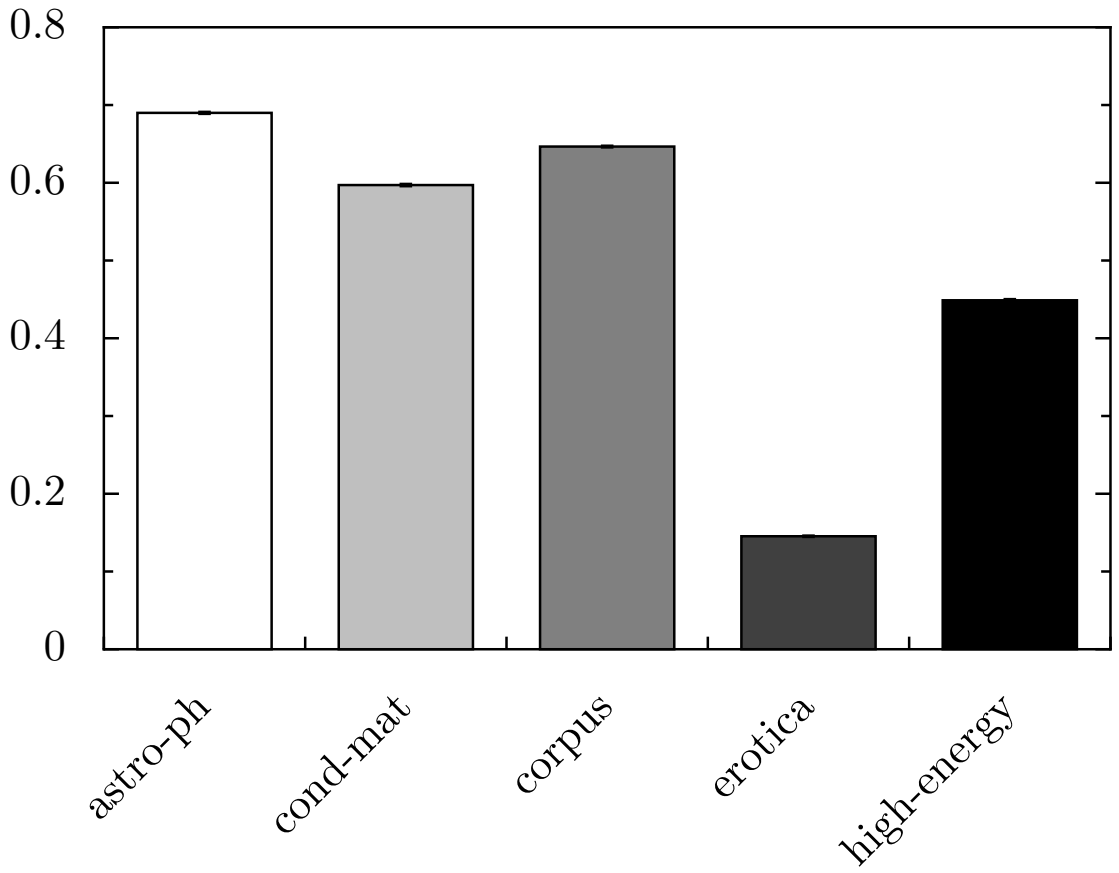


FIGURE 8.1: Cosine similarity between original and reconstructed document vectors.

by,

$$P(c|d) = \frac{P(c)P(d|c)}{P(d)}, \tag{8.1}$$

$$P(c|d) \propto P(c)P(d|c), \tag{8.2}$$

$$P(c|d) \propto P(c) \prod_{w \in W} P(w|c)^{n_{w,d}}, \tag{8.3}$$

where $c \in \{0, 1\}$ corresponding to private and public documents respectively, W is the set of words, and $n_{w,d}$ is the number of times word w occurs in document d. We use the log form of these equations to avoid numerical underflow,

$$\ln P(c|W) \propto \ln P(c) + \sum_{w \in W} n_{w,d} \ln P(w|c) \tag{8.4}$$

Class probabilities are given by,

$$P(c) = \frac{1 + n_c}{2 + |D|}, \tag{8.5}$$

where n_c is the number of documents in class c, and D is the set of all documents. Word

probabilities are given by,

$$P(w|c) = \frac{1 + \sum_{d \in D_c} n_{w,d}}{|W| + \sum_{d \in D_c} \sum_{w \in W} n_{w,d}}, \tag{8.6}$$

where D_c is the set of documents in class c.

This technique should work well when public and private documents can be clearly separated, for example, one might want to remove all documents which look like invoices for online transactions, or bank statements. This technique will be less effective when there isn't a clear difference between public and private documents, for example, when dealing with out-of-context documents. In this case the classifier may filter public and private documents indiscriminately.

8.3.1 Implementation

My multinomial classifier is implemented in pure python, only making use of the standard libraries to work with logarithms. In spite of this the classifier can be trained quickly, classify hundreds to thousands of examples per second, and has low memory requirements, making it ideal for use on low-powered mobile devices.

In order to run the multinomial classifier on a device, the server needs to transmit the prior probabilities for each class, as well as the probabilities for each word for each class. Assuming 20,000 words and 64-bit precision, this comes to just over 300 KB.

8.4 Per-User Filtering

To filter documents on a per-user basis we need to train a classifier for each user. While the user may provide examples of public documents, they may not be willing or able to provide examples of documents which they do not want included in their profile.

Semi-supervised learning techniques combine elements of supervised and unsupervised learning techniques (Zhu, 2005; Chapelle et al., 2010). While unlabelled training examples may be easy to obtain, labelled training examples require more effort to produce; for instance, while it is trivial to obtain a large number of documents, it takes longer to obtain a set of documents classified as "public" or "private" by a user. Semi-supervised techniques make use of a set of labelled examples, like conventional supervised techniques, but can also make use of a larger set of unlabelled examples to improve accuracy.

8.4.1 Positive Naïve Bayes

I make use of a semi-supervised variant of naïve Bayesian classification, called *Positive Naïve Bayes* (PNB) described in Denis et al. (2002). As well as being able to make use of unlabelled examples, this technique is designed not to need any negative examples at all. The method differs from conventional naïve Bayesian classification in the way the probabilities are calculated for private documents. We must also provide estimates for the class probabilities as these cannot be estimated from only positively labelled examples. Different choices for these estimates are described in a later section.

In this section I use different notation for the probability of selecting a positive document, $P(1)$, from the probability of generating a word in a positive document $Q(1)$. $P(w|1)$ is calculated in the same way as the global classifier, but calculating $P(w|0)$ is slightly more complicated. Estimates for negative word probabilities must be estimate from positive word probabilities, unlabelled examples, and class probability estimates. Starting with

$$Q(w_i) = Q(w_i|0)Q(0) + Q(w_i|1)Q(1), \tag{8.7}$$

where $Q(w_i)$ is the probability of generating w_i, and $Q(1)$ is the probability of generating a word in a positive document. This is rearranged to give,

$$Q(w_i|0) = \frac{Q(w_i) - Q(w_i|1) \times Q(1)}{1 - Q(1)}, \tag{8.8}$$

and then

$$\hat{Q}(w_i|0) = \frac{N(w_i, UD) - \hat{Q}(w_i|1) \times \hat{Q}(1) \times N_u}{(1 - \hat{Q}(1)) \times N_u}, \tag{8.9}$$

using the unlabelled documents to estimate probabilities, where $N(w_i, UD)$ is the number of times w_i occurs in the unlabelled documents, and N_u is the number of times all words occur in unlabelled documents.

If the set of documents is small then some of these probabilities are negative, so they must be normalised leading to

$$\hat{P}(w|0) = \frac{1 + \max(0, R(w))}{|W| + (1 - Q(1)) \times N_u}, \tag{8.10}$$

$$R(w) = N_u - P(w|1)Q(1) \times N_u,$$

$$P(w|0) = \frac{1}{Z}\hat{P}(w|0), \quad Z = \sum_{w' \in W} \hat{P}(w'|0).$$

where N_u is the total number of words in unlabelled documents.

If document length is assumed to be independent of document class we use $Q(1) = P(1)$, otherwise

$$Q(1) = \min(\frac{N_p}{|D_p|} \times P(1) \times \frac{|D_u|}{N_u}, \frac{1 + P(1)}{2}). \tag{8.11}$$

where D_p is the set of positive (public) documents, and N_p is the total number of words in positive documents.

In addition to public and unlabelled documents taken from a user's profile we also add an additional set of unlabelled documents taken from the public and private training data. We add a set of documents equal in size to the user's profile. This enables the system to work in cases where the unlabelled documents contain no negative examples, or all negative examples.

This technique should work well where we cannot neatly separate public and private documents, and can accommodate users' interests changing over time.

8.4.2 Bayesian Prior

My initial experiments with the per-user classifier provided disappointing results, as recall was low, especially when few positive examples were provided. To improve the performance of the system when little data is available I decided to add a Bayesian prior on the word probabilities, $P(w|1)$, for the positive class.

The prior used for the multinomial distribution, $\text{Mult}(X)$, is the Dirichlet distribution $\text{Dir}(\alpha)$, such that $X \sim \text{Dir}(\alpha)$. The Dirichlet distribution is conjugate to the multinomial distribution, so that the parameters can be updated simply by adding word counts to the prior parameters,

$$\alpha'_w = \alpha_w + n_w. \tag{8.12}$$

When a Dirichlet prior is used our word probabilities become,

$$P(w|1) = \frac{\alpha_w + \sum\limits_{d \in D_p} n_{w,d}}{s + \sum\limits_{d \in D_p} \sum\limits_{w \in W} n_{w,d}}, \tag{8.13}$$

$$s = \sum_{w \in W} \alpha_w. \tag{8.14}$$

When all the parameters of the Dirichlet distribution are equal to one it is equivalent to Laplacian smoothing.

As we know to which group each profile belongs, we can apply different priors for each group. The public corpus is split into two groups of documents, one containing astrophysics publications, and the other condensed-matter physics publications. These subcorpora are used to learn two sets of parameters which can be applied to the corresponding profiles in our experiments.

8.4.3 Choice of Prior

The nature of semi-supervised learning makes it difficult to estimate the probability of positive examples, $P(1)$. In my experiments I used different proportions of public documents to private documents, and so unlike in Denis et al. (2002), it isn't possible to pick one prior, or to supply the correct probability for each classifier without giving it an unfair advantage.

As we do not know the correct probability we have chosen to set the probability of picking a positive example, and of picking a negative example to be equal, $P(0) = P(1) = 0.5$. Picking a lower value of $P(1)$ may increase precision, at the expense of recall, but in my experiments I found that recall is near perfect so this is not necessary.

I tested these options on the private datasets, with one model using $P(1) = 0.25$, and another $P(1) = 0.5$. I also tested one model by supplying a prior based on the unlabelled training data.

Taking the amount of public training provided by the user as 1, the amount of public training data is given by: $1\frac{1}{2} \times |\text{Extra}|$. The amount of private training data is given by: Contamination Ratio $+ \frac{1}{2} \times |\text{Extra}|$. The size of the extra training data is: $1 +$ Contamination Ratio. This leads to the following equation for P(1),

$$P(1) = \frac{1\frac{1}{2} + \frac{1}{2} \times \text{Contamination Ratio}}{1 + 1 + \text{Contamination Ratio} + \text{Contamination Ratio}}, \tag{8.15}$$

$$P(1) = \frac{3 + \text{Contamination Ratio}}{4 \times (\text{Contamination Ratio} + 1)}. \tag{8.16}$$

This equation is specific to my application and should not be used generally. Note that the PNB model assumes that the probability of encountering a positive example generally is equal to picking a positive example in the unlabelled training data.

8.4.4 Document length independence.

Denis et al. (2002) provides two different definitions of $Q(1)$, depending on if document length is assumed to be independent of document class or not. The only dataset for which we believe that document length is dependent on document class is the Erotica dataset, which contains documents which are typically around ten times larger than the public documents.

8.4.5 Implementation

My PNB implementation was largely written in pure Python, though the code to extract Dirichlet parameters from the corpus was written in a variant of Python called

Cython, which is compiled to C for speed. The classification code is identical to the NB implementation, but the probabilities used are slightly more complicated to calculate. The technique ran more slowly in my experiments, but this is largely because we used many PNB classifiers compared to one NB classifier. The training phase is slightly more expensive, but this will only be carried out infrequently, and so isn't a real barrier to deploying this classifier on mobile devices.

To run this technique a mobile device requires from the server the parameters of the prior distribution, corresponding to their rough area of expertise, which will be just over 150KB. The device also requires an additional source of unlabelled public and private training examples, in word-frequency format. Depending on the number of examples sent, this should amount to a few tens to hundreds of kilobytes at most.

Chapter 9

Evaluating Privacy

For my experiments I first divided the public corpus into three approximately equal datasets. The first and second portions were used for training and testing, while the third was used to form the out-of-context dataset. For each experiment, the private datasets was divided in half, and together with the public dataset the first half was used to train classifiers and build the LSA projection, and the second half was used to form profiles and queries. We used 10-fold cross-validation, averaging the results across each run of the experiment. The per-user filtering techniques require a number of positive training examples taken from each profile and I supplied between 10% and 50% of the positive examples in my experiments. These correspond to an average of between four and twenty positive examples.

For the performance experiments I divided the public data, and contaminated each profile with increasing proportions of random private documents, from 0.25 to 4 times the amount of public documents in a profile. For the privacy experiments I contaminated a proportion of the profiles, from an eighth to half of all profiles, with four times the number of private documents as public documents.

For all experiments but the classification experiment the pool of documents being filtered is the same as the one used to train the per-user classifiers. Documents randomly selected to act as positive examples for the PNB classifier are not removed from the pool of documents being classified. This raises the possibility that they may be misclassified. While in a real system these documents would be marked as public, this may give an unfair advantage to the PNB classifier in these experiments.

9.1 Metrics

Different metrics are required for the classification and information retrieval experiments, which are described in the next sections.

9.1.1 Information Retrieval

I use *Mean Absolute Precision* (MAP) to measure performance, which is the *Average Precision* (AP) averaged over all queries. The *Average Precision* is simply the precision of the top-r results of a query averaged over each relevant result at rank r. The equations are given below,

$$P(r) = \frac{|\{\text{Relevant retrieved docs <= rank r}\}|}{r}, \tag{9.1}$$

$$AP = \frac{\sum_{r=1}^{N} P(r) \times \text{Relevant}(r)}{|R|}, \tag{9.2}$$

$$MAP = \frac{\sum_{q=1}^{Q} AP(q)}{|Q|}, \tag{9.3}$$

$$\text{Relevant}(r) = \begin{cases} 1 & \text{if r is relevant} \\ 0 & \text{if r is not relevant,} \end{cases} \tag{9.4}$$

where R is the set of relevant documents, r is the rank, N is the number of relevant documents retrieved, and Q is the set of queries. MAP is maximised when all the relevant documents are at the top of the list of results. It would be possible to use more evaluation metrics if we considered the best-k documents, or set a threshold of similarity, below which other results are discarded.

RPrecision is the precision at the rank equal to the number of relevant results for a query, averaged over all queries,

$$\text{RPrecision}(Q) = \frac{\sum_{q=1}^{Q} P(|R_q|)}{|Q|}, \tag{9.5}$$

where R_q is the set of documents relevant to query q. RPrecision is always less than or equal to MAP, so scores tend to be lower.

9.1.2 Classification

To evaluate classification performance I use a number of simple metrics based on the number of true positive, true negative, false positive, and false negative classifications made by each classifier. I begin by looking at precision and recall,

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}, \tag{9.6}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}. \tag{9.7}$$

Precision measures how effectively private (negative) documents are filtered, and is connected to the privacy-preservation experiment. Recall measures the ability of the classi-

fier to filter private documents without misclassifying public documents, and is connected to the information retrieval performance experiments.

Classification accuracy and the F1 score give an overall picture of the classifiers' performance.

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{|P| + |N|}, \tag{9.8}$$

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{9.9}$$

where $|P|$ and $|N|$ are the sets of positive and negative examples respectively. Accuracy simply gives the proportion of examples which were classified correctly, while the F1 score is a harmonic mean of precision and recall.

We also plot *Receiver Operating Characteristic* (ROC) curves to give a more detailed picture of classification performance. ROC curves plot the true positive rate (TPR) against the false positive rate (FPR) as the threshold for classifying an example as positive decreases. A perfect classifier would give all positive examples a higher score or probability of being positive than all negative examples, which would give a TPR of 1 and a FPR of 0. A random classifier would be as likely to classify a negative example as positive, as a positive example. This classifier would produce a ROC curve going diagonally from the origin to a TPR of 1 and a FPR of 1.

ROC curves are produced by combining the true classification of each example, with the probability each classifier has assigned to this example being positive. As we are working with very small probabilities using logarithms, we instead use a ratio of probabilities,

$$\hat{P}(x = 1) \approx \ln(P(x = 1)) - \ln(P(x = 0)), \tag{9.10}$$

where x is an testing example. This list is then sorted in descending order of $\hat{P}(x = 1)$, and we iterate through this list moving up one step if an example's true classification is positive, and right one step if it is negative, eventually bringing us to (1, 1). These steps are calculated as follows,

$$\Delta_x = \frac{1}{|P|}, \quad \Delta_y = \frac{1}{|N|}. \tag{9.11}$$

When plotting ROC curves, the results of each run of each experiment are combined to produce a single list of predictions.

9.2 Model Parameters

Before carrying out these experiments I evaluated the system with uncontaminated profiles using the public corpus, to determine the "best" number of dimensions to keep in the LSA model, as shown in Figure 9.1. Higher scores of MAP and RPrecision indicate

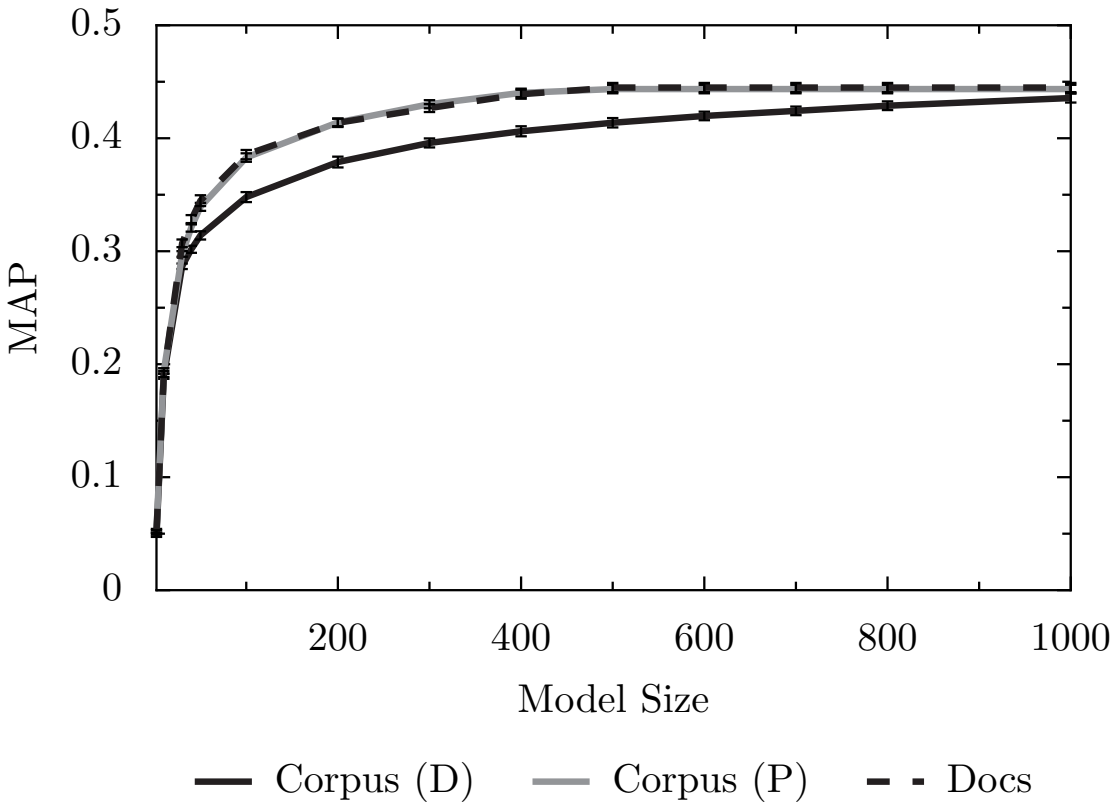


FIGURE 9.1: Performance under varying model dimensions

better performance. 'Docs' refers to a model built from the documents themselves, while 'Corpus (P)' and 'Corpus (D)' refer to models built from a public corpus using profiles and individual documents respectively. The best results are achieved when the number of dimensions equals the number of profiles, that is 500, and the SVD is exact rather than an approximation. Reasonable results are obtained down to a rank of around 100, which also speeds up computations and slightly improves privacy-preservation compared to using 500 dimensions. In the rest of this thesis LSA models are built using 100 dimensions.

9.3 Positive Naïve Bayes Parameters

In this section I consider the effect that using different priors and assumptions of document length and class independence has on performance for the positive naïve Bayes technique. In this section I have omitted the graphs for experiments using the erotica dataset where the parameters have little effect, and for the out-of-context dataset where the results are very similar to those for the high-energy physics datasets. These graphs can be found in Appendix B.

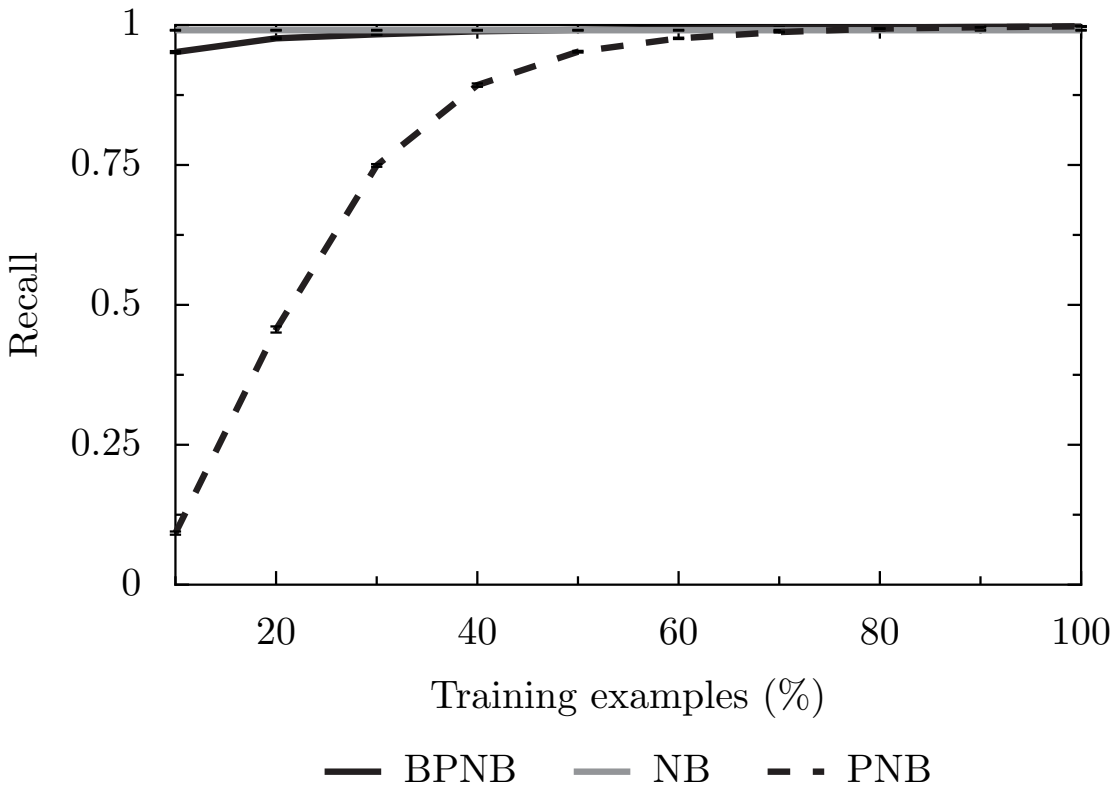


FIGURE 9.2: Effect of Bayesian prior on high-energy physics dataset.

9.3.1 Bayesian Prior

My initial experiments showed that recall was low when using the positive naïve Bayes classifier. In Figure 9.2 we see the effect of adding a Bayesian prior to recall for the high-energy physics dataset. 'NB' refers to the global technique, 'PNB' to the positive naïve Bayes technique, and 'BPNB' to the positive naïve Bayes technique with a Bayesian prior. Higher levels of recall are better. From this graph you can see that the addition of a Bayesian prior significantly and dramatically improves recall. At the level of 10% of positive training examples, the model with a Bayesian prior performs approximately as well as the model without a Bayesian prior does with 50% of training examples. This shows that if a Bayesian prior is used good performance can be achieved with a much smaller number of user-provided positive examples.

The parameters of the Dirichlet distribution correspond to having observed a given word $\alpha_w - 1$ times. By multiplying the parameters of the Dirichlet distribution by a constant we increase the equivalent number of words which have been observed, which should increase recall. I tested the classification accuracy of the PNB technique using different multipliers across each of the private datasets, as shown in Figure 9.3. I found that for the high-energy physics dataset, and the out-of-context dataset classification accuracy increased significantly up to a multiplier of four, but for the erotica dataset the multiplier used didn't make any difference, as the performance was already excellent. In the following experiments I use a multiplier of four.

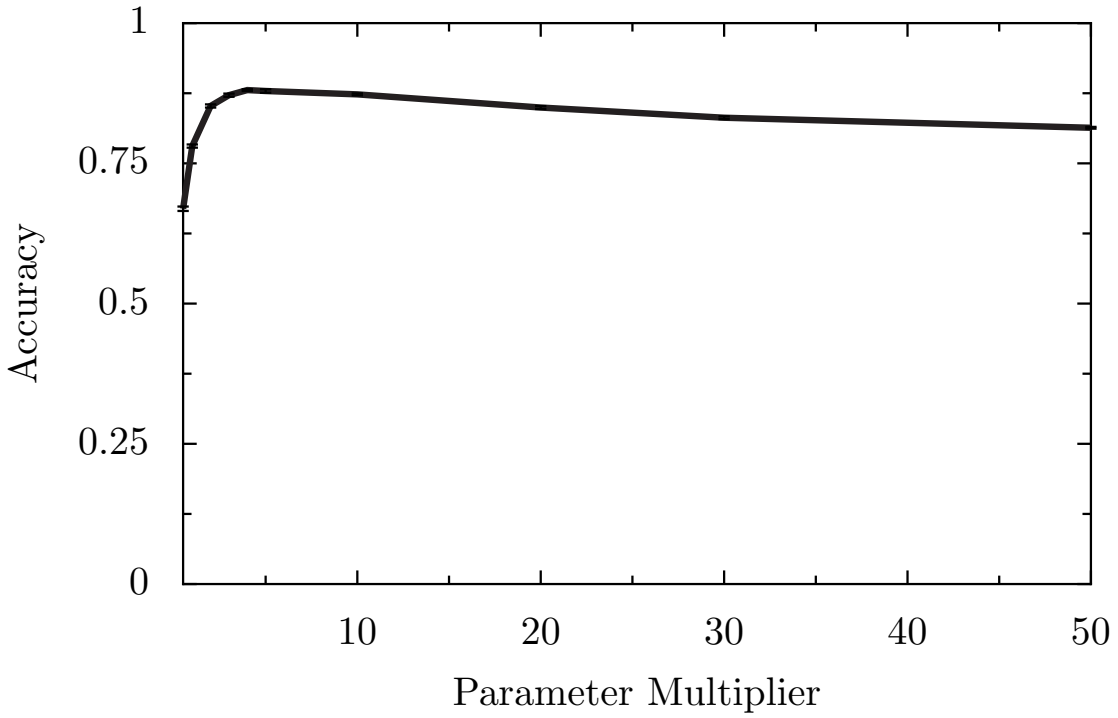


FIGURE 9.3: Effect of prior multiplier on high-energy physics dataset.

9.3.2 Choice of Prior

In this experiment I looked at the effect of using different priors on the PNB technique. I tested three Bayesian PNB classifiers using 25% of training examples, one using $P(1) = 0.25$, another $P(1) = 0.5$, and finally one using probabilities calculated from the data, indicated with 'Data'. The results of these experiments are shown in Figure 9.4. I found that although supplying a prior based on the data did slightly improve recall at low levels of contamination, at high levels of contamination it performed worse than using $P(1) = 0.5$. Using $P(1) = 0.25$ reduced recall as expected. In the following experiments we set $P(0) = P(1) = 0.5$.

9.3.3 Document length independence.

In Figure 9.5 we show the results of testing the PNB technique using the independent (I) and dependent versions of $Q(1)$ for the high-energy physics dataset. There is no significant difference between each version for any of the datasets tested, although in the following experiments the dependent version is used as it may be more robust.

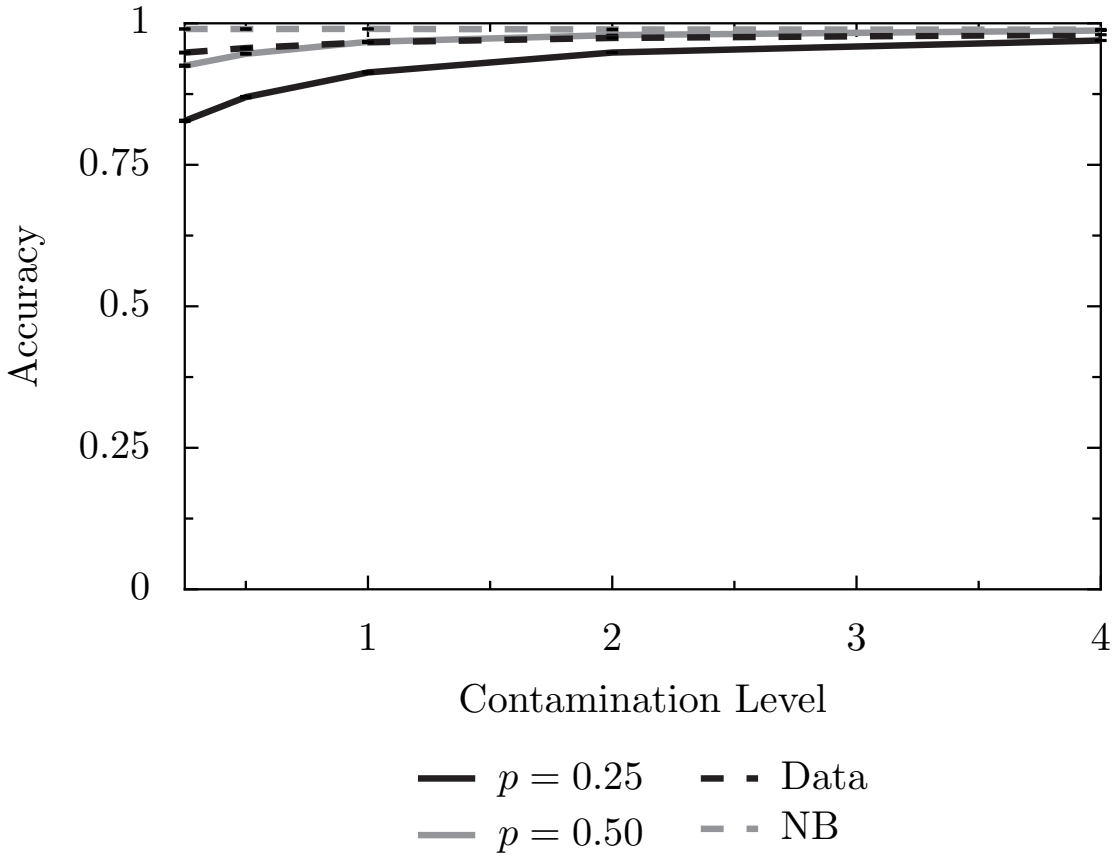


FIGURE 9.4: Effect of prior choice on high-energy physics dataset.

9.4 Results

The following section contains the results of the classification, privacy-preservation, and performance experiments. In the following graphs "Plain" refers to the uncontaminated baseline, "Term" to the term-filtered results, "Corpus" refers to the projected results, "Dirty" to the contaminated baseline, "NB" to the global multinomial filter, and "BPNBX%" to the Bayesian PNB where X is the percentage of positive examples provided for each user. The standard error of the results of each experiment are plotted as error bars. I have decided to omit the F1 score and accuracy results from the body of this thesis, as they do not tell us much that cannot be seen from the precision and recall graphs. These results can be found in Appendix A.

9.4.1 Classification

For these classification experiments we present precision, recall, and ROC curves. Precision and recall are presented with varying amounts of private information used to train the classifier, while ROC curves use equal amounts of public and private information to train the classifier. Higher values of precision and recall indicate better performance, while the larger the area under a ROC curve and the closer it is to the top-left corner

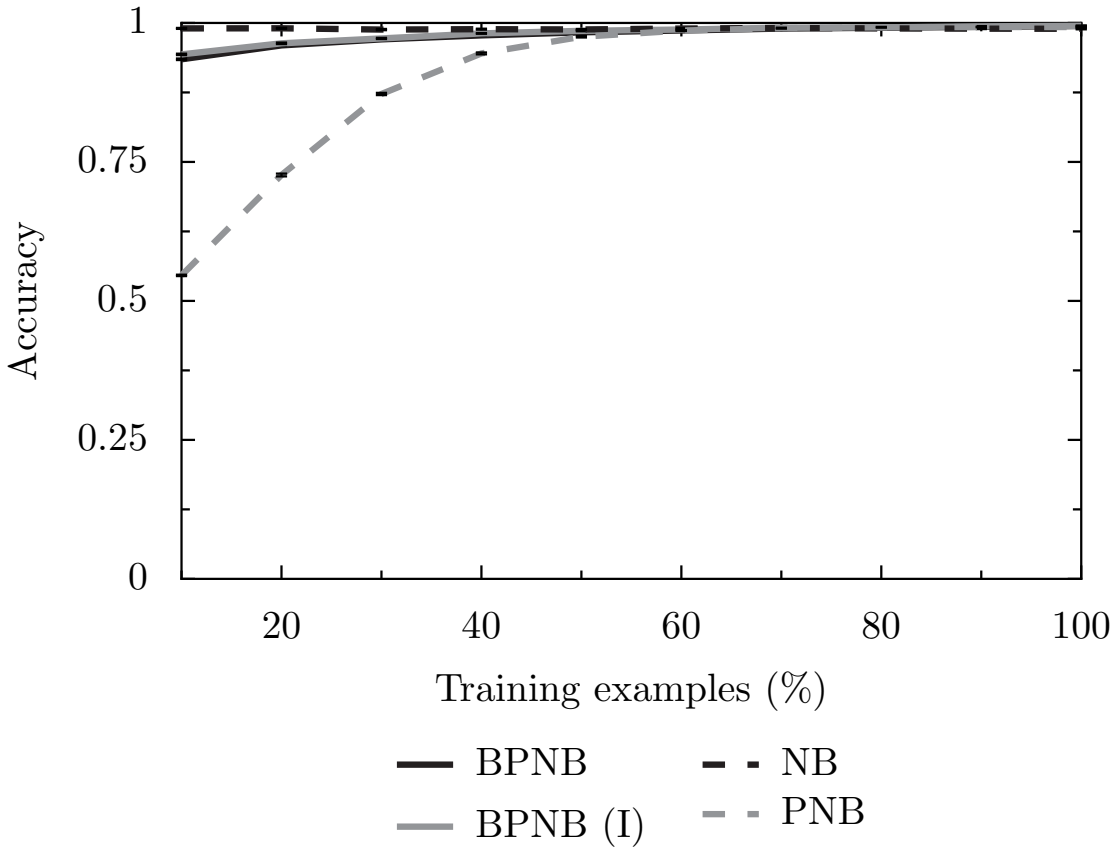


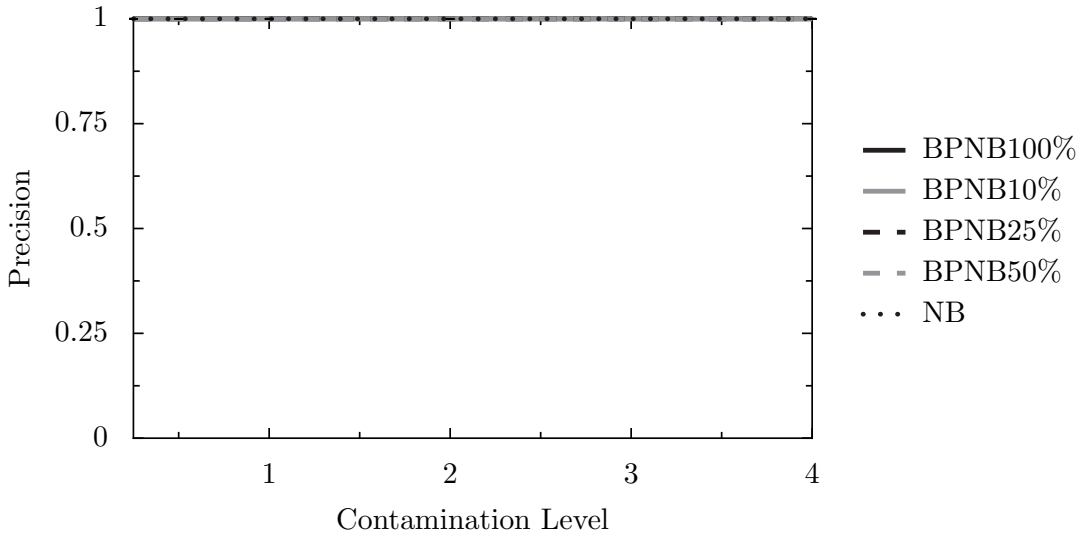
FIGURE 9.5: Effect of document length independence assumptions on high-energy physics dataset.

the better the performance.

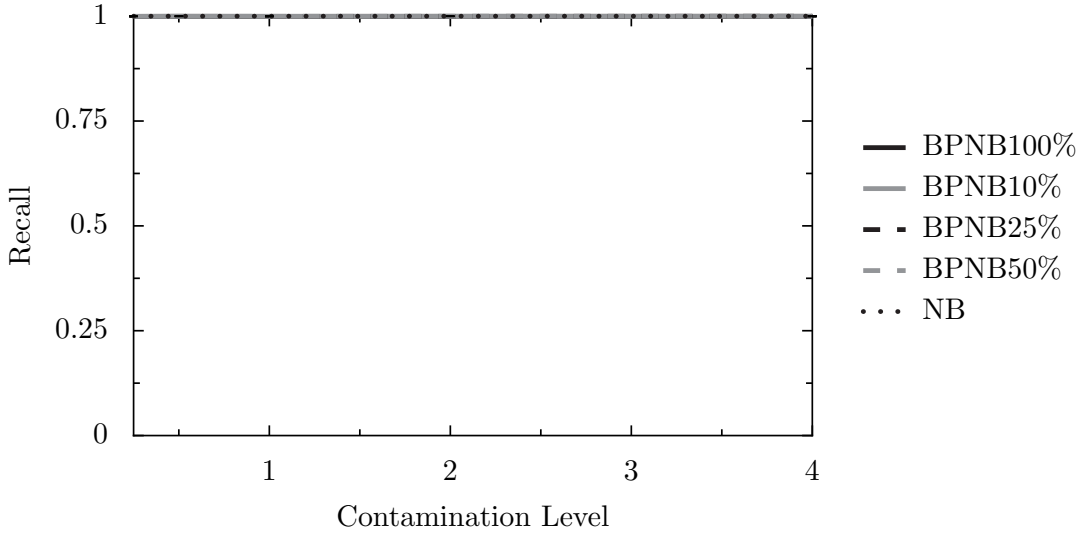
Figure 9.6.1 and 9.6.2 show the classification precision and recall results for the erotica dataset. Both the naïve Bayes and the positive naïve Bayes techniques perform almost perfectly on this dataset, even when the PNB classifier has few positive examples. Figure 9.6.3 shows the ROC curves for this experiment, which are perfect or extremely close to it for every technique. Performance is near perfect because in this case private data is easily separated from public data.

The classification precision and recall results for the high-energy physics dataset are shown in Figure 9.7.1 and 9.7.2. Here precision is near 100% even when the PNB classifier has access to only 10% of positive examples. Recall is lower at low levels of contamination, and when fewer positive examples are available. The PNB classifier has a tendency to underestimate the probability of positive documents, producing false negatives rather than false positives, when there is insufficient information. The ROC curves in Figure 9.7.3 are near perfect, even when only 10% of examples are available. It is difficult to separate each technique by looking at the ROC curves.

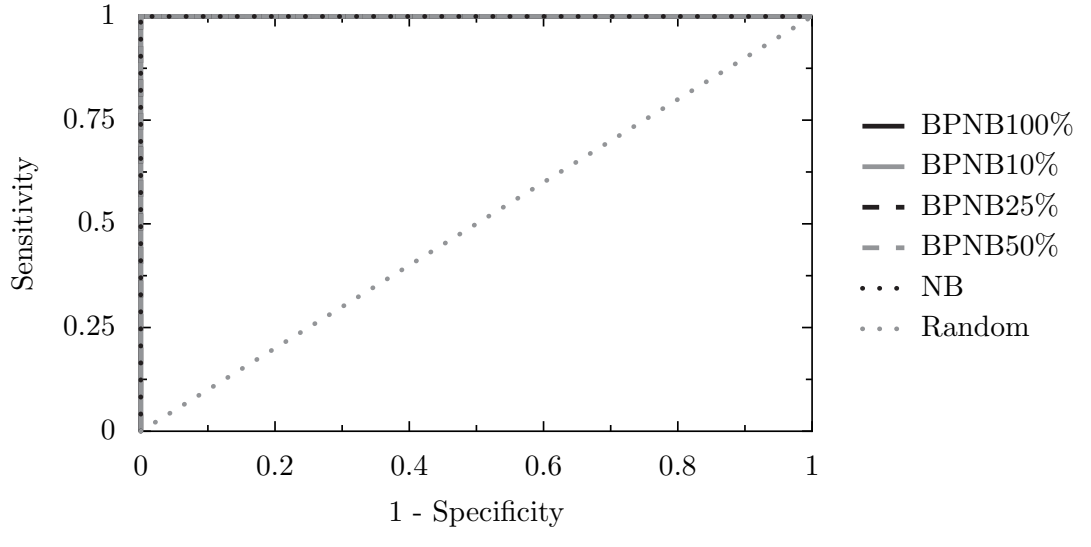
Figure 9.8.1 and 9.8.2 show the classification precision and recall results for the out-of-context datasets. As expected the global classifier cannot differentiate between public



9.6.1: Precision

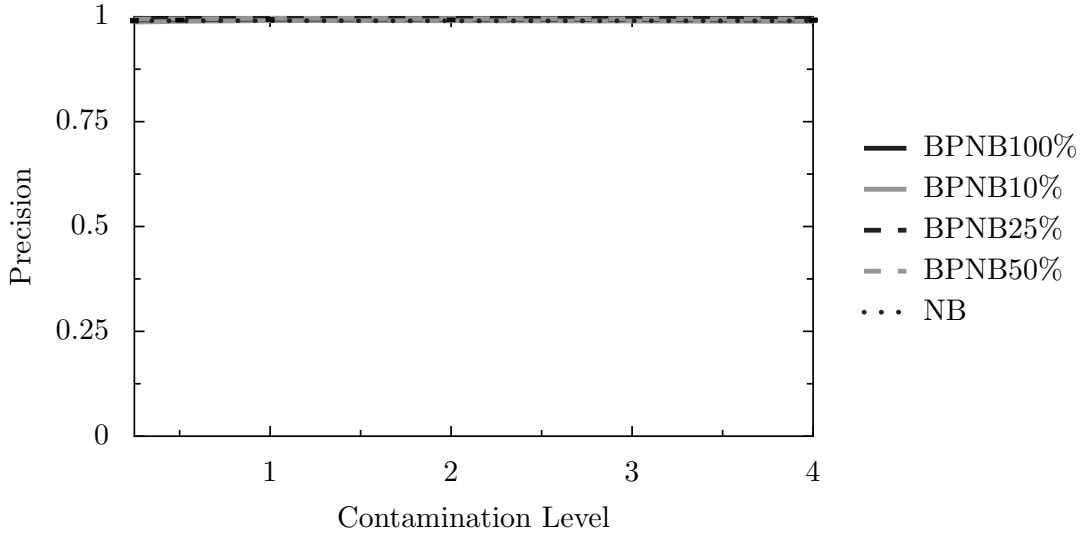


9.6.2: Recall

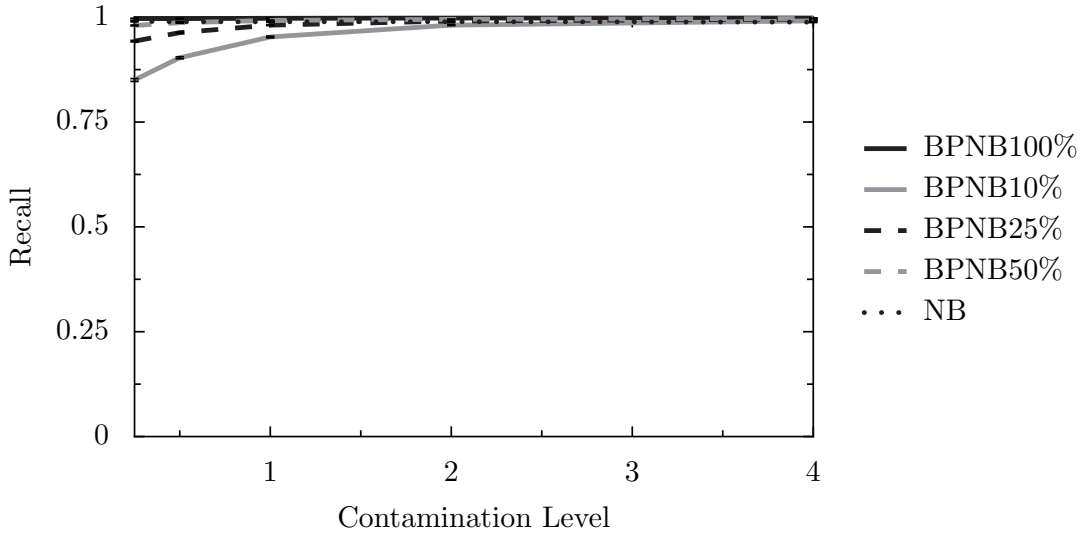


9.6.3: ROC

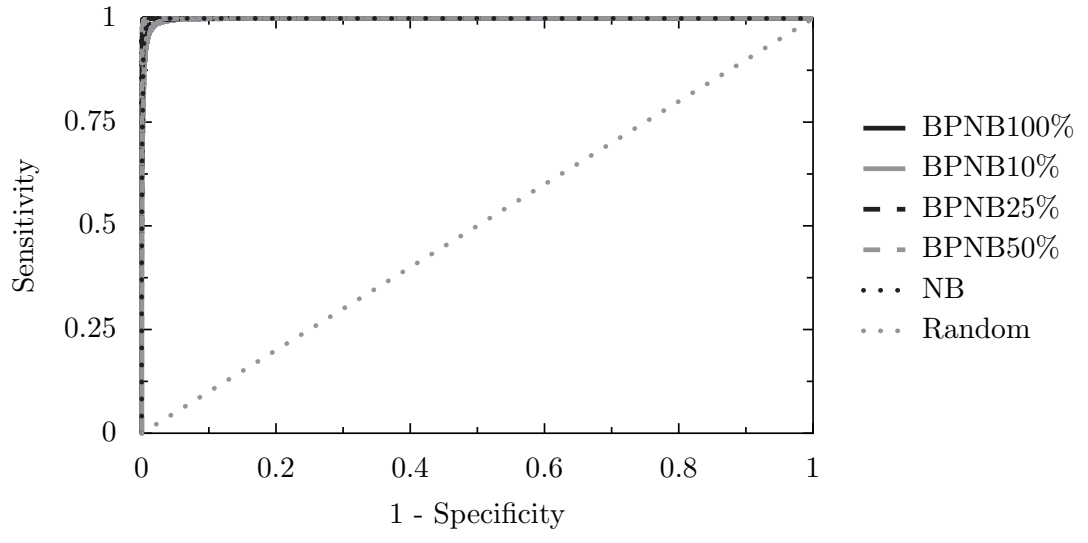
FIGURE 9.6: Classification Performance on Erotica Dataset



9.7.1: Precision



9.7.2: Recall



9.7.3: ROC

FIGURE 9.7: Classification Performance on High-Energy Physics Dataset

and private documents, and achieves around 50% precision and recall. As the PNB classifier is trained on information from each user it performs much better in this task, with very high precision and recall scores. The ROC curves in Figure 9.8.3 are near perfect for the PNB classifiers, but slightly lower than for the other datasets. It is difficult to separate the PNB techniques on the ROC curve. The Bayesian PNB technique with 10% of examples does not quite reach the top-left corner as recall is not perfect. The NB classifier performs very close to randomly. This drop in performance compared with the high-energy physics dataset may because the astro-physics and condensed-matter sets are more similar to each other than to the high-energy physics dataset.

The loss of recall in the out-of-context and high-energy physics experiments is likely because the documents being added are more similar to present in users' profiles, and are harder to separate than the documents from the erotica dataset.

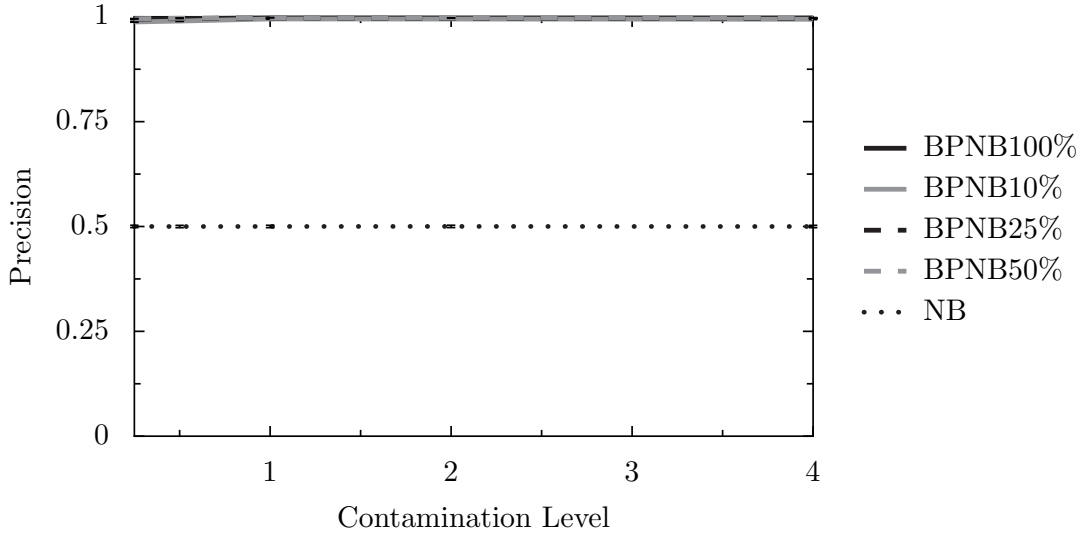
9.4.2 Privacy Preservation

For the privacy-preservation experiment lower scores indicate better performance, as high MAP and RPrecision scores indicate that contaminated profiles have been detected. In these graphs "Contaminated Profile Level" refers to the proportion of profiles that have been contaminated with private data rather than the amount of private data profiles have been contaminated with. The best results would be achieved by producing the same results as the uncontaminated baseline; points closer to the uncontaminated baseline are better. Scores increase towards the right of these graphs as the proportion of contaminated profiles increase and it becomes easier to find them.

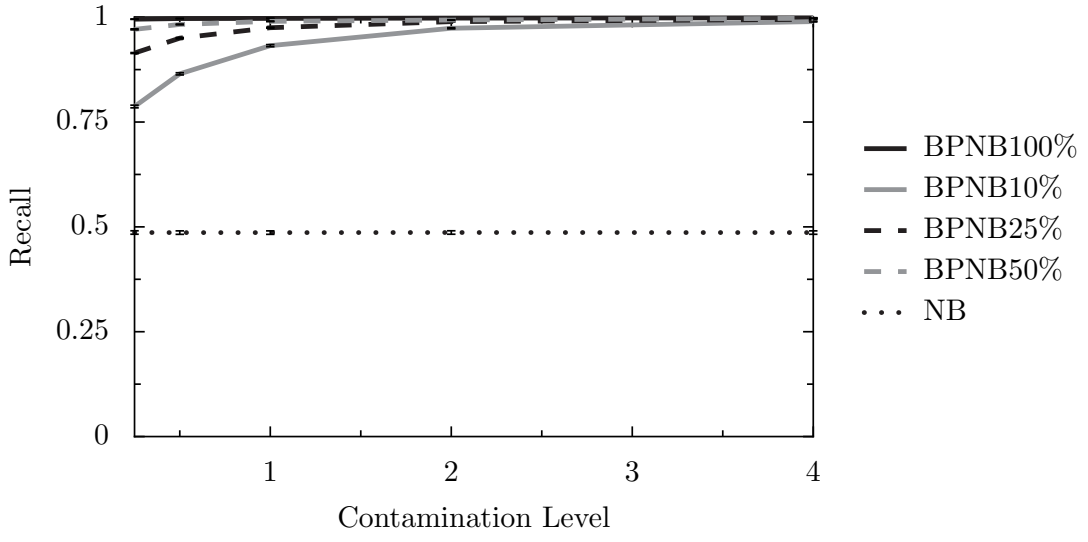
Figure 9.9.1, 9.9.2, and 9.9.3 show the results of the privacy-preservation experiment using the erotica, high-energy physics, and out-of-context datasets respectively. On the erotica dataset there is little difference in performance between the naïve Bayes, positive naïve Bayes, and uncontaminated profiles showing that these techniques are effective at removing private information. Scores for contaminated profiles are near one, indicating it would be very easy to detect contaminated profiles. Corpus projection does little to help, and the results of term filtering are almost identical to not filtering profiles at all.

Contaminating profiles using the high-energy physics dataset produces similar results to with the erotica dataset. Projection and term filtering do little to improve privacy. While the NB classifier performs perfectly, the PNB classifier does not do quite so well. As the precision of the PNB classifier was perfect, this difference in performance is not only due to letting private documents through, but rather through filtering public documents.

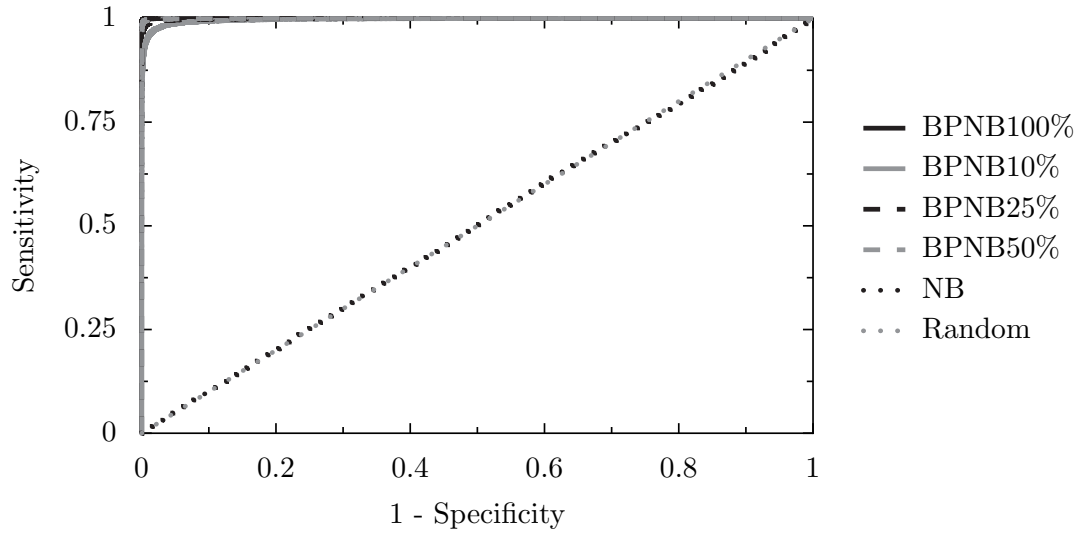
Many profiles, and therefore PNB classifiers, have no negative examples except those added as unlabelled examples. Recall is lower at lower levels of contamination, and so recall is actually higher on profiles which have been contaminated than those which



9.8.1: Precision



9.8.2: Recall



9.8.3: ROC

FIGURE 9.8: Classification Performance on Out-of-Context Dataset

haven't. As these profiles have fewer documents filtered they may be larger, and more likely to match a query than the uncontaminated documents, which leads to the difference in performance.

Using the out-of-context dataset scores are lower overall because uncontaminated profiles will match queries belonging to the same category. Here the PNB technique performs very well, but global document filtering, term filtering, and projection perform little better than no filtering at all as these techniques fail to remove private information, or in the case of the global document filter remove public and private information indiscriminately.

9.4.3 Performance

For the information retrieval performance experiments higher results are better. The difference in scales between the authorship-based relevance and class-based relevance experiments is because in the latter case there are far more relevant profiles for each query and therefore the results are correspondingly higher.

Figure 9.10.1, Figure 9.10.2, and Figure 9.10.3 show the performance results for the erotica, high-energy physics, and out-of-context datasets respectively. The scores on these experiments are partly dependent on the presence of authors in the same group who are not authors of these papers. For this reason it is better to look at the relative performance of each technique compared with the uncontaminated baseline, than the absolute performance.

Contaminating profiles using the erotica dataset produces a significant drop in performance which increases as more contamination is added. Both naïve Bayes techniques perform as well as the uncontaminated baseline, while projection reduces the effect of contamination slightly. Term filtering improves performance slightly, but the improvement not statistically significantly compared with no filtering at all.

Using the high-energy physics dataset to contaminate profiles has a slightly lower effect on results than with the erotica dataset, presumably because public documents are more similar to private documents in this case. Here the PNB techniques suffer from low performance with fewer positive examples, which is probably because recall is lower. The difference in performance is more pronounced than in the classification experiments because of the narrow measure of relevance used. The global filtering technique performs almost as well as the uncontaminated baseline. Term filtering again has little effect. It is interesting that the use of a corpus actually degrades performance more than not filtering at all; the corpus derived model may preserve more private information than the model derived from contaminated documents.

The range of the performance results for the out-of-context dataset are higher as there is

a much larger effect on performance. The per-user classifiers perform around the same as for the high-energy physics dataset. Term filtering performs similarly to the unfiltered results. As the global classifier removes documents indiscriminately performance degrades quickly. The corpus projection performs worse than unfiltered results as with the high-energy physics dataset. As private and public documents are similar in this case the model built from the corpus may be less noisy and more accurate than the one built from contaminated profiles, enabling private out-of-context data to be represented more accurately.

Figure 9.11.1, 9.11.2, and 9.11.3 show the class-based relevance performance results for the erotica, high-energy physics, and out-of-context datasets respectively. For these experiments the scores are much higher than the other performance experiments as there are more relevant profiles for each query. For the erotica and high-energy physics datasets the picture looks similar to before. On these datasets at a high level of contamination the MAP stays around 0.9, which means that only one in ten of the results belong to the wrong category. The out-of-context results show a much clearer response, so that at a high level of contamination projection and global filtering perform worse than random. For the other datasets the drop in performance, even for the unfiltered results, may not be noticeable to a user of the system.

9.5 Conclusion

In this half of the thesis I set out to develop techniques to filter private information from automatically generated profiles with no user input. This task proved to be too difficult, and so we settled on techniques which required little user input, and made assumptions about the types of private information being removed.

I had little success with passive techniques based on term filtering and corpus projection. These only filter information such as names, email addresses, or passwords and fail to filter more subtle or complex private information. From these failures we moved on to look at techniques which remove whole private documents from profiles.

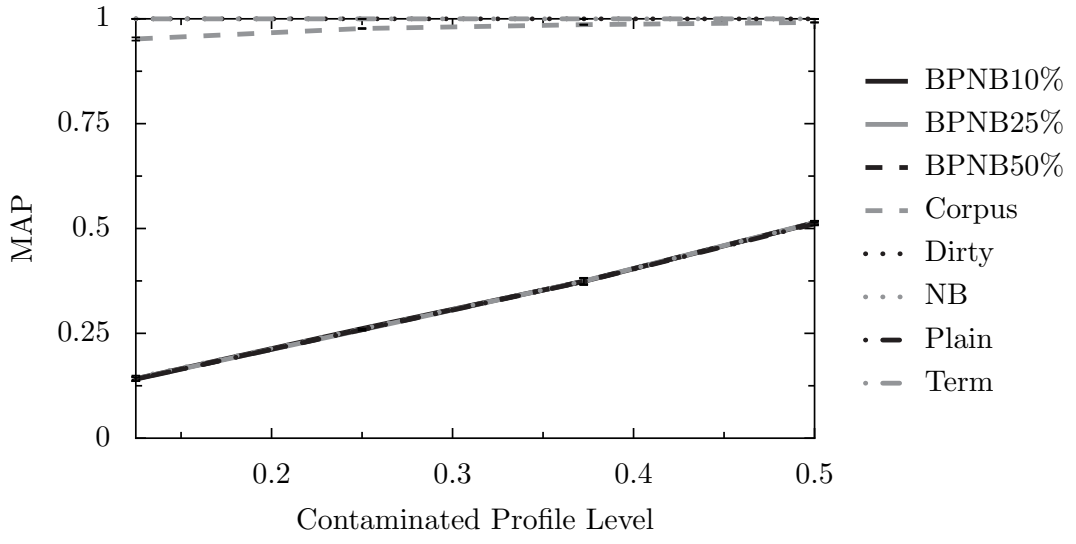
In cases where the types or topics described by private documents are known, and we wish to remove this information from all profiles, a global multinomial naïve Bayes classifier works well. It removes most private documents and leaves most public documents in the profiles.

In cases where the private information to be removed varies between profiles, a per-user positive naïve Bayes classifier works well, providing the user has supplied enough documents to train the system. Recall may be lower than the global classifier, but precision remains high, preserving privacy. The per-user technique can even cope in situations where subjects which make up the bulk of other users' profiles are considered

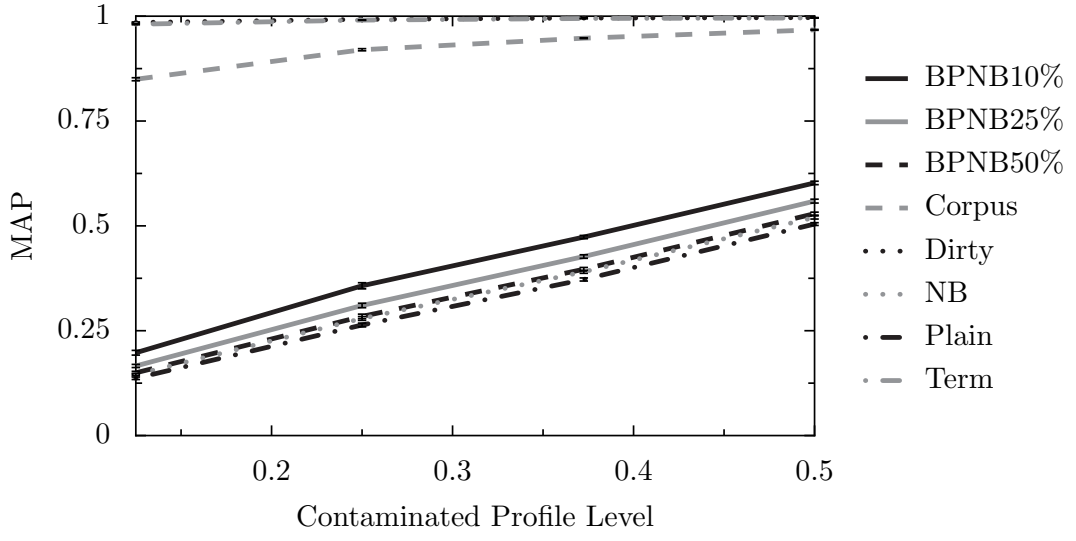
private by others.

While we have achieved good results using these techniques there is still room for improvement. In particular the positive naive Bayes technique requires sufficient training data to ensure reasonable levels of recall. We would like to achieve similar levels of performance using as few examples as possible.

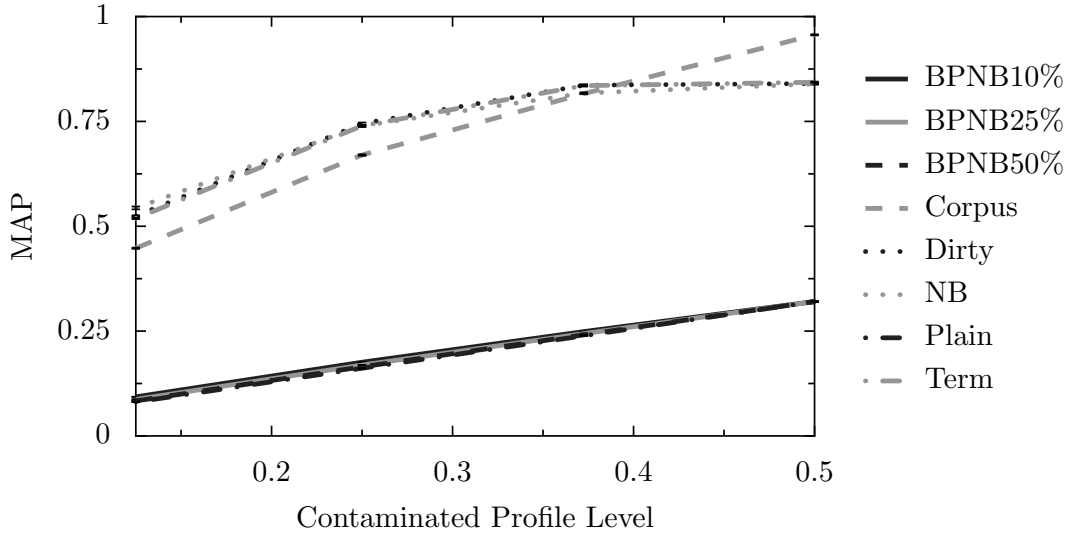
Another problem that we have not investigate is how the classifiers behave when the private documents used to train the system are different from those used to contaminate profiles. The global classifier will fail in this situation if the private documents used to train the system are significantly different than those used to contaminate profiles. The per-user classifier should cope better as the private documents are drawn from the profiles themselves, but we also add additional unlabelled examples which may cause problems. A related problem is the addition of public documents representing new user interests, or multidisciplinary work. If these new interests are substantially different to the user's current interests they may be treated as private and removed.



9.9.1: Erotica MAP

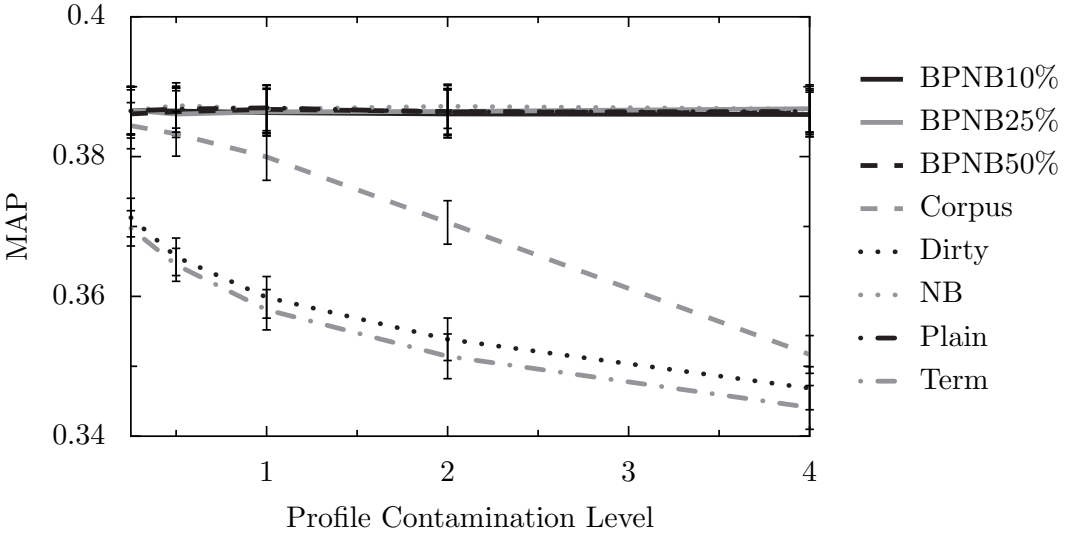


9.9.2: High-Energy Physics MAP

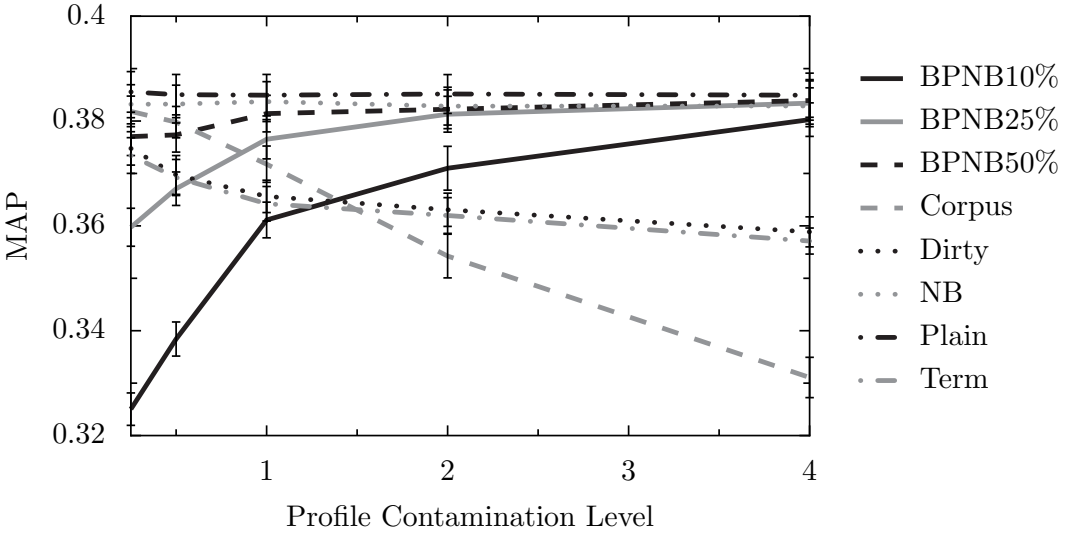


9.9.3: Out-of-Context MAP

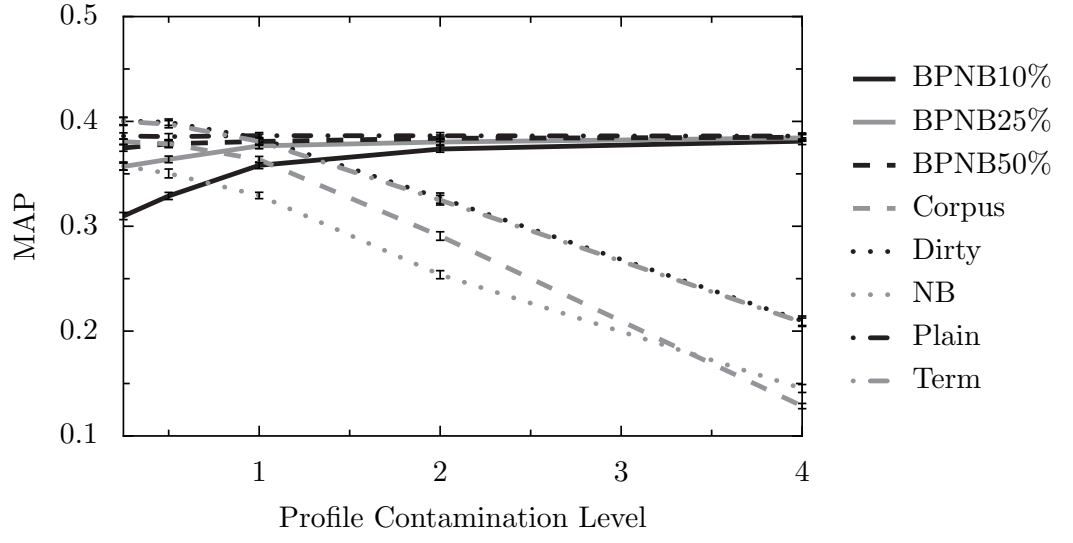
FIGURE 9.9: Privacy-preservation performance.



9.10.1: Erotica MAP

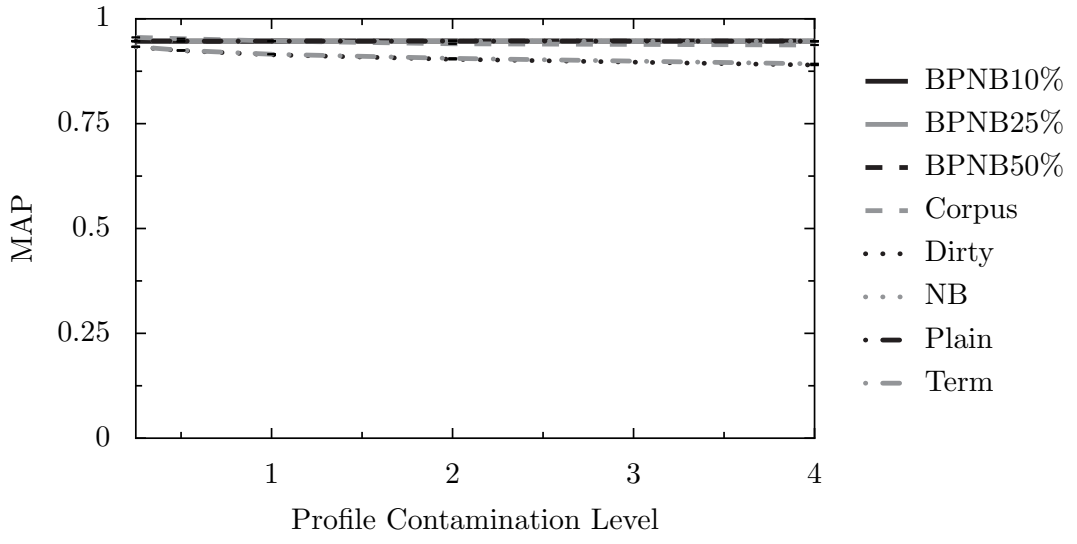


9.10.2: High-Energy Physics MAP

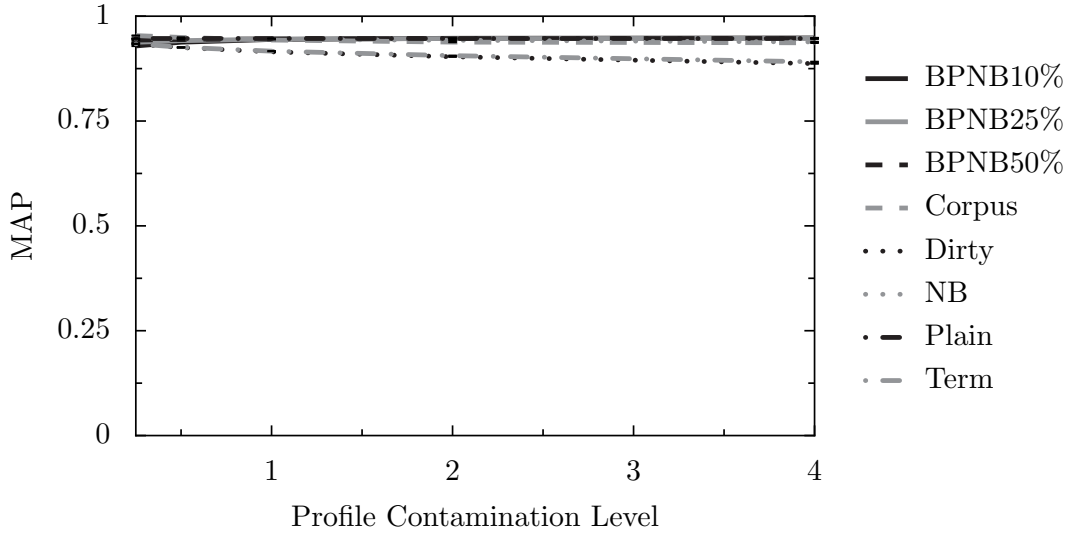


9.10.3: Out-of-Context MAP

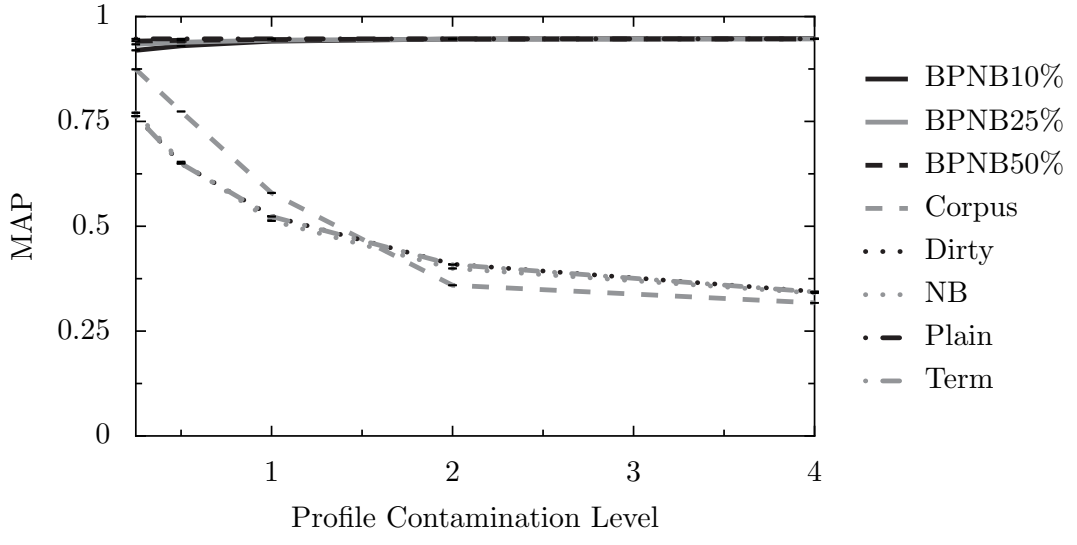
FIGURE 9.10: Information retrieval performance using authorship as relevance.



9.11.1: Erotica MAP



9.11.2: High-Energy Physics MAP



9.11.3: Out-of-Context MAP

FIGURE 9.11: Information retrieval performance using binary relevance.

Chapter 10

Conclusion

The IK system is a mobile distributed information sharing and filtering system, intended to solve the problem of finding information through the use of social and contextual information. The system was designed to facilitate the sharing of information and expertise whilst safeguarding user privacy and security. Recommender systems and text-based information retrieval systems were quickly identified as the technologies which would be used to implement the IK system. My task was to investigate how machine learning could be used to improve these technologies and meet the goals of the project.

Distributed and context-aware systems can create problems of data sparsity, and new techniques had to be developed to deal with them. The requirement for automatically generated profiles to be shared in a distributed environment also created privacy problems which had to be addressed directly. In the next sections, I discuss the findings of each half of my thesis and briefly outline directions for future work.

10.1 Recommender Systems

Recommender systems are information filtering systems which build models of user preferences based on rating behaviour in order to recommend users, items, or content. Collaborative filtering recommender systems exploit similarities between users' or items' rating behaviour in order to predict unseen ratings. This thesis looked at three main problems with recommender systems, that they are generally unable to deal with context, are usually centralised or monolithic, and that they usually generate recommendations in an ad-hoc fashion.

Most recommender systems do not make use of context; they do not take into account the situation in which a recommendation was requested. Context can be incorporated by using it to filter a ratings matrix, or to build a separate model for each context. It may also be incorporated by comparing the similarity between the current context and the one

in which the rating was made. Current context-aware recommendation algorithms suffer from an exacerbated sparsity problem, which could be remedied through the application of machine learning techniques.

The IK project requires recommendations to be provided in a mobile environment where information is not held centrally. Most recommender systems operate in a centralised manner, which means they are usually slow to update, and may not be able to react to a user's immediate needs. This also makes it difficult to incorporate contextual information. In contrast, a collection of distributed peer-to-peer recommender systems should be able to react more quickly to a user's changing needs, and so are more appropriate for the IK project.

The final piece of my work on collaborative filtering recommender systems aimed to provide a means of generating recommendations in a principled Bayesian way. I found that probabilistic recommender systems using Bayesian priors could achieve good results even under conditions of data sparsity, but the choice of model was very important. I found that the best results were achieved using a Gaussian prior on user ratings with a Gaussian-Gamma prior on user co-ratings.

Because of time constraints and other factors we were unable to complete our investigations into probabilistic recommender systems. While I obtained good results with the univariate Gaussian distribution, there may be another distribution which proves a better fit. We did not look at adding contextual information to the recommender, or to using it in a distributed setting. The use of a Bayesian framework makes it relatively simple to add additional information, the most obvious extra information to add would be item content, and user and item metadata. For example item features such as genre, or user features such as age or gender could be used to improve results. Contextual information could be added to the recommendation process in the same way.

10.2 Privacy-Preserving Profiling

The second half of this thesis considered the problem of automatically generating text-based profiles of experts' interests and expertise by processing documents stored on their devices. These profiles are then used in a text-based information retrieval system to generate expert recommendations in a semi-automatic fashion. The use of automatically generation produces profiles that are more up-to-date and consistent than those created manually, with less effort on the part of the users. Generating profiles automatically does, however, raise privacy concerns.

Users share increasing amounts of information with social networking sites, their peers, employers, and the outside world in the form of profiles and status updates. Recent events have shown that we cannot trust others to guarantee the security of our data,

and so it is important that solutions are developed for privacy-preserving profiling which do not rely on the competence of others. While the system is not as open as a general-purpose social network, there is still a possibility that user profiles could be compromised by a rogue employee, or an outside hacker. Private information in profiles may degrade the performance of the system, as well as putting employees' privacy or jobs at risk.

In this thesis I set out to develop a framework for the evaluation of privacy-preserving profiling algorithms. I developed a series of experiments designed to test the ability of privacy-preserving algorithms to preserve information retrieval performance and user privacy. I used publicly available sources of information to build corpora of public and private documents, making it possible to repeat and improve upon my experiments.

I implemented and tested a number of solutions to filter private information from contaminated profiles. I looked at using a corpus-derived dictionary to filter terms from documents; a corpus-derived LSA projection; and two document filtering methods, one using a global classifier, and the other a classifier for each user. The results of my experiment showed a large degradation in performance when using contaminated profiles. When evaluating privacy-preservation contaminated profiles could be identified reliably, with profile sanitisation techniques providing a significant improvement.

I found that filtering terms had little effect on the results. I believe this is because many of the words present in our private documents were present in our public documents. While term-filtering fails to remove private interests, it should still remove private information such as financial details. Using a corpus-derived projection was also ineffective. The technique failed to work as expected because there is enough information in the public corpus to represent the private documents. Its performance is likely made worse through the removal of public information.

Filtering whole documents provided much better performance. If the type of information being filtered is known in advance then a simple multinomial naïve Bayes classifier works very well. This technique fails, however, when it is not possible to separate documents globally into different classes. I found that a positive naïve Bayesian classifier trained on a small number of positive examples, a larger collection of unlabelled examples, and augmented with a Bayesian prior on the positive (public) class produced good results, even in the case where public and private documents could not be separated globally. At lower levels of contamination and fewer positive examples recall is lower, but precision remains high, preserving privacy.

There are several improvements that could be made to our experiments, particularly to the datasets used. In these experiments I used profiles from two groups, and a single source of contamination per experiment. In reality users may belong to many groups, and most users would like to keep multiple types of information out of their profiles. It would be interesting to repeat these experiments using multiple sources of contamination per experiment, perhaps training classifiers with one type of private information, and

contaminating profiles with another. It would be interesting to test the system using smaller keyword queries, as attackers would be unlikely to query the system with large chunks of text.

There is also room for improvement in our filtering techniques. In Denis et al. (2003) a version of positive naïve Bayes is described which adds negative examples to the positive and unlabelled examples. The addition of a small number of negative examples may help recall where few positive examples are available. In this thesis I did not exploit co-authorship information, which could be used to build a better picture of user interests through their relationships with their peers.

Currently a system based on positive naïve Bayes seems to provide good enough privacy-preserving performance to be used within a system like IK. Filtering is near perfect, and the use of latent semantic analysis will soften the effect of small numbers of private documents remaining in a profile. Without sufficient training data the system suffers from lowered recall, and this decreases information retrieval performance. We don't know if the performance loss measured in our experiments will have a tangible effect on the utility of the system to an end user. Further experiments with real profiles and real relevance data may be needed.

Appendix A

Additional Privacy Results

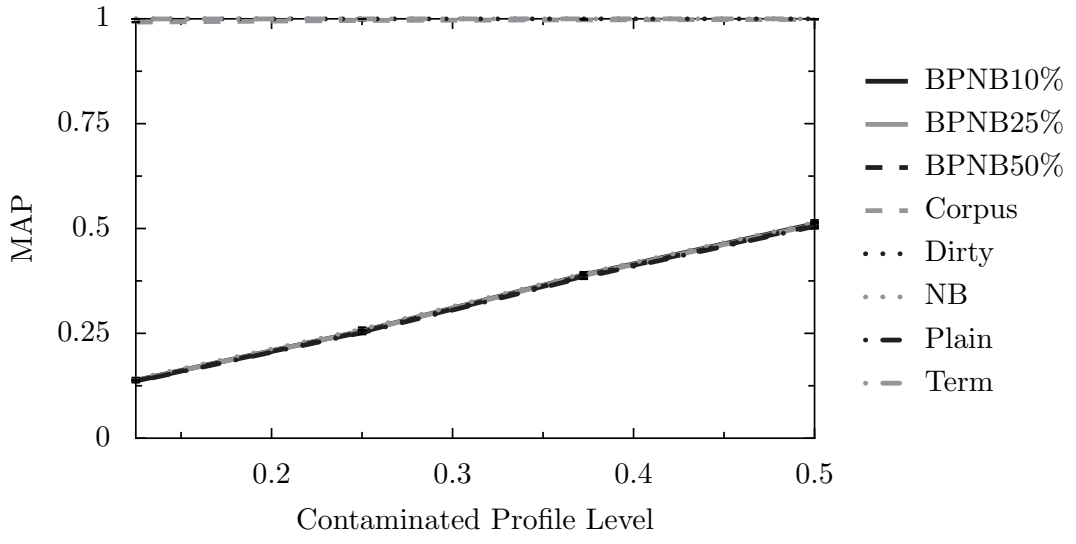
A.1 Results at high-dimensions

In this section the results obtained using an LSA of rank 500 are presented. Figure A.1 and Figure A.2 show the results of the privacy-preservation experiment using MAP and RPrecision respectively.

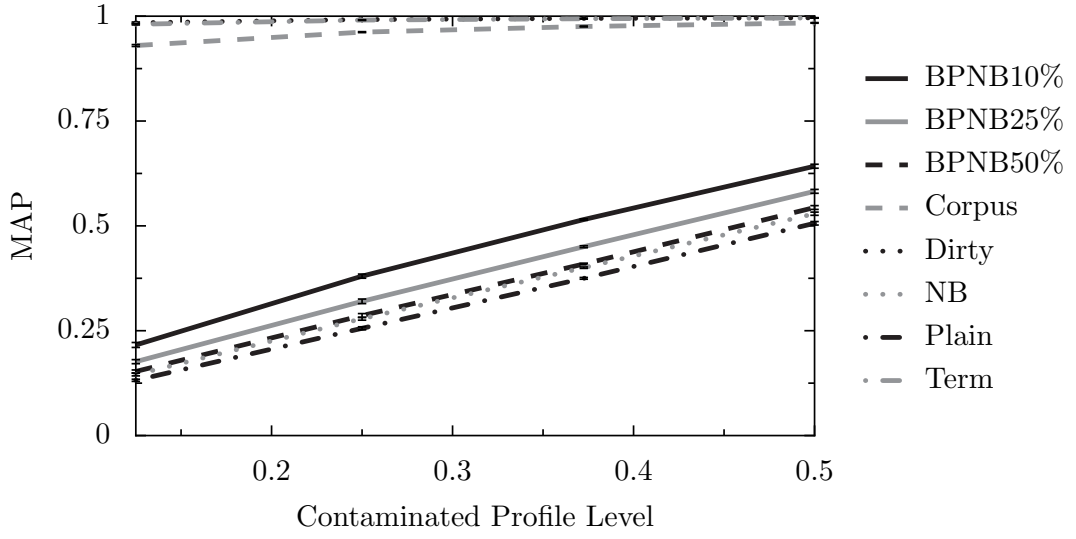
Figure A.3 and Figure A.4 show the results of the performance experiment using MAP and RPrecision respectively.

Figure A.5 and Figure A.6 show the results of the class-based performance experiment using MAP and RPrecision respectively.

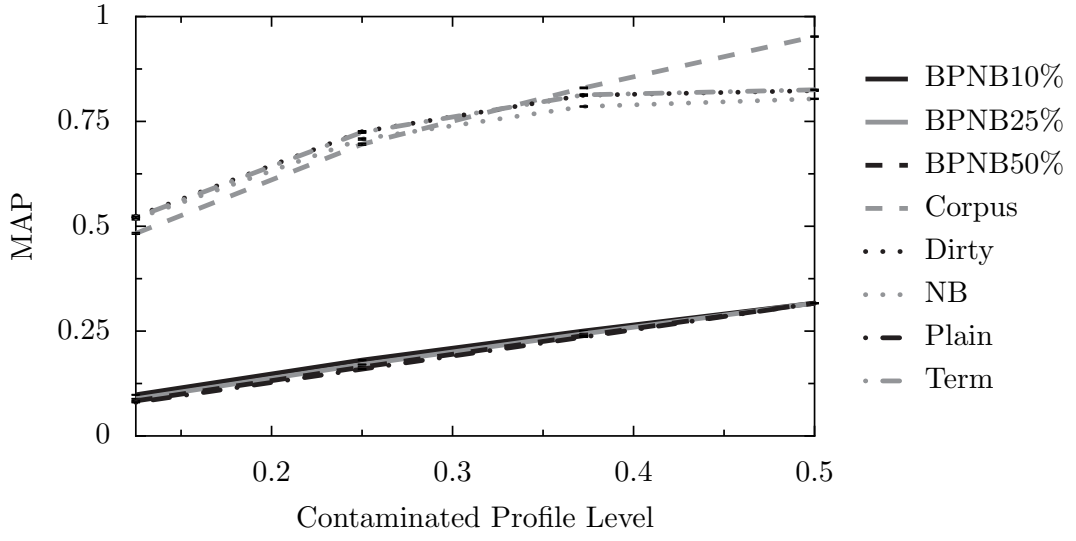
A.2 Additional Classification Metrics



A.1.1: Erotica

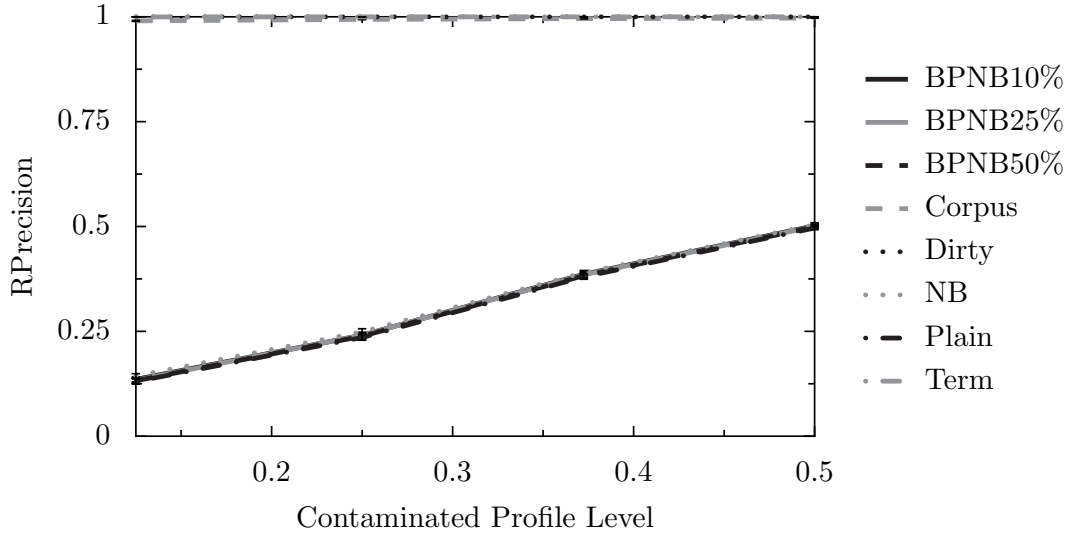


A.1.2: High-Energy Physics

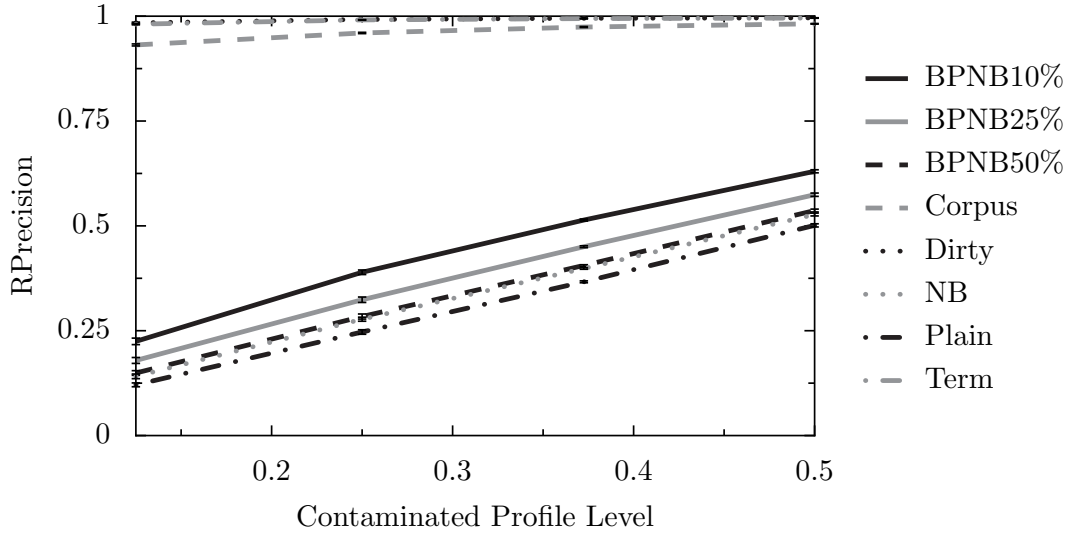


A.1.3: Out-of-Context

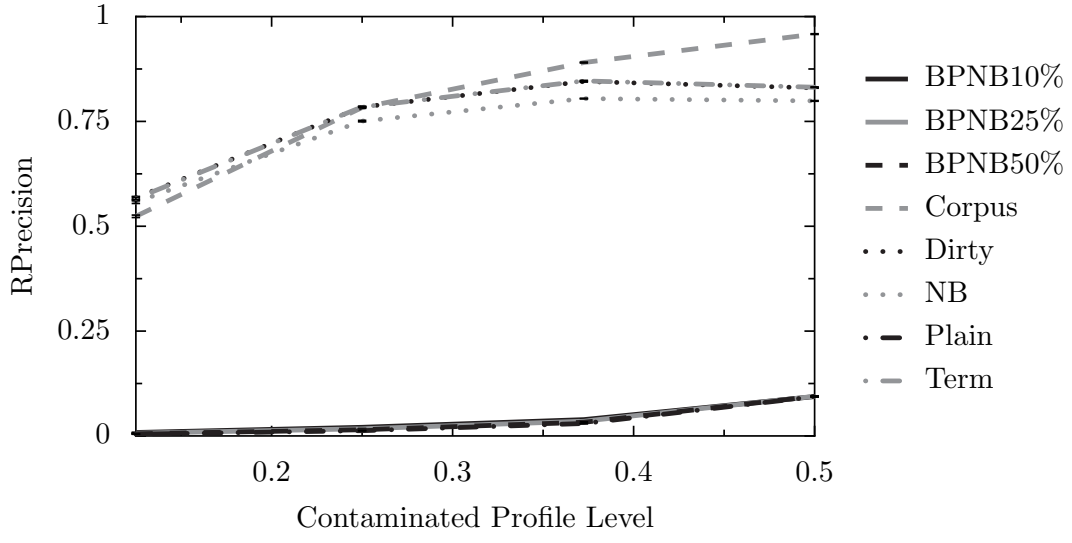
FIGURE A.1: Privacy-preservation MAP



A.2.1: Erotica

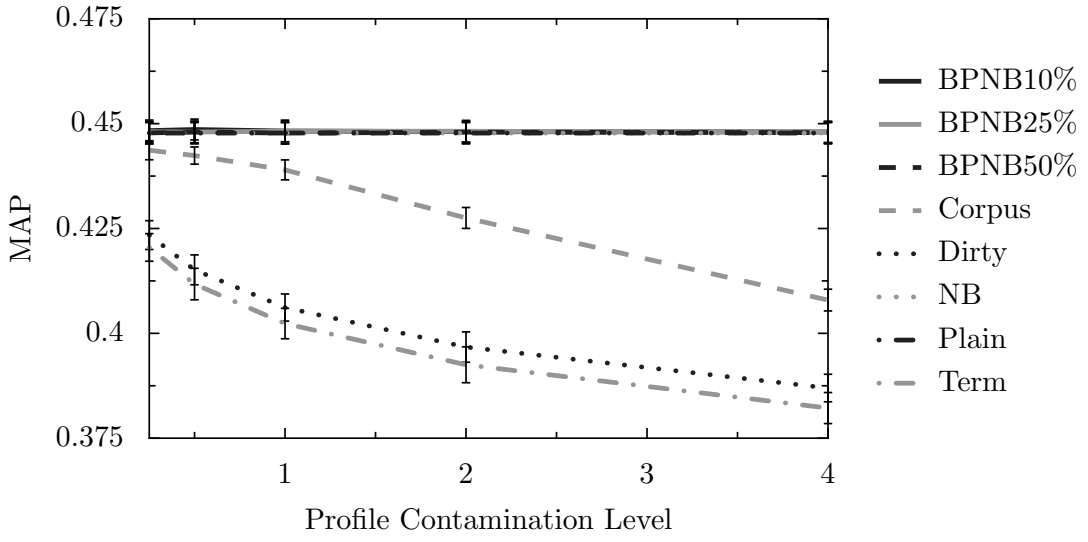


A.2.2: High-Energy Physics

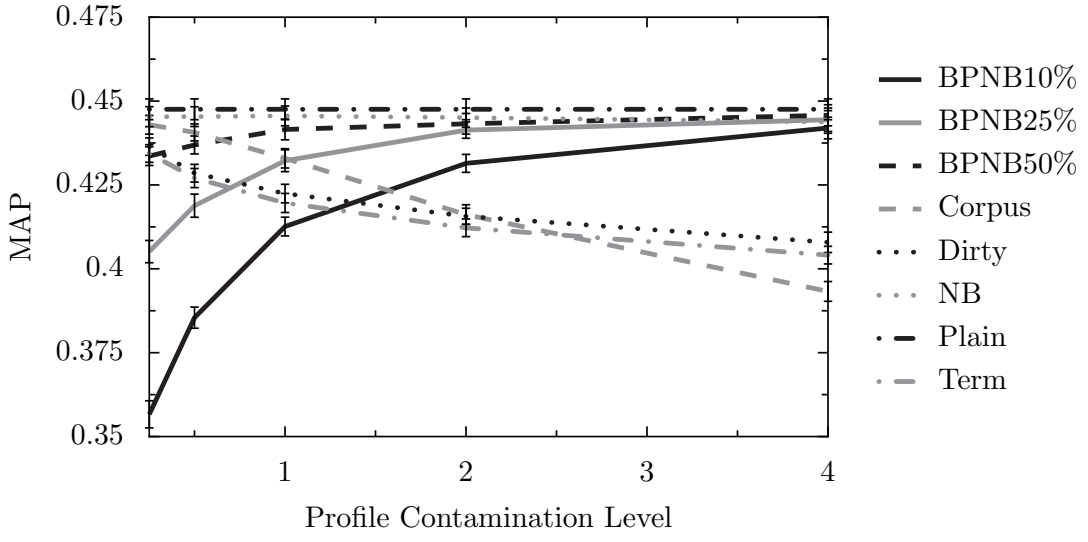


A.2.3: Out-of-Context

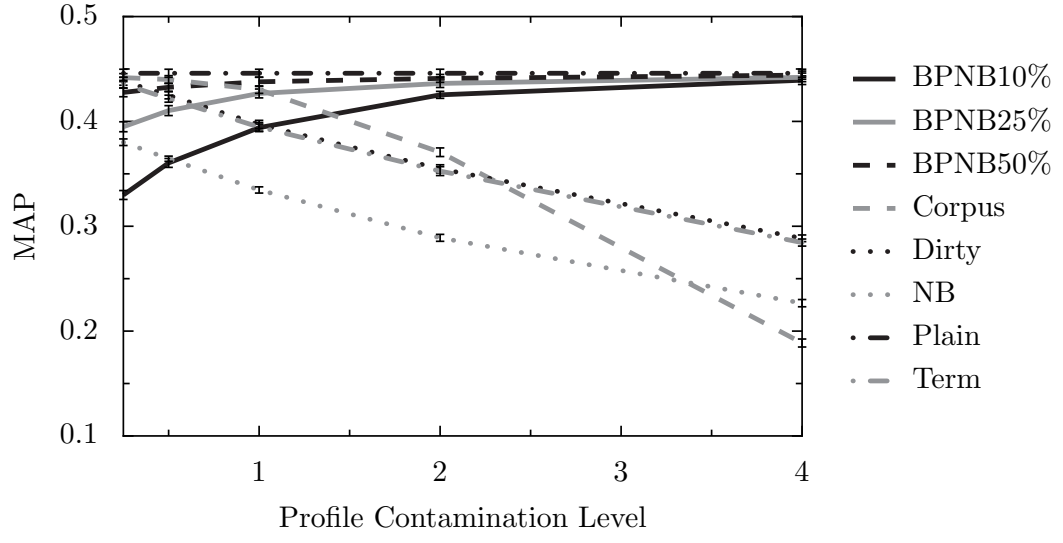
FIGURE A.2: Privacy-preservation RPrecision



A.3.1: Erotica

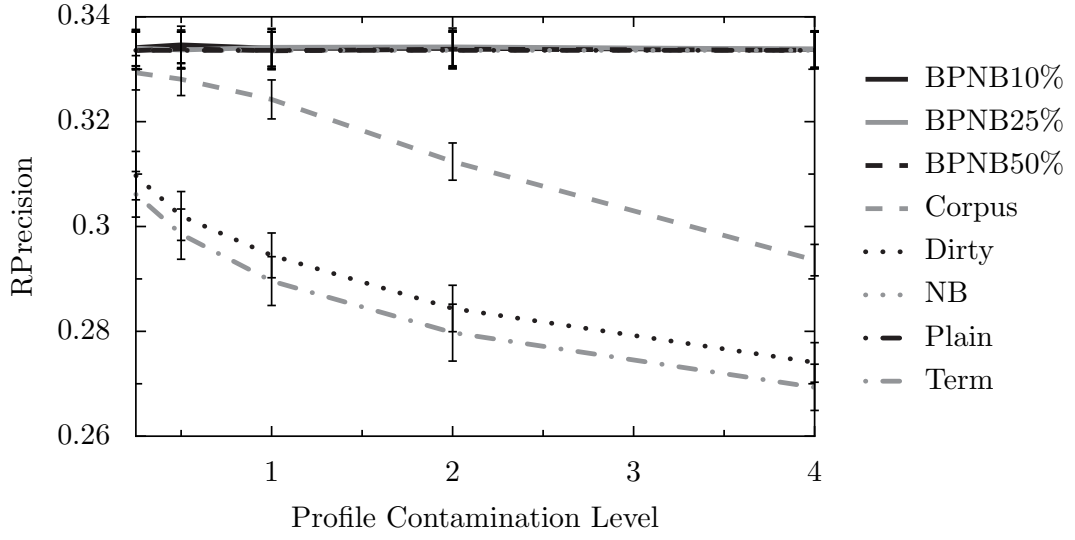


A.3.2: High-Energy Physics

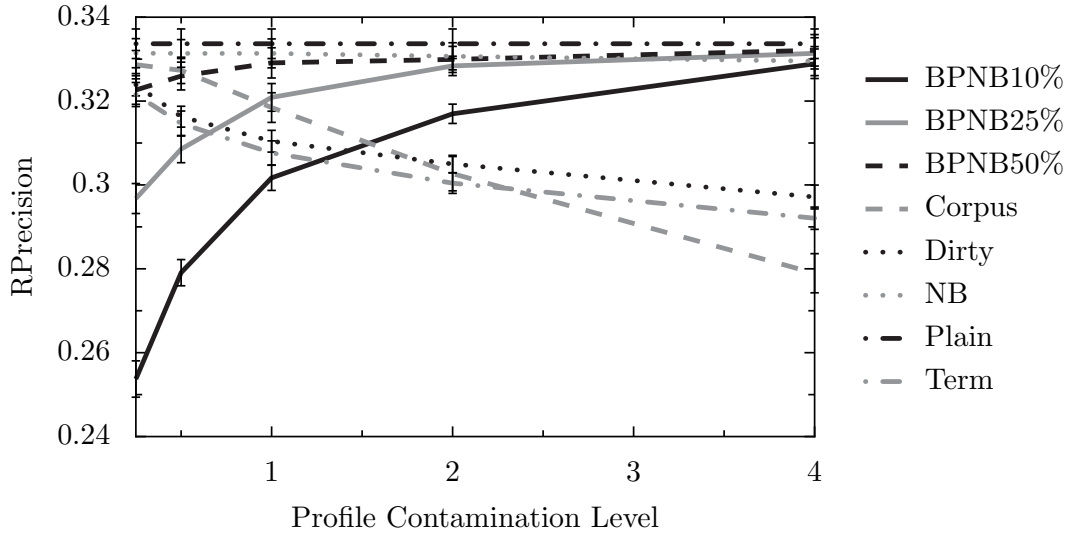


A.3.3: Out-of-Context

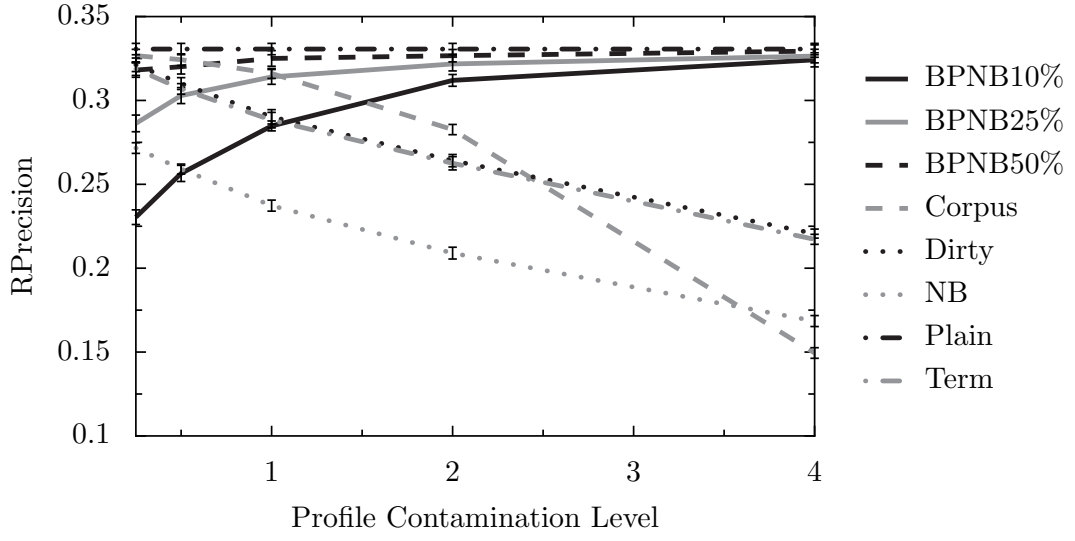
FIGURE A.3: Performance MAP



A.4.1: Erotica

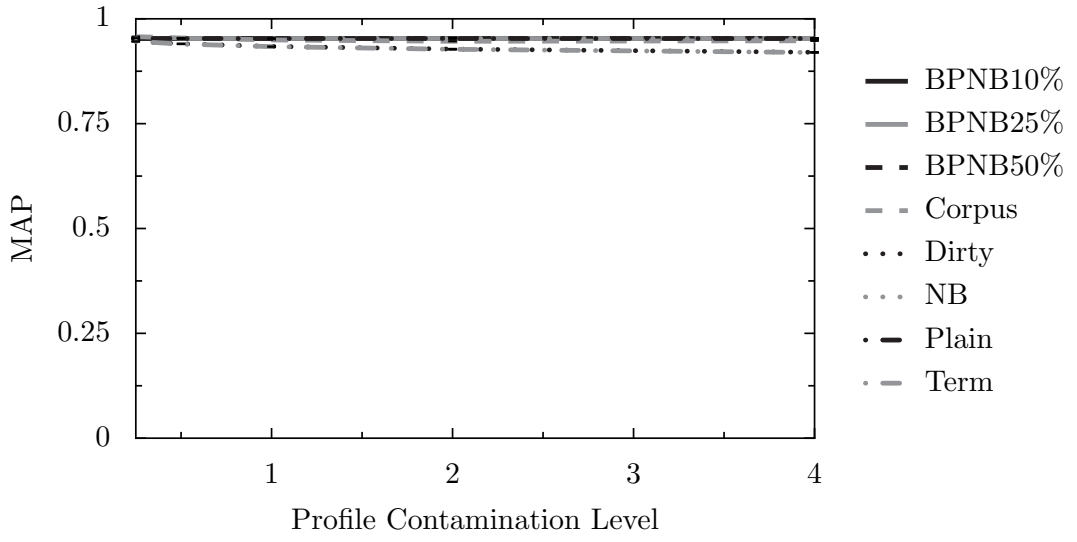


A.4.2: High-Energy Physics

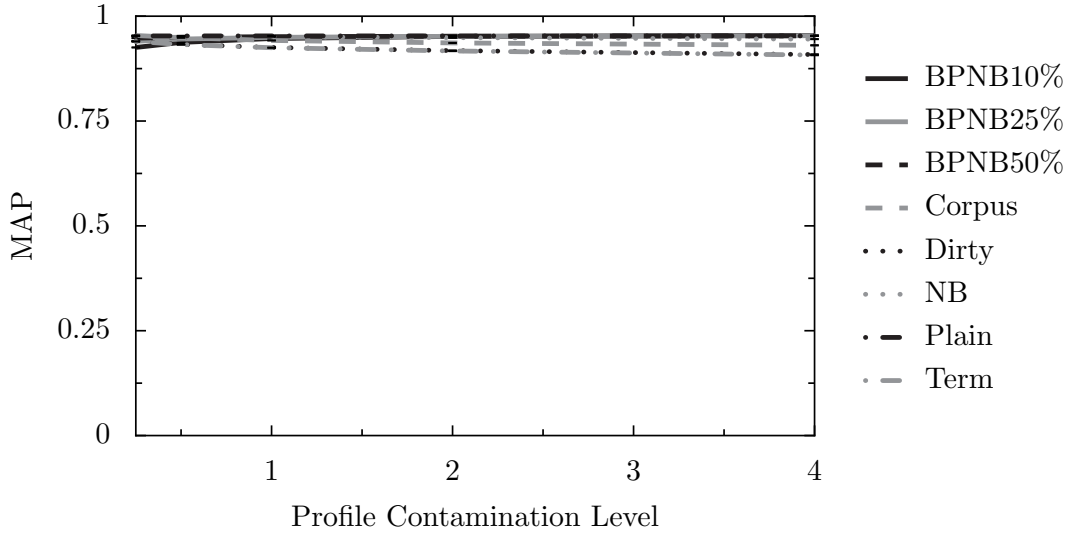


A.4.3: Out-of-Context

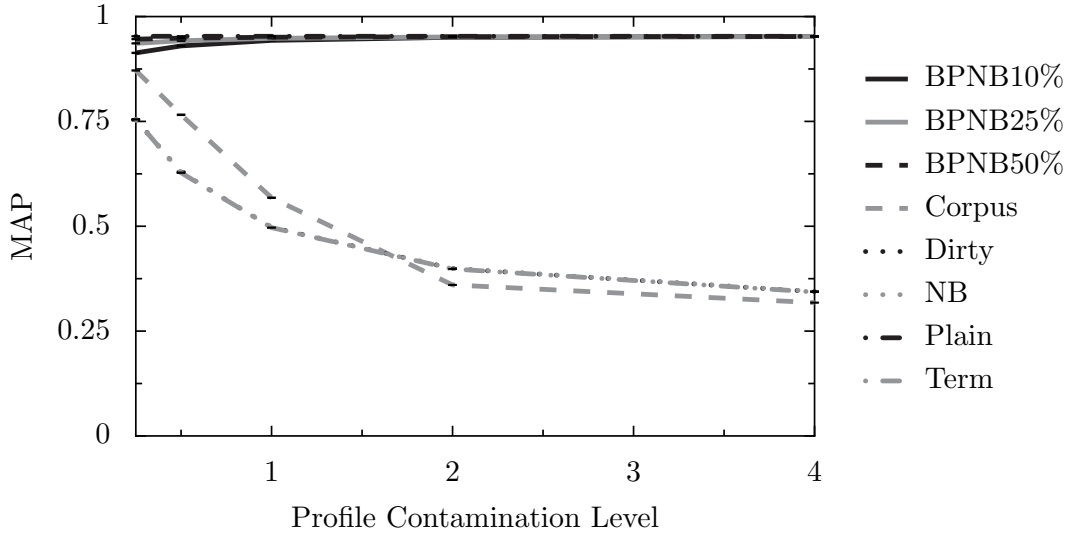
FIGURE A.4: Performance RPrecision



A.5.1: Erotica

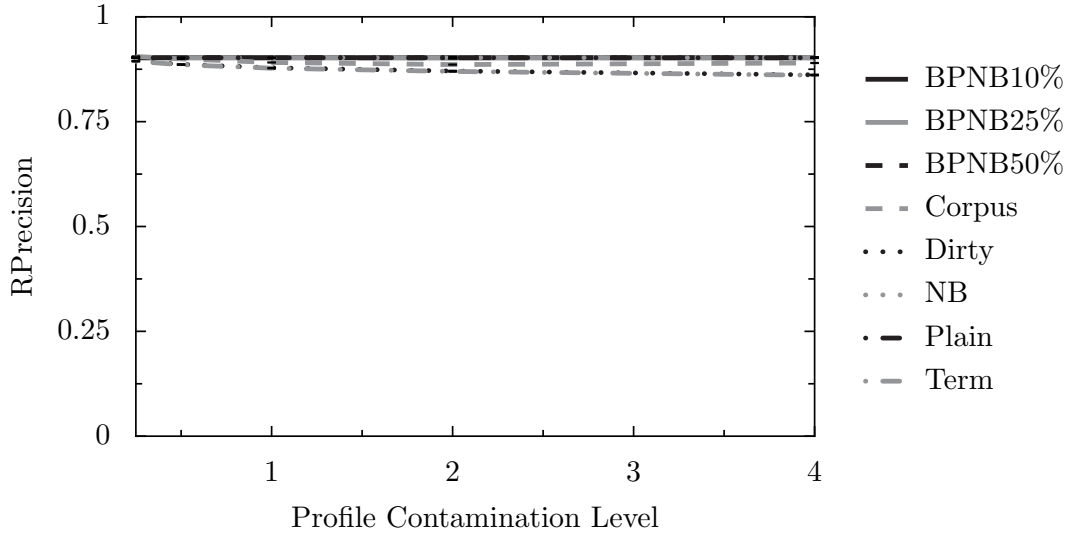


A.5.2: High-Energy Physics

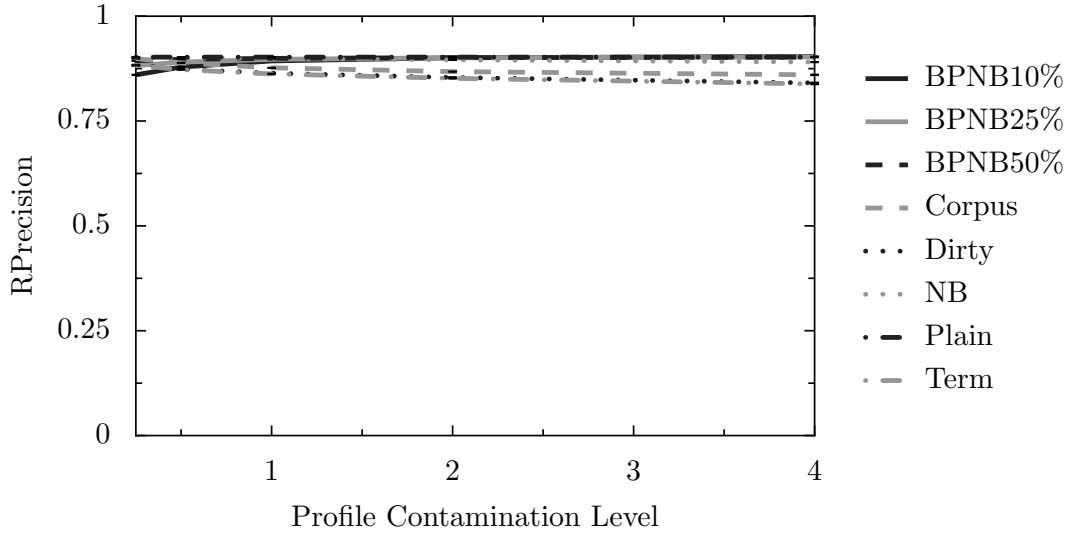


A.5.3: Out-of-Context

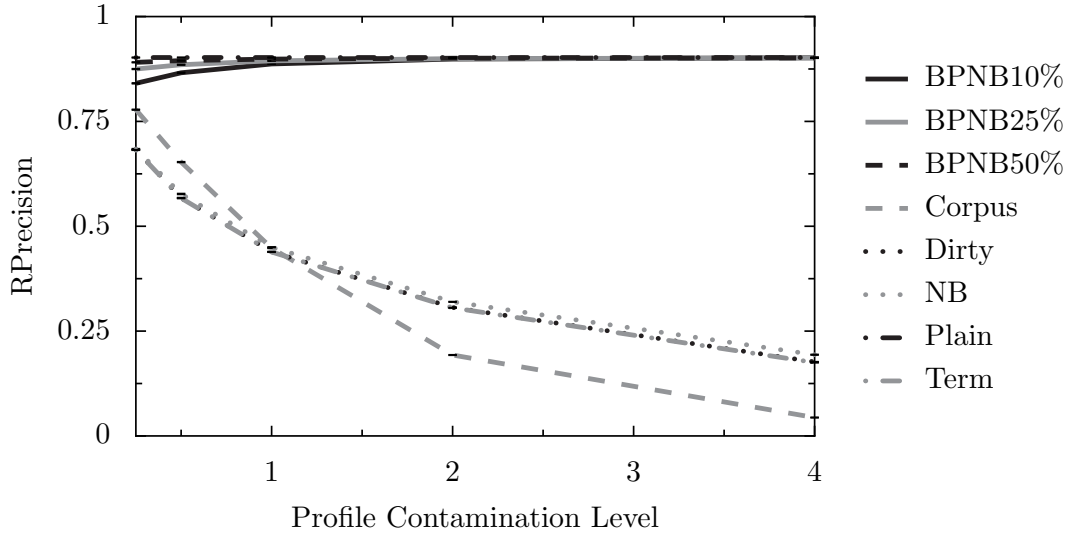
FIGURE A.5: Class Performance MAP



A.6.1: Erotica



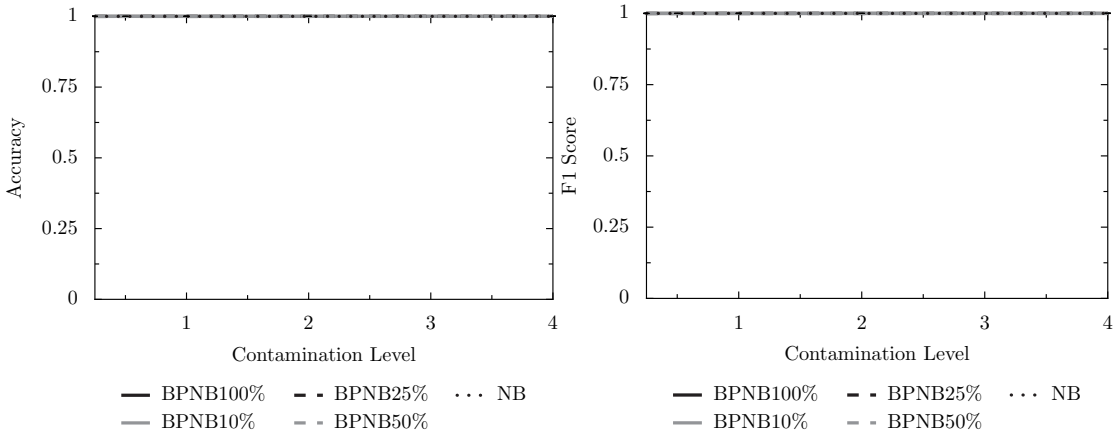
A.6.2: High-Energy Physics



A.6.3: Out-of-Context

FIGURE A.6: Class Performance RPrecision

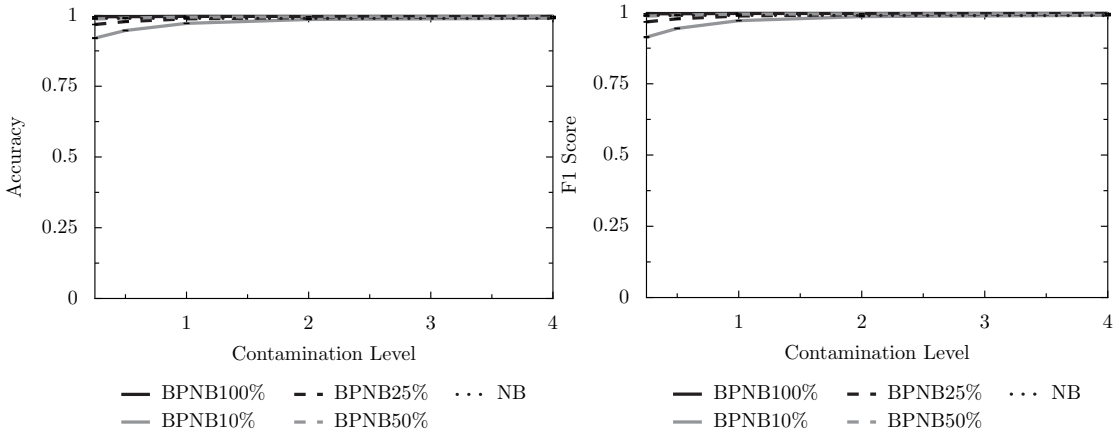
Figure A.7.1 shows the classification accuracy scores for the erotica dataset, and Figure A.7.2 shows the f1-score for the erotica dataset.



A.7.1: Accuracy A.7.2: F1 Score

FIGURE A.7: Classification Performance on Erotica Dataset

Figure A.8.1 shows the classification accuracy scores for the high-energy physics dataset, and Figure A.8.2 shows the f1-score for the high-energy physics dataset.

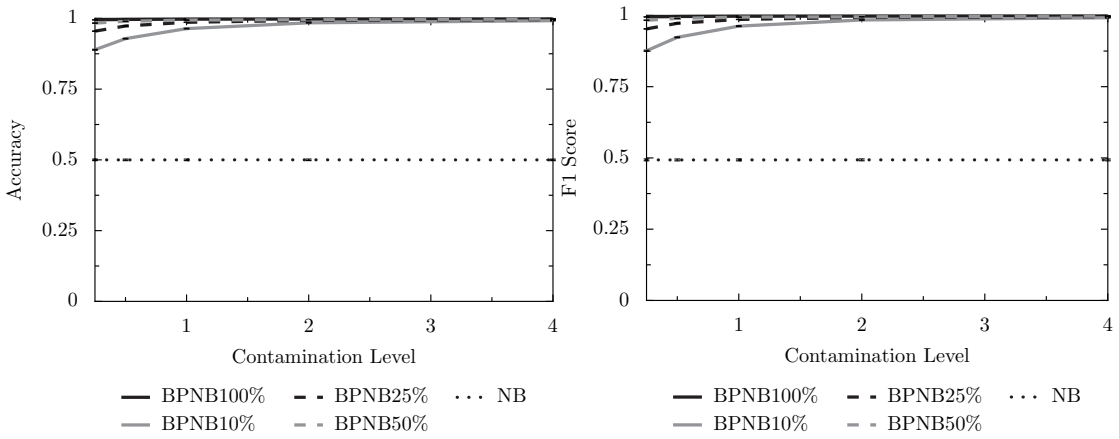


A.8.1: Accuracy A.8.2: F1 Score

FIGURE A.8: Classification Performance on High-Energy Physics Dataset

Figure A.9.1 shows the classification accuracy scores for the out-of-context dataset, and Figure A.9.2 shows the f1-score for the out-of-context dataset.

A.3 RPrecision Results



A.9.1: Accuracy

A.9.2: F1 Score

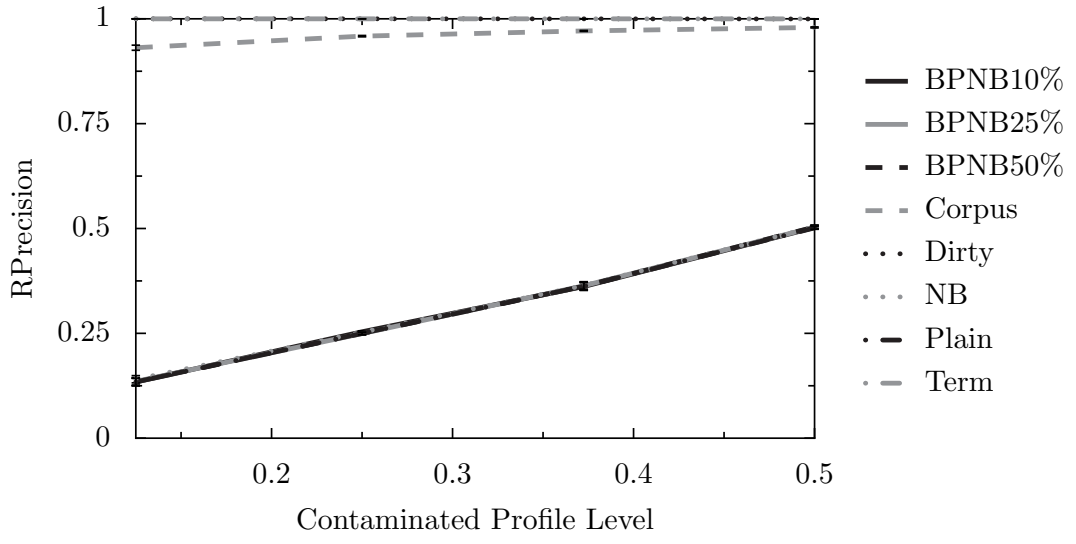
FIGURE A.9: Classification Performance on Out-of-Context Dataset

Figure A.10.1, Figure A.10.2, and Figure A.10.3 show the privacy-preservation RPrecision results for the erotica, high-energy physics, and out-of-context datasets respectively.

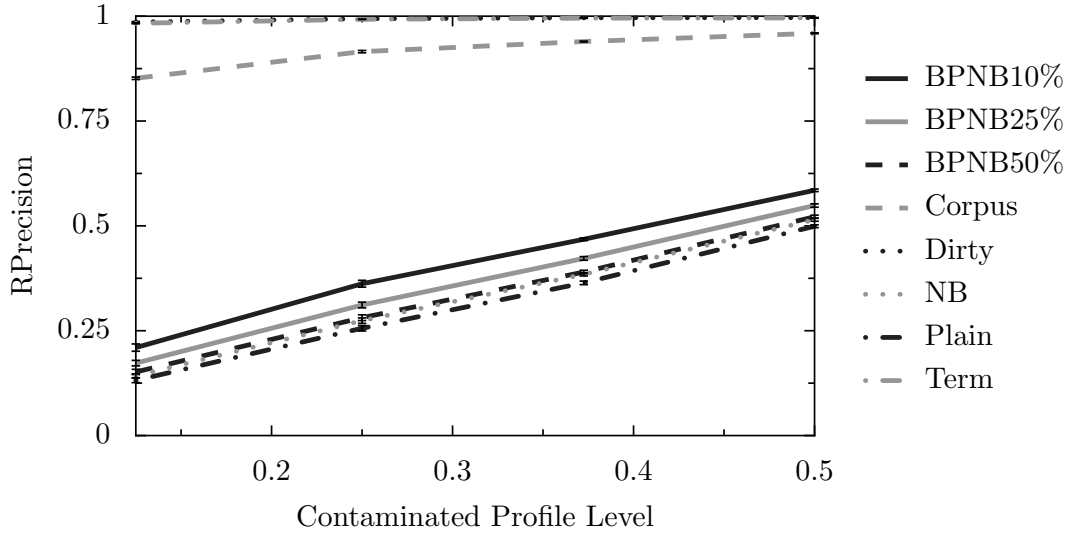
Figure A.11.1, Figure A.11.2, and Figure A.11.3 show the performance RPrecision results for the erotica, high-energy physics, and out-of-context datasets respectively.

Figure A.12.1, Figure A.12.2, and Figure A.12.3 show the class-based performance RPrecision results for the erotica, high-energy physics, and out-of-context datasets respectively.

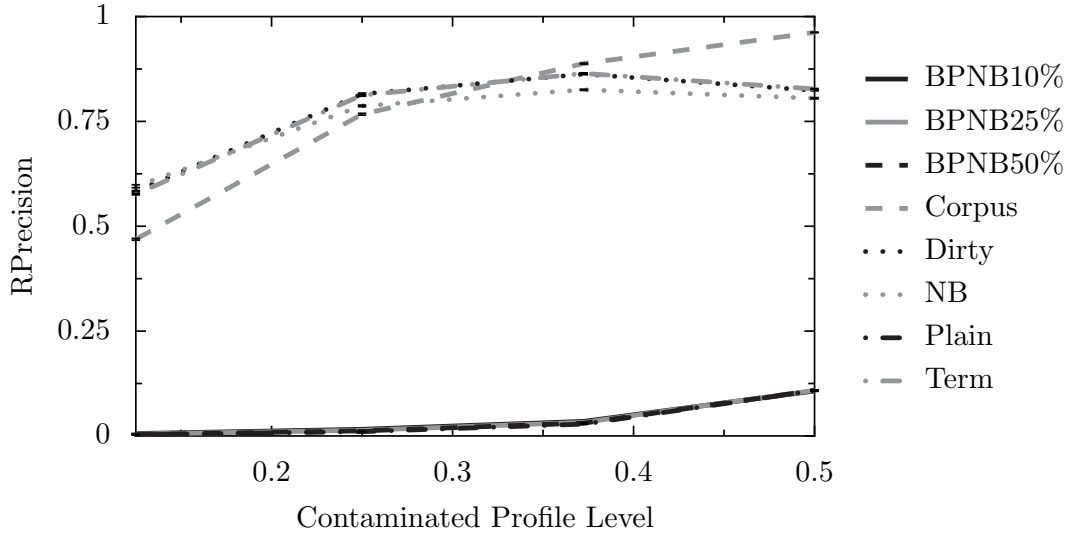
A.4 Results on small datasets



A.10.1: Erotica

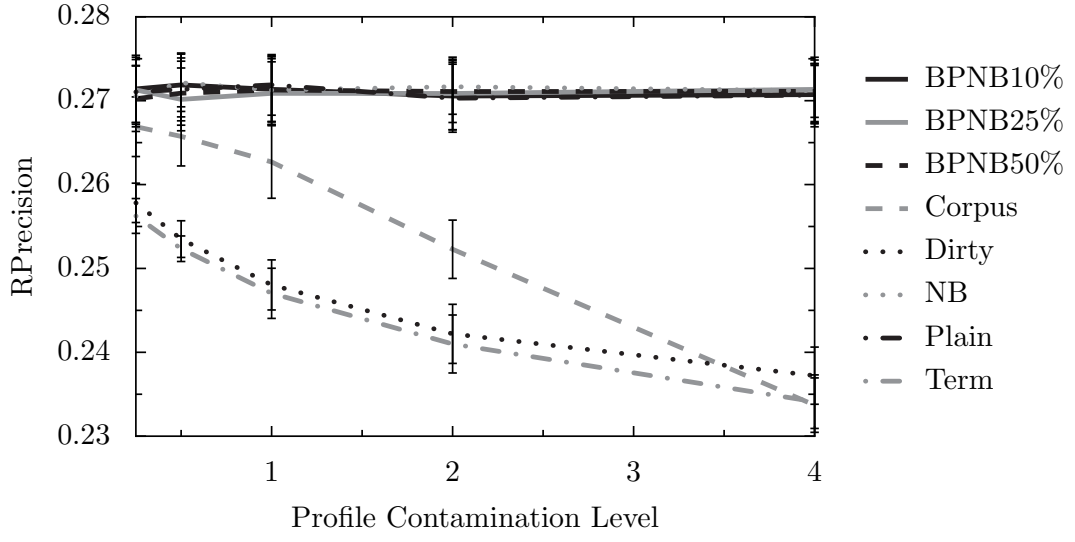


A.10.2: High-Energy Physics

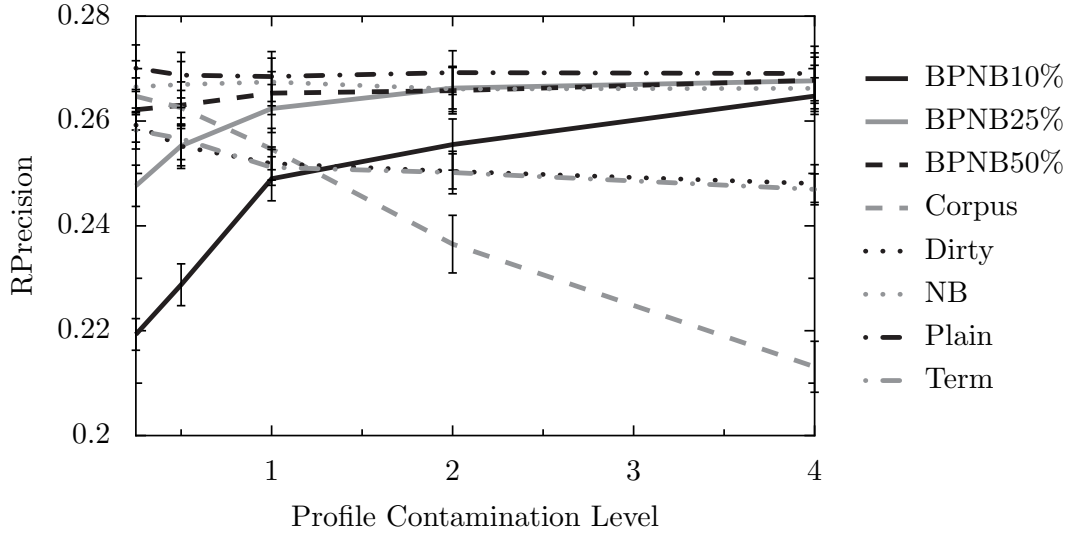


A.10.3: Out-of-Context

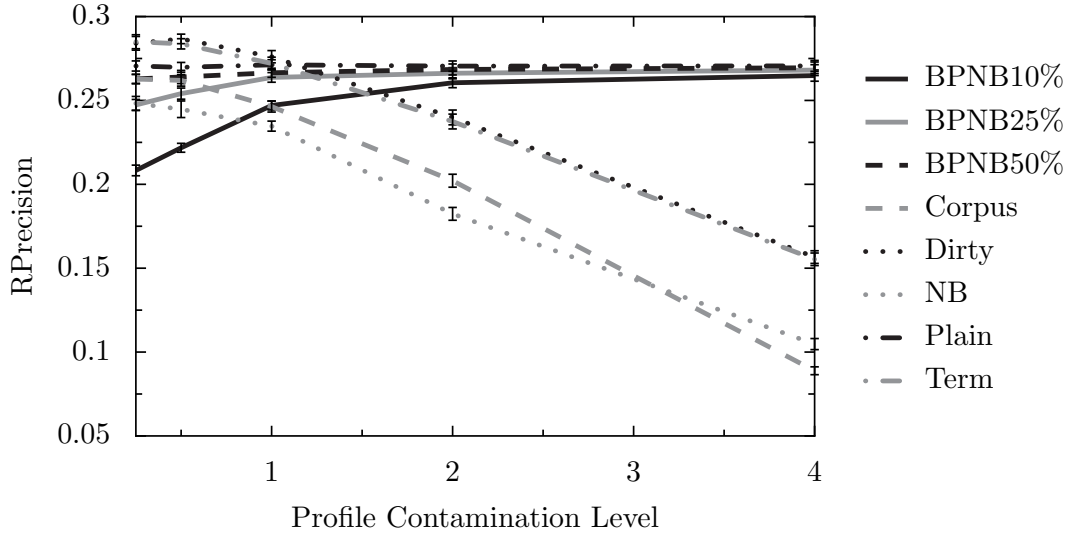
FIGURE A.10: Privacy-preservation RPrecision



A.11.1: Erotica

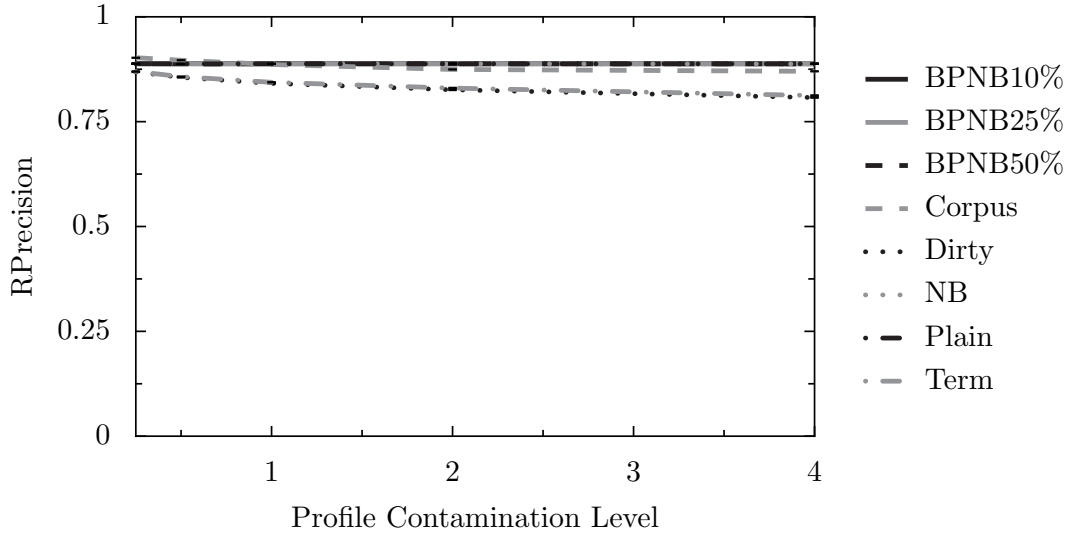


A.11.2: High-Energy Physics

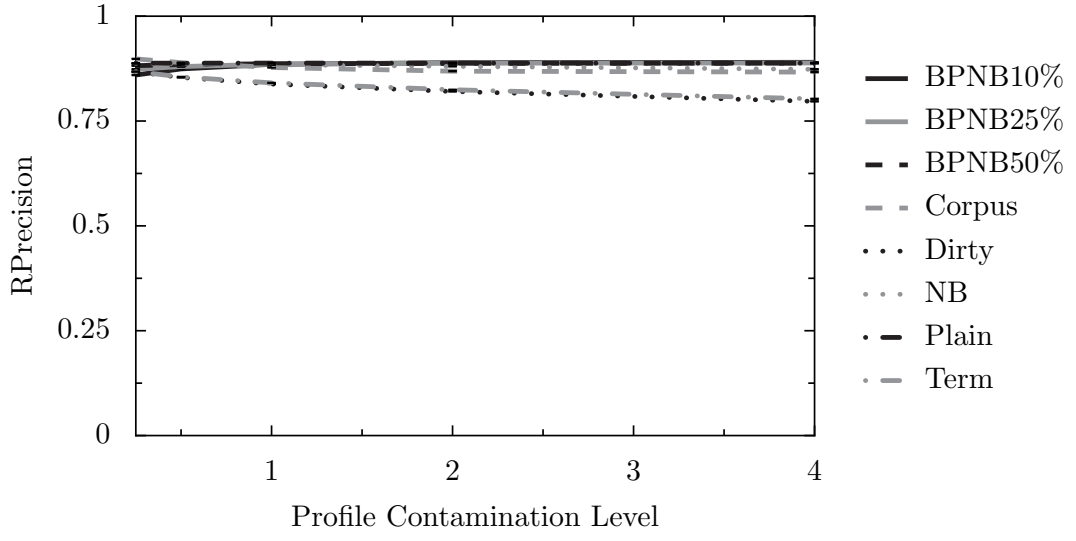


A.11.3: Out-of-Context

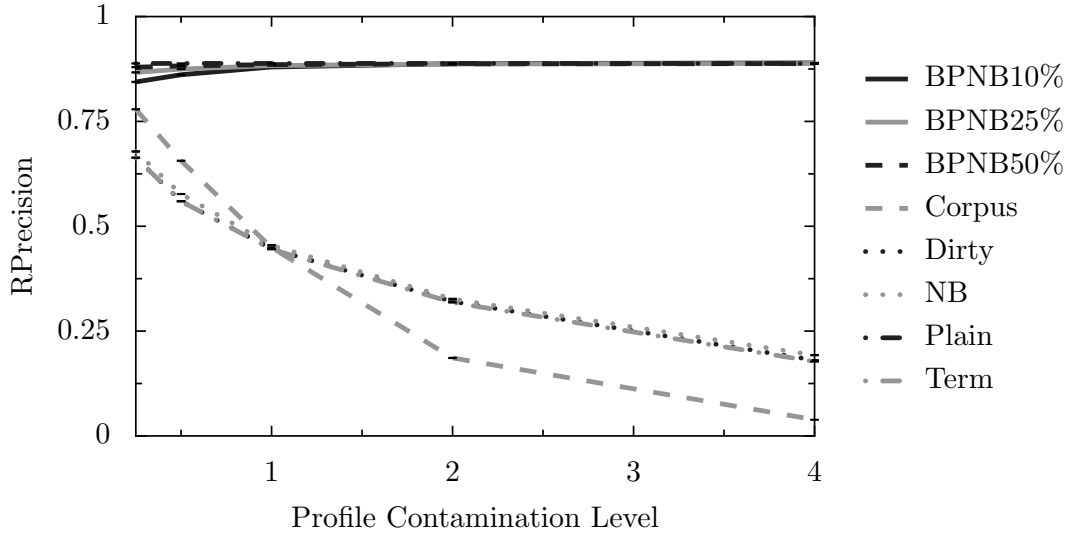
FIGURE A.11: Performance RPrecision



A.12.1: Erotica



A.12.2: High-Energy Physics



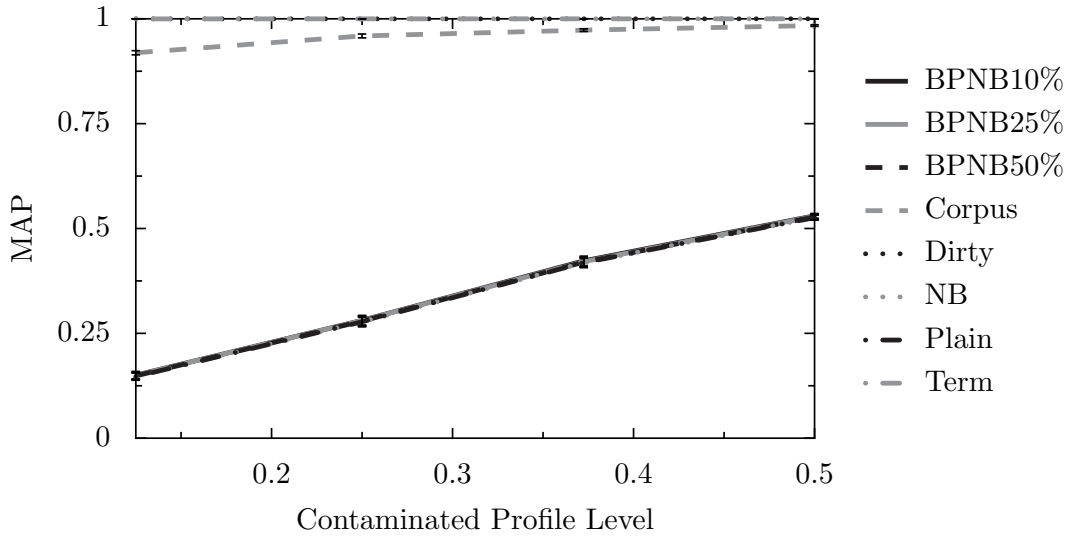
A.12.3: Out-of-Context

FIGURE A.12: Class Performance RPrecision

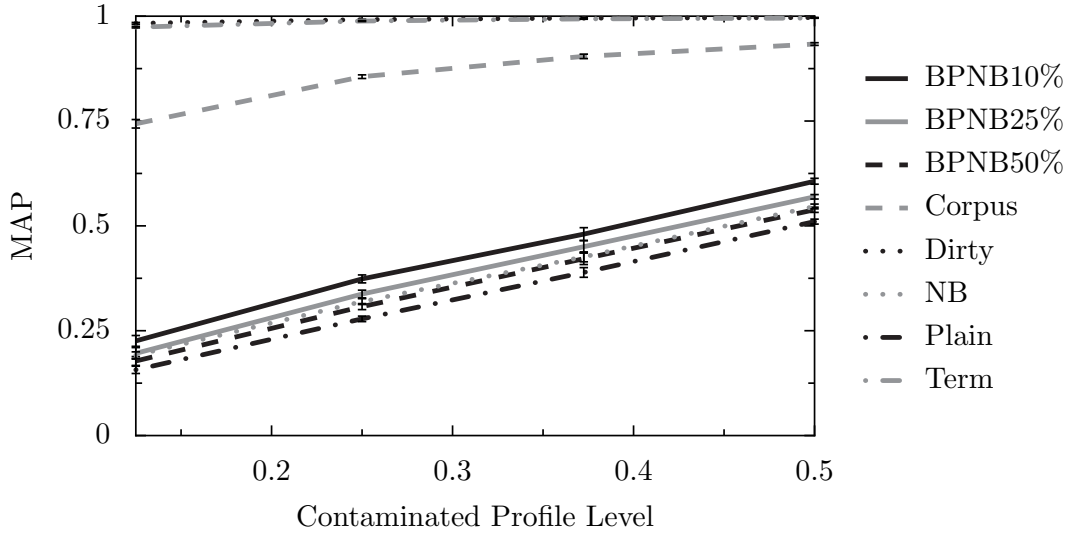
In this section the results obtained using the small datasets with a rank of 100 are given. Figure A.13.1, Figure A.13.2, and Figure A.13.3 show the privacy-preservation RPrecision results for the erotica, high-energy physics, and out-of-context datasets respectively. Figure A.14.1, Figure A.12.2, and Figure A.12.3 show the privacy-preservation RPrecision results for the erotica, high-energy physics, and out-of-context datasets respectively.

Figure A.15.1, Figure 9.11.2, and Figure 9.11.3 show the class-based performance RPrecision results for the erotica, high-energy physics, and out-of-context datasets respectively. Figure A.16.1, Figure A.12.2, and Figure A.12.3 show the class-based performance RPrecision results for the erotica, high-energy physics, and out-of-context datasets respectively.

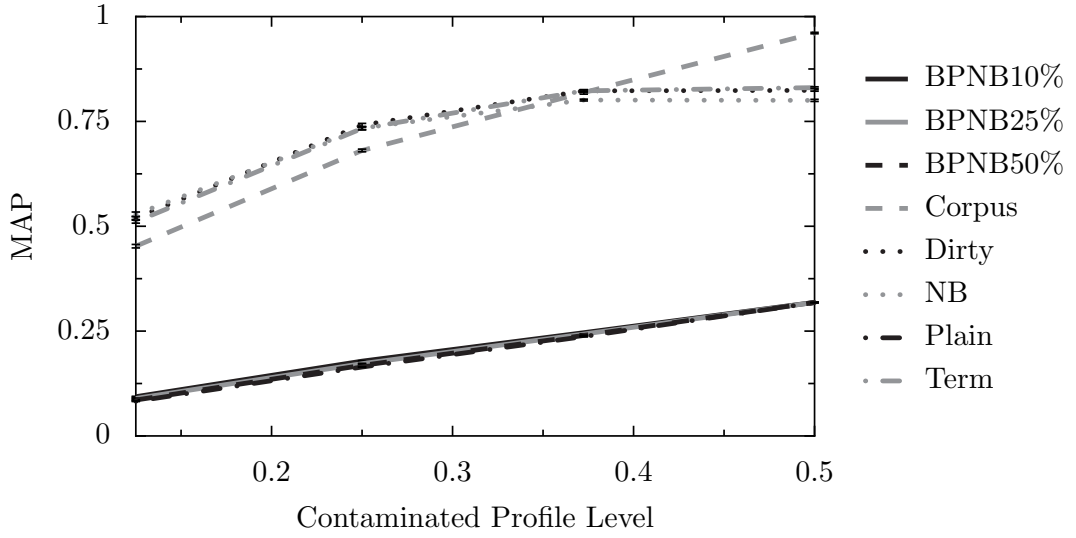
Figure A.17.1, Figure 9.11.2, and Figure 9.11.3 show the class-based performance RPrecision results for the erotica, high-energy physics, and out-of-context datasets respectively. Figure A.18.1, Figure A.12.2, and Figure A.12.3 show the class-based performance RPrecision results for the erotica, high-energy physics, and out-of-context datasets respectively.



A.13.1: Erotica

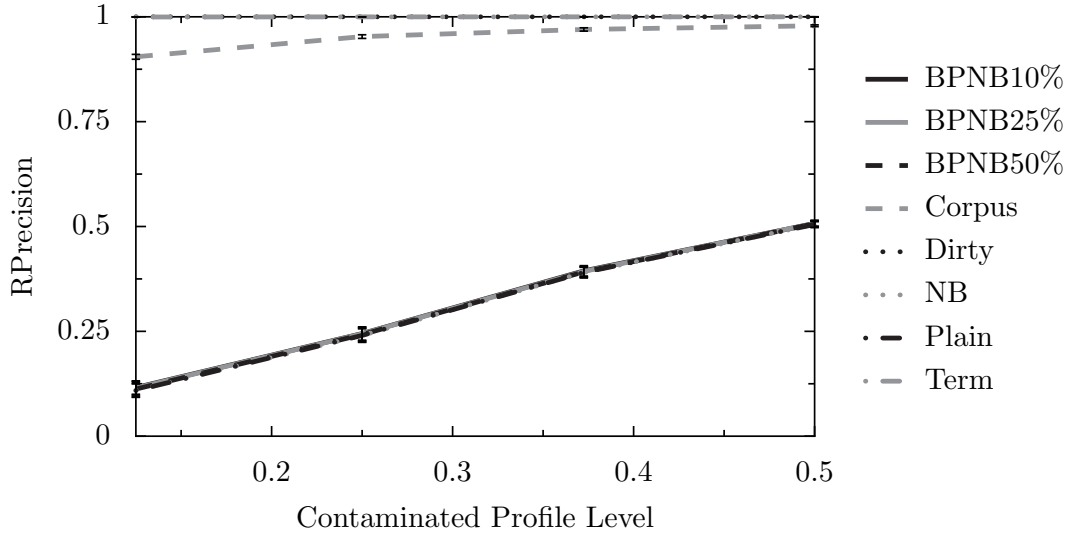


A.13.2: High-Energy Physics

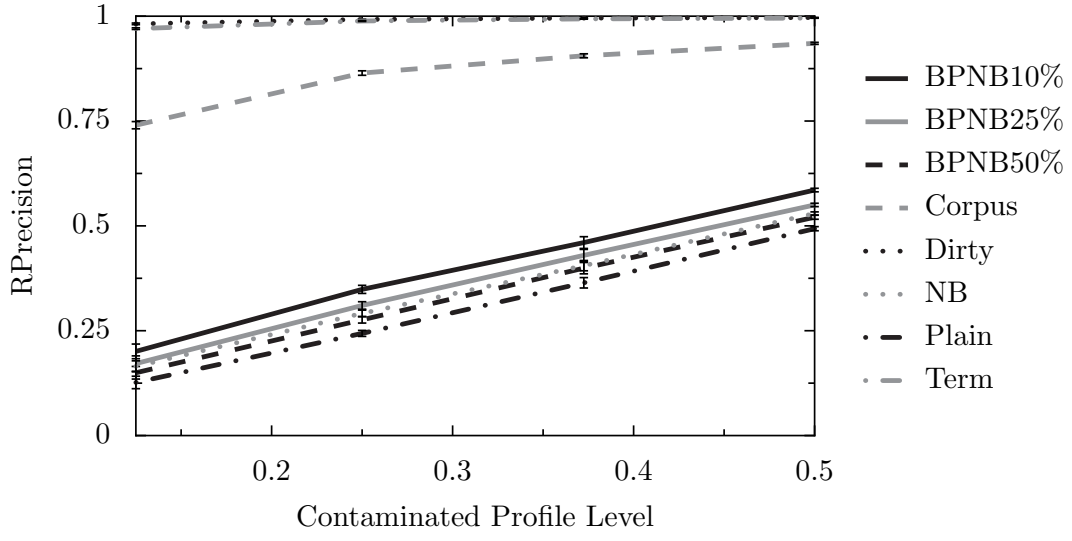


A.13.3: Out-of-Context

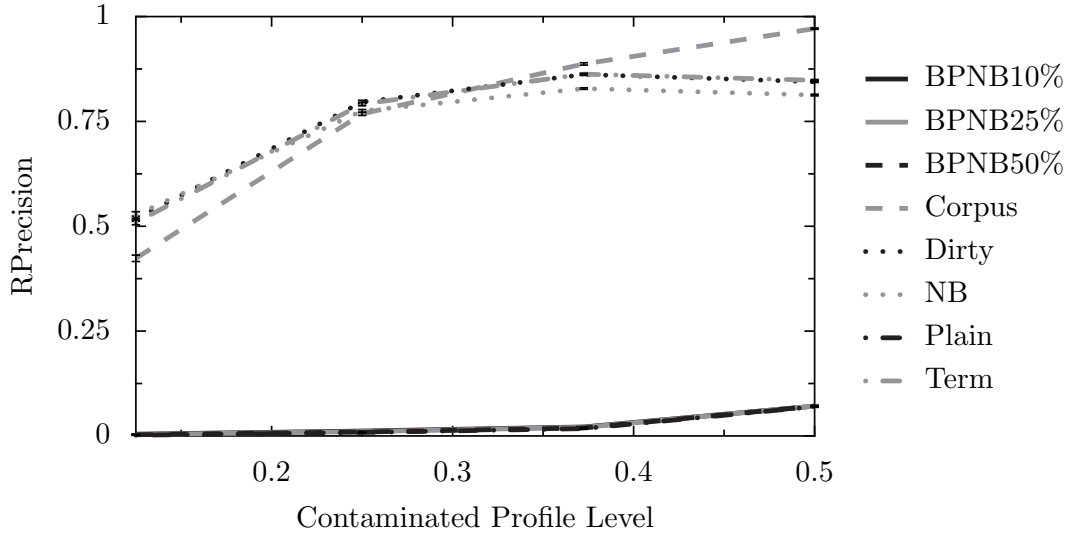
FIGURE A.13: Privacy-preservation MAP



A.14.1: Erotica

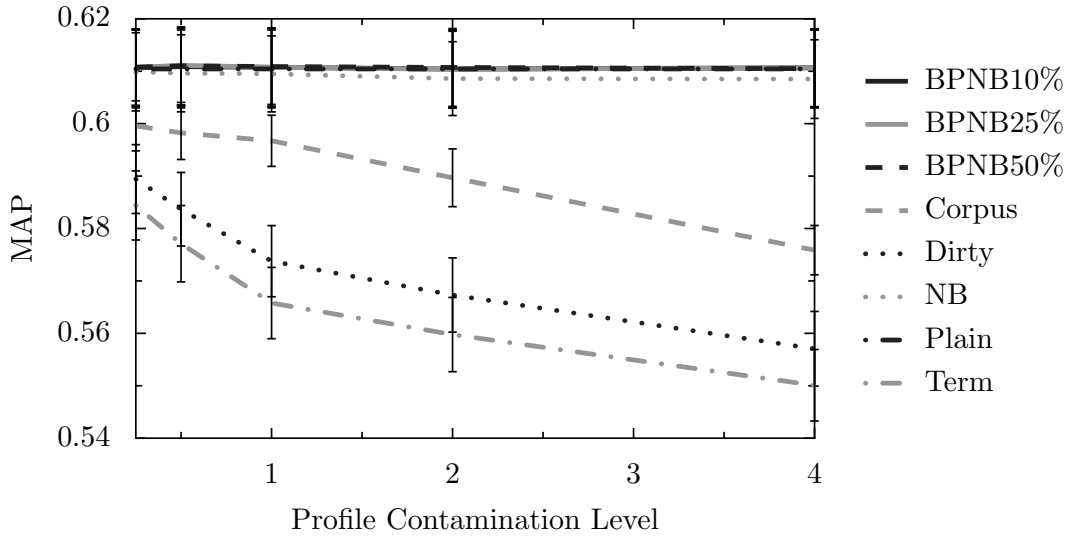


A.14.2: High-Energy Physics

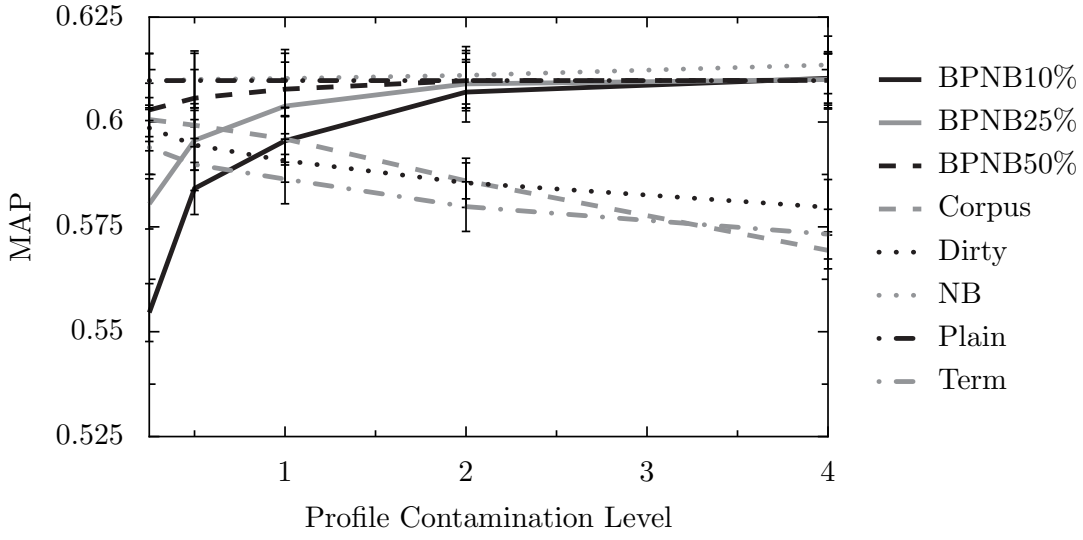


A.14.3: Out-of-Context

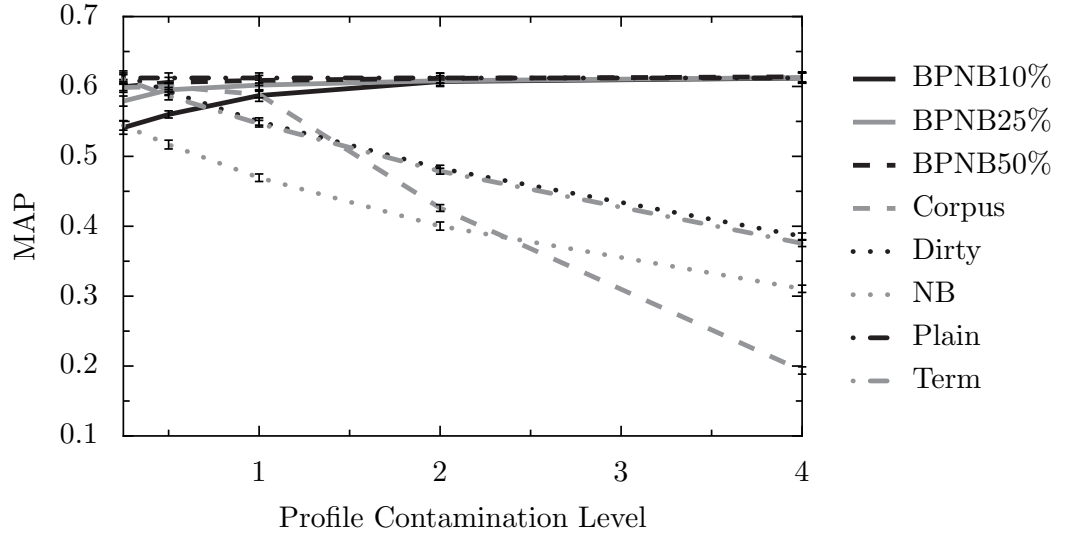
FIGURE A.14: Privacy-preservation RPrecision



A.15.1: Erotica

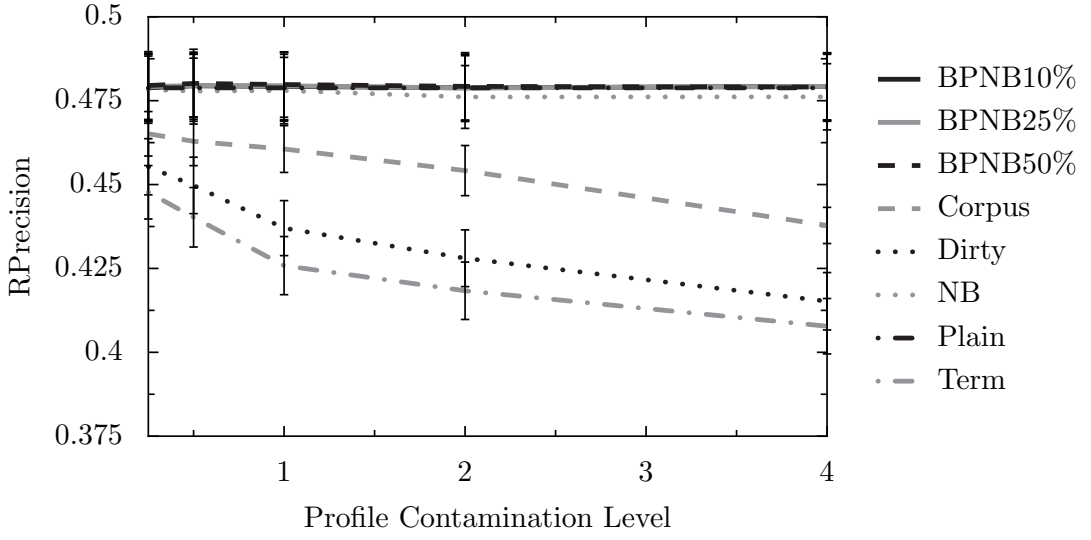


A.15.2: High-Energy Physics

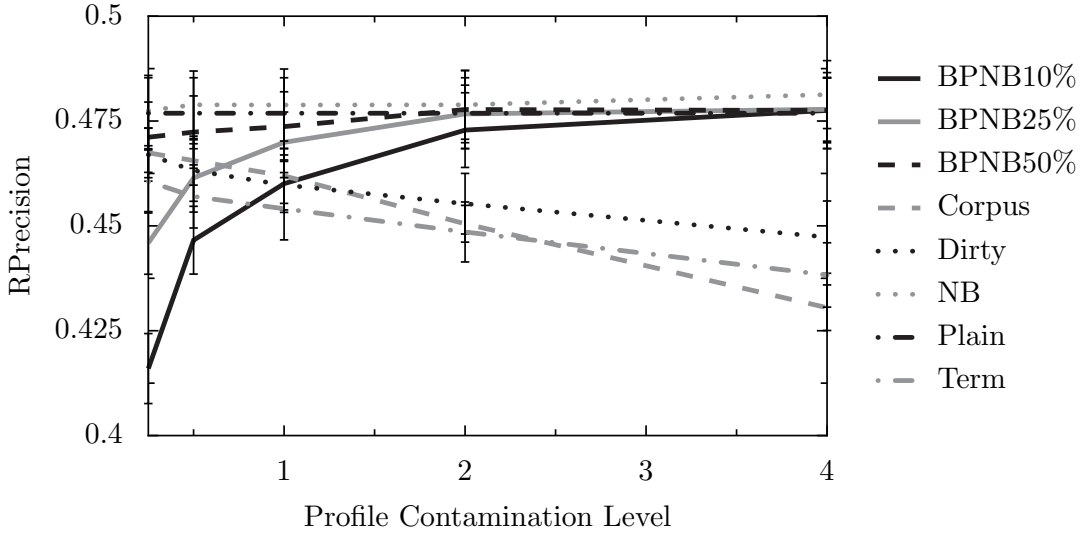


A.15.3: Out-of-Context

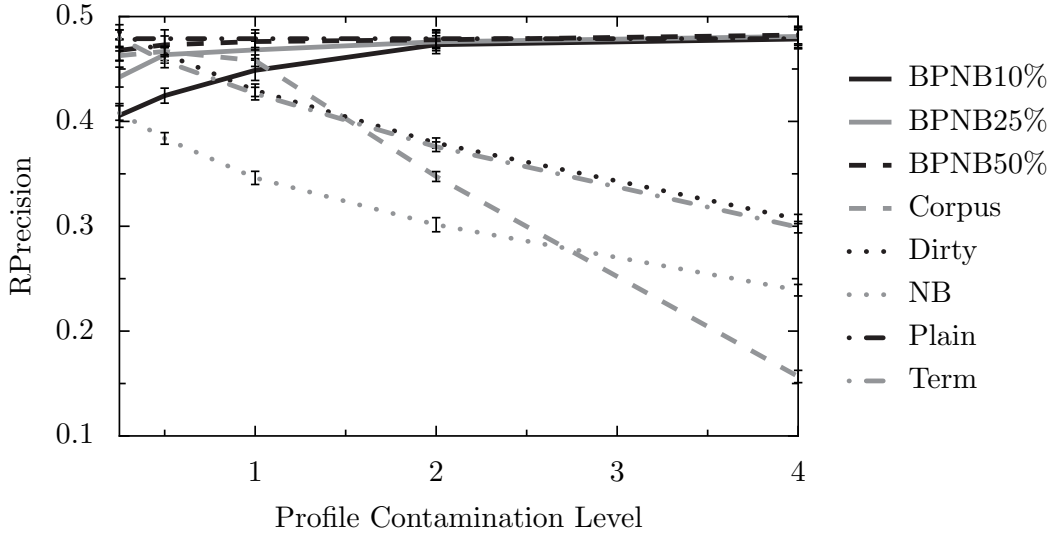
FIGURE A.15: Performance MAP



A.16.1: Erotica

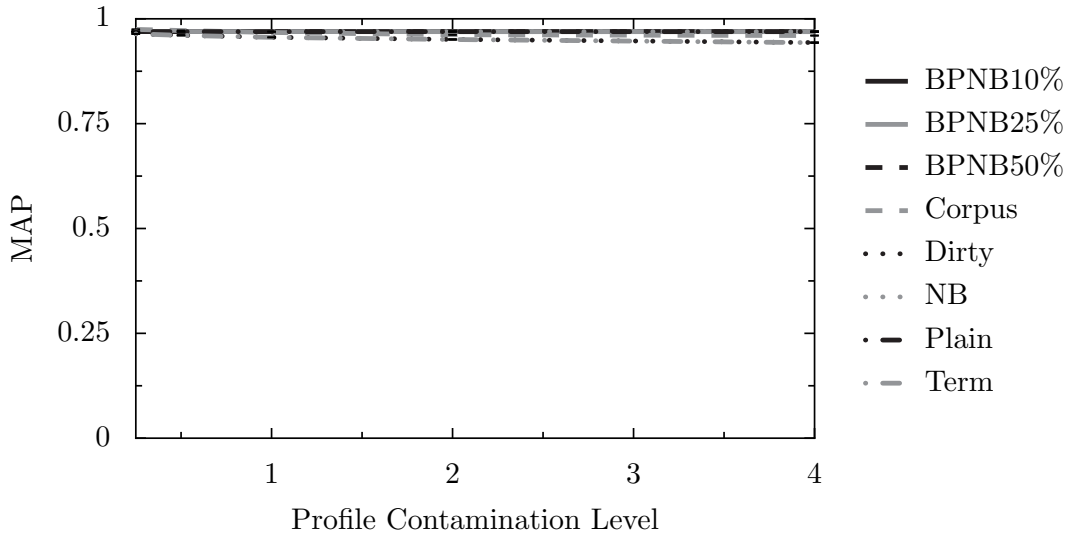


A.16.2: High-Energy Physics

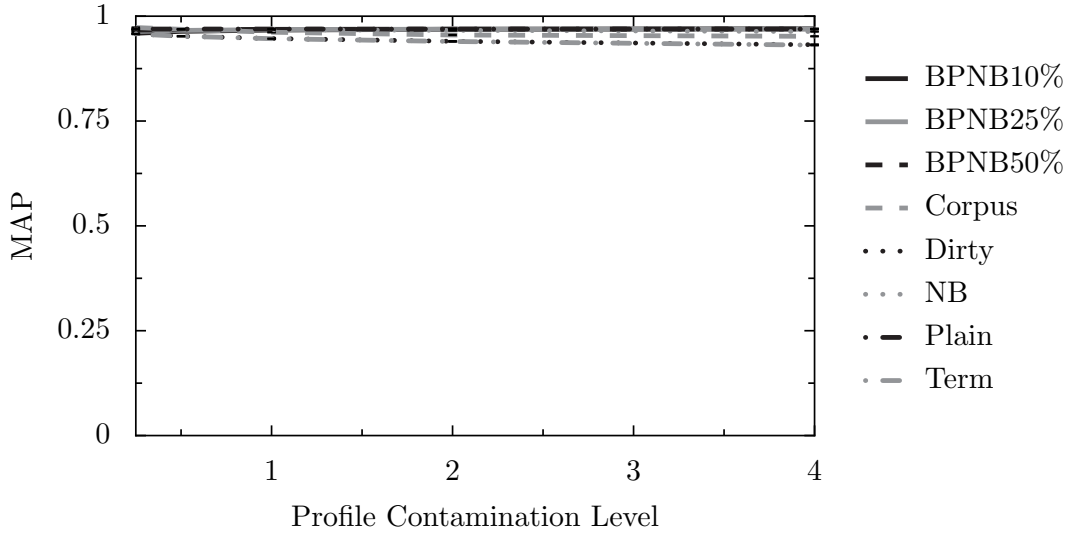


A.16.3: Out-of-Context

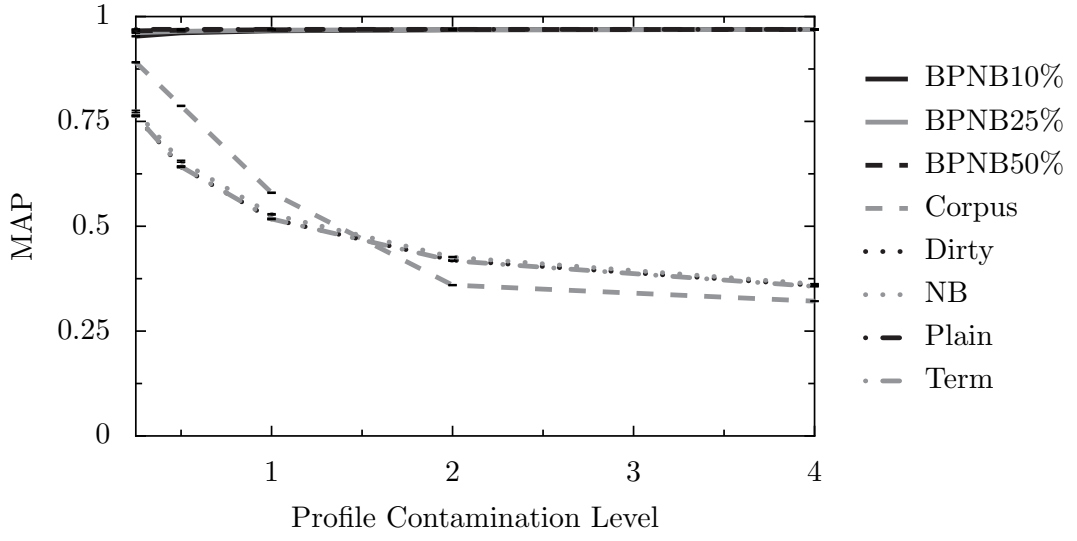
FIGURE A.16: Performance RPrecision



A.17.1: Erotica

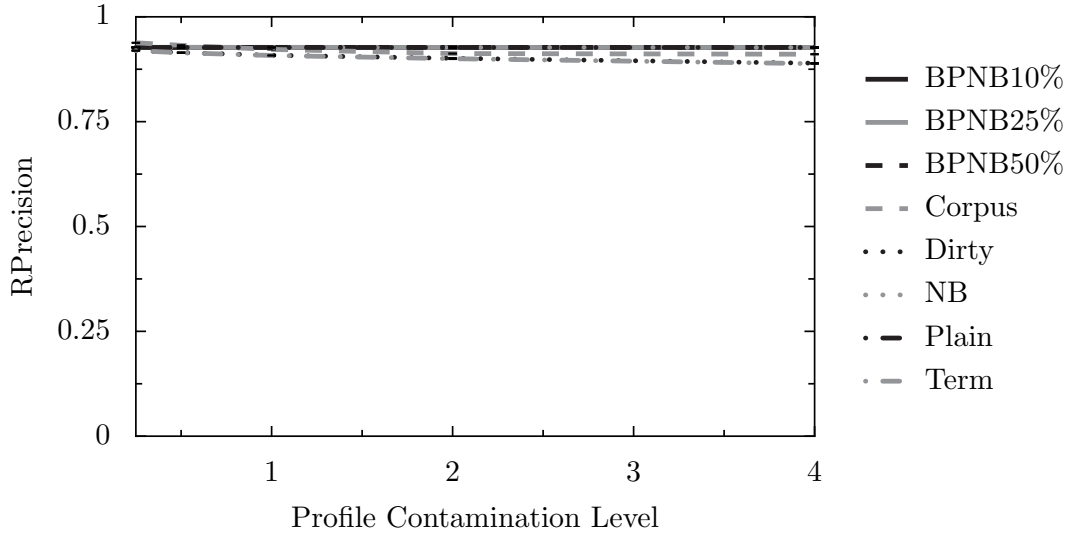


A.17.2: High-Energy Physics

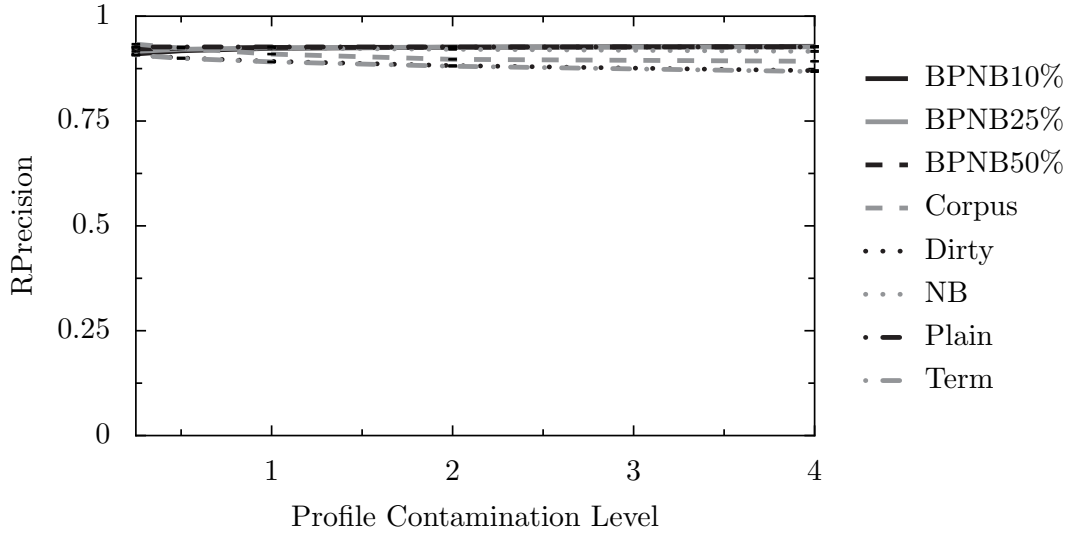


A.17.3: Out-of-Context

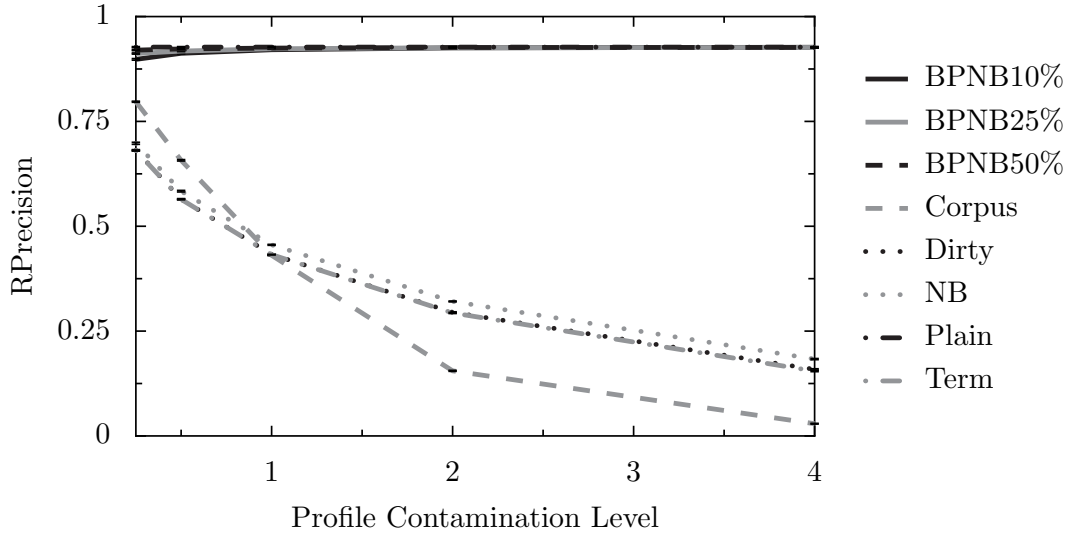
FIGURE A.17: Class Performance MAP



A.18.1: Erotica



A.18.2: High-Energy Physics



A.18.3: Out-of-Context

FIGURE A.18: Class Performance RPrecision

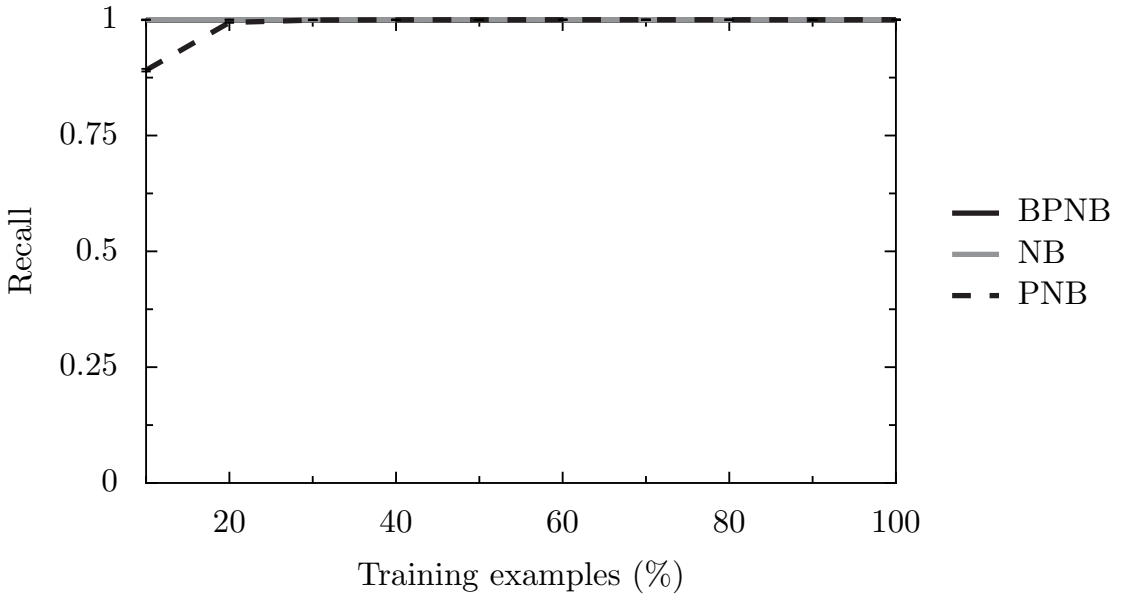
Appendix B

Additional PNB Parameter Results

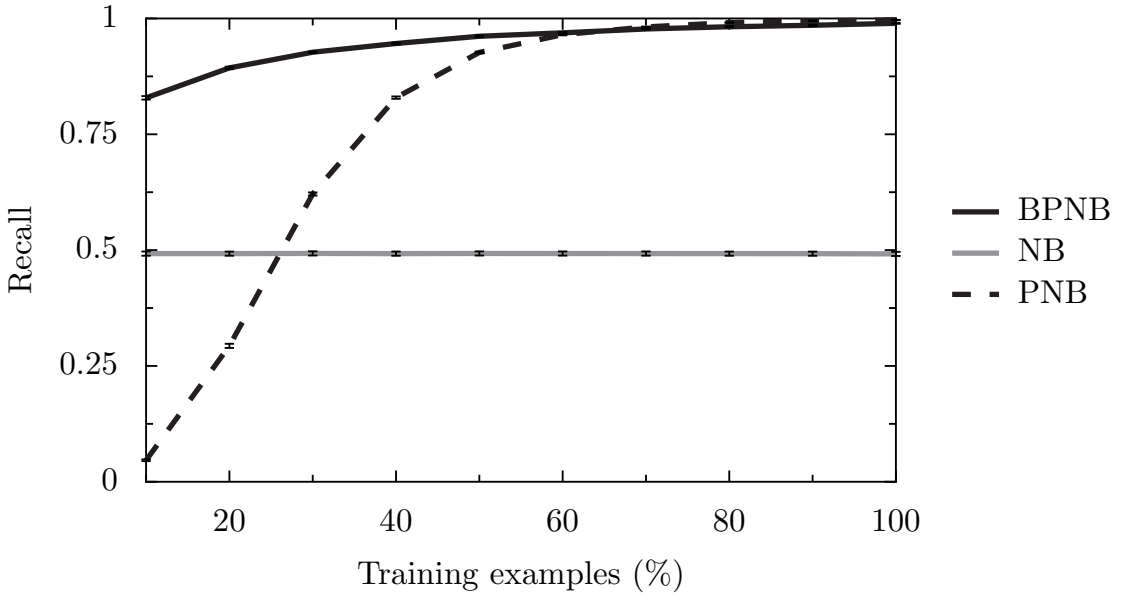
B.1 Bayesian Prior

Figure B.1.1 and Figure B.1.2 show the results of the experiments with Bayesian prior for the erotica and out-of-context datasets respectively.

B.2 Prior Multiplier



B.1.1: Erotica Dataset

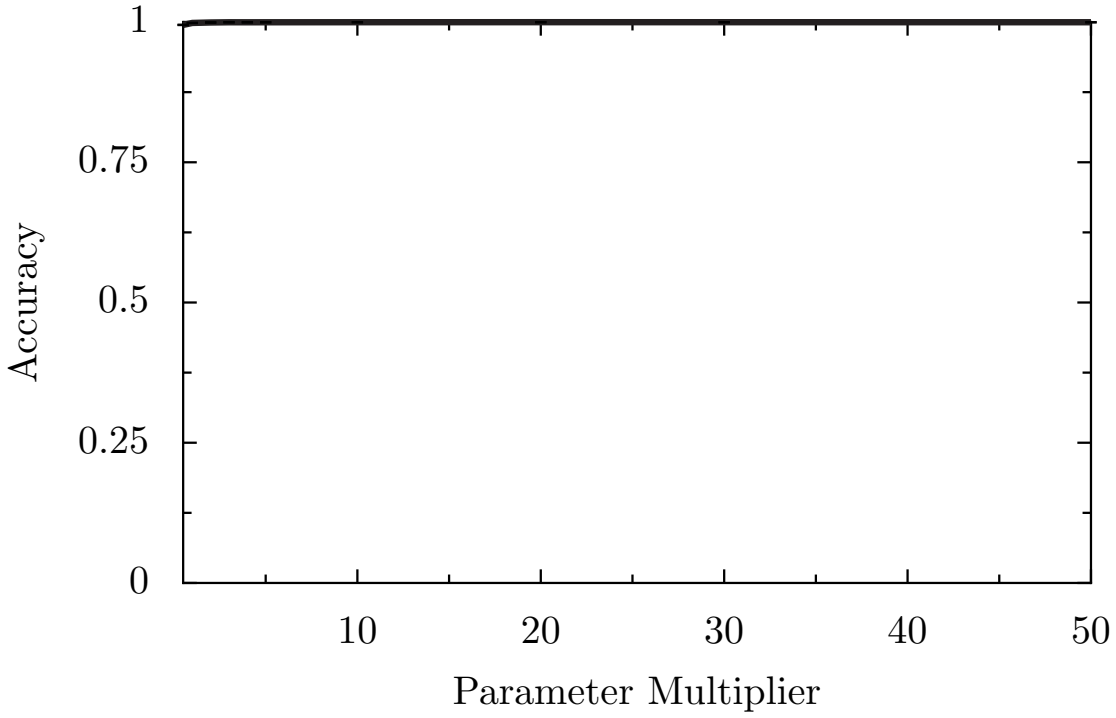


B.1.2: Out-Of-Context Dataset

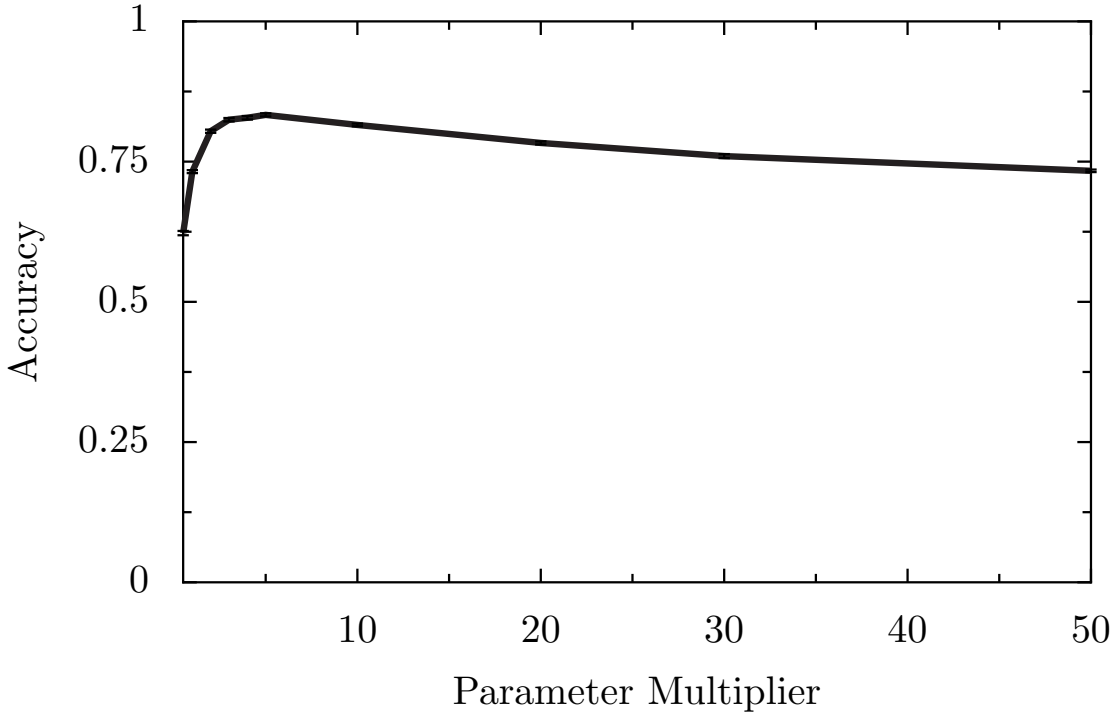
FIGURE B.1: Addition of a Bayesian Prior

Figure B.2.1 and Figure B.2.2 show the results of the experiments with a prior multiplier for the erotica and out-of-context datasets respectively.

B.3 Choice of P(1)



B.2.1: Erotica Dataset

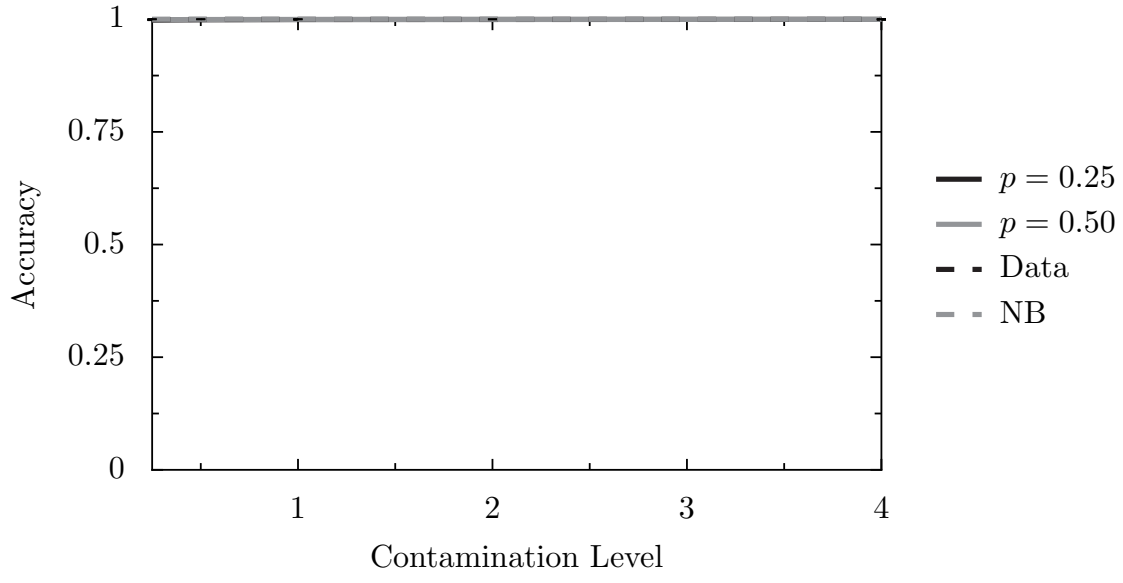


B.2.2: Out-Of-Context Dataset

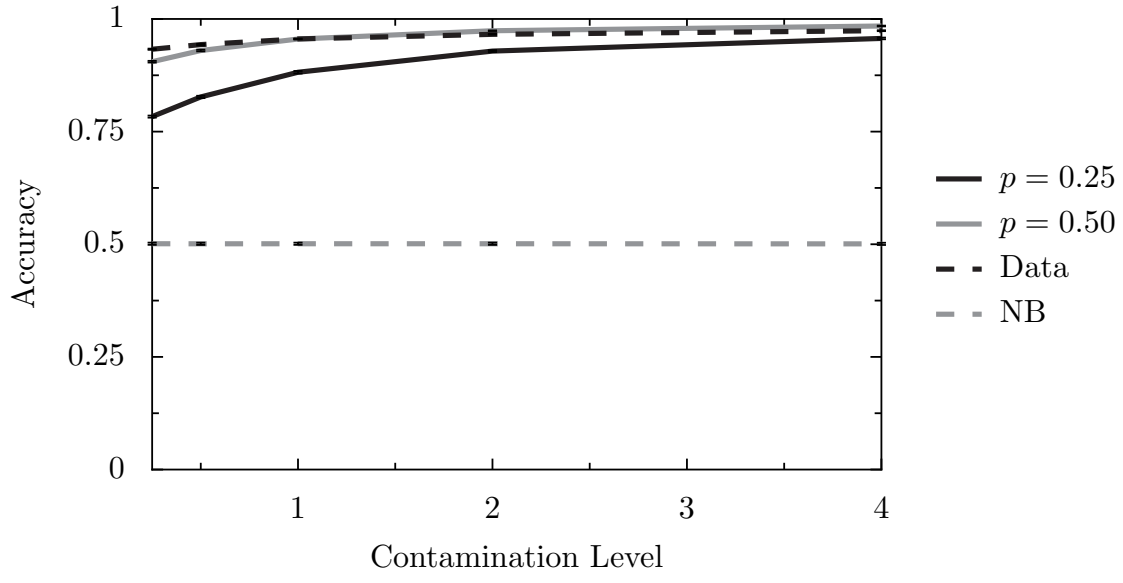
FIGURE B.2: Parameter Multipliers.

Figure B.3.1 and Figure B.3.2 show the results of the experiments with different values of $P(1)$ for the erotica and out-of-context datasets respectively.

B.4 Document Length Independence



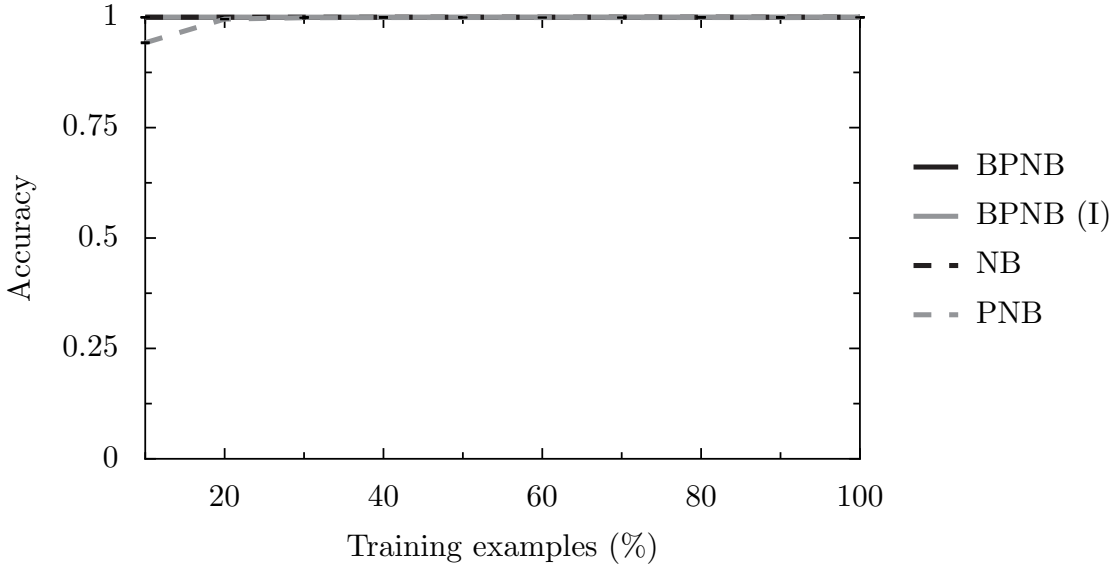
B.3.1: Erotica Dataset



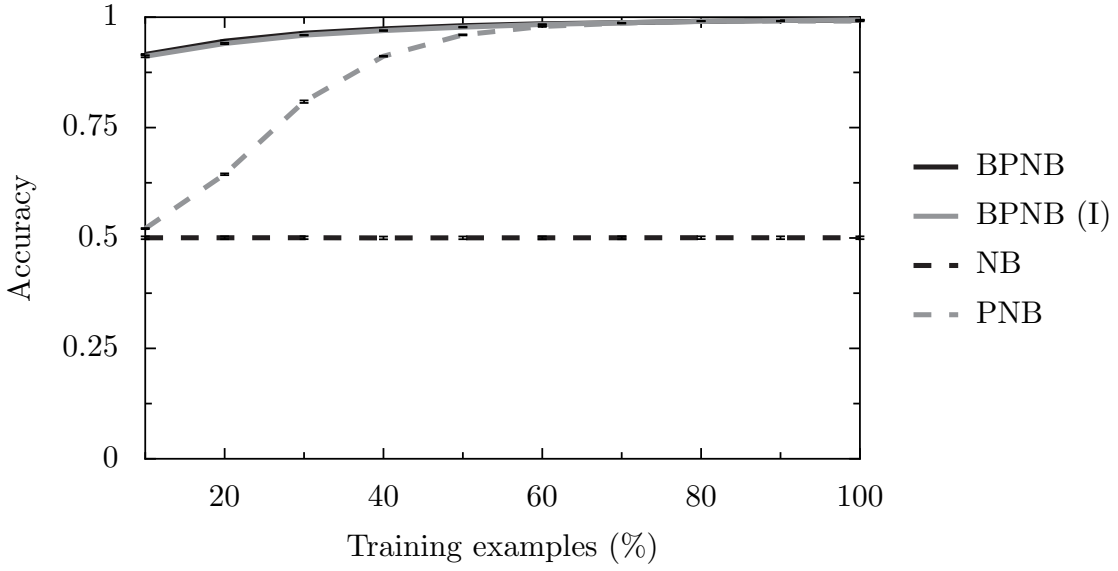
B.3.2: Out-Of-Context Dataset

FIGURE B.3: Choice of P(1).

Figure B.4.1 and Figure B.4.2 show the results of the experiments with document length class independence assumptions, for the erotica and out-of-context datasets respectively.



B.4.1: Erotica Dataset



B.4.2: Out-Of-Context Dataset

FIGURE B.4: Document Length Class Independence

Bibliography

Gediminas Adomavicius, Ramesh Sankaranarayanan, Shahana Sen, and Alexander Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Trans. Inf. Syst.*, 23(1):103–145, 2005. ISSN 1046-8188.

Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17:734–749, 2005.

Dakshi Agrawal and Charu C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '01, pages 247–255, New York, NY, USA, 2001. ACM. ISBN 1-58113-361-8.

Michael Arrington. AOL proudly releases massive amounts of private data. `http://techcrunch.com/2006/08/06/aol-proudly-releases-massive-amounts-of-user-search-data/`, August 2006. Retrieved on Monday 17th October 2011.

Yoav Shoham Marko Balabanovic. Fab: content-based, collaborative recommendation. In *Communications of the ACM archive*, volume 40, pages 66–72, Miami, Florida, USA, 1997.

Jamie Banford, Alisdair McDiarmid, and James Irvine. Estimating the strength of ties in communication networks with a small number of users. In *Proceedings of the 2010 6th International Conference on Wireless and Mobile Communications*, ICWMC '10, pages 191–195, Washington, DC, USA, 2010a. IEEE Computer Society. ISBN 978-0-7695-4182-2.

Jamie Banford, Alisdair McDiarmid, and James Irvine. Foaf: improving detected social network accuracy. In *Proceedings of the 12th ACM international conference adjunct papers on Ubiquitous computing*, Ubicomp '10 Adjunct, pages 393–394, New York, NY, USA, 2010b. ACM. ISBN 978-1-4503-0283-8.

Thomas Barnard and Adam Prügel-Bennett. Experiments in bayesian recommendation, January 2011a.

Thomas Barnard and Adam Prügel-Bennett. Privacy-preserving profiling. In *DMIN2011*, July 2011b.

Elisa Bertino, Igor Nai Fovino, and Loredana Parasiliti Provenza. A framework for evaluating privacy preserving data mining algorithms*. *Data Min. Knowl. Discov.*, 11:121–154, September 2005. ISSN 1384-5810.

D. Billsus and M. Pazzani. Learning collaborative information filters. In *Proc. Intl Conf. Machine Learning*, 1998.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. In Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani, editors, *NIPS*, pages 601–608. MIT Press, 2001.

John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *UAI*, pages 43–52. Morgan Kaufmann, 1998.

J. Brown, P. and J. F. Jones, G. Context-aware retrieval: Exploring a new environment for information retrieval and information filtering. *Personal Ubiquitous Comput.*, 5 (4):253–263, 2001. ISSN 1617-4909.

Robin Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002. ISSN 0924-1868.

Kevin Chai, Vidyasagar Potdar, and Elizabeth Chang. A survey of revenue models for current generation social software's systems. In *Proceedings of the 2007 international conference on Computational science and its applications - Volume Part III*, ICCSA'07, pages 724–738, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 3-540-74482-5, 978-3-540-74482-5.

Olivier Chapelle, Bernhard Schlkopf, and Alexander Zien. *Semi-Supervised Learning*. The MIT Press, 1st edition, 2010. ISBN 0262514125, 9780262514125.

Annie Chen. Context-aware collaborative filtering system: Predicting the user's preference in the ubiquitous computing environment. In Thomas Strang and Claudia Linnhoff-Popien, editors, *LoCA*, volume 3479 of *Lecture Notes in Computer Science*, pages 244–253. Springer, 2005. ISBN 3-540-25896-5.

Yeon Choi, Joon, Seok Song, Hee, Hie Kim, and Soung. MCORE: a context-sensitive recommendation system for the mobile web. *Expert Systems*, 24(1):32–46, February 2007. ISSN 0266-4720.

Alexandre de Spindler, Moira C. Norrie, and Michael Grossniklaus. Collaborative filtering based on opportunistic information sharing in mobile ad-hoc networks. In Robert Meersman and Zahir Tari, editors, *OTM Conferences (1)*, volume 4803 of *Lecture Notes in Computer Science*, pages 408–416. Springer, 2007. ISBN 978-3-540-76846-3.

Bernhard Debatin, Jennette P. Lovejoy, Ann-Kathrin Horn, and Brittany N. Hughes. Facebook and online privacy: Attitudes, behaviors, and unintended consequences. *Journal of Computer-Mediated Communication*, 15(1):83–108, 2009. ISSN 1083-6101.

Morris H. DeGroot. *Optimal Statistical Decisions*. McGraw-Hill Book Company, 1970.

F Denis, Rmi Gilleron, and M Tommasi. Text classification from positive and unlabeled examples. *IPMU02 9th International Conference on Information Processing and Management of Uncertainty in Knowledge Based Systems*, pages 1927–1934, 2002.

Francois Denis, Anne Laurent, Rmi Gilleron, and Marc Tommasi. Text classification and co-training from positive and unlabeled examples. In *Proceedings of the ICML 2003 Workshop: The Continuum from Labeled to Unlabeled Data*, pages 80–87, 2003.

Albrecht Enders, Harald Hungenberg, Hans-Peter Denker, and Sebastian Mauch. The long tail of social networking.: Revenue models of social networking sites. *European Management Journal*, 26(3):199–211, 2008. ISSN 0263-2373.

Christopher Fox. A stop list for general text. *SIGIR Forum*, 24:19–21, September 1989. ISSN 0163-5840.

Hugh Glaser, Ian Millard, and Les Carr. RKBExplorer: Repositories, linked data and research support. In *Eprints User Group, Open Repositories 2009*, May 2009.

David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35(12):61–70, 1992. ISSN 0001-0782.

Gustavo González, Josep Lluís de la Rosa, and Miquel Montaner. Embedding emotional context in recommender systems. In David Wilson and Geoff Sutcliffe, editors, *FLAIRS Conference*, pages 454–459. AAAI Press, 2007. ISBN 978-1-57735-319-5.

Nathaniel Good, J. Ben Schafer, Joseph A. Konstan, Al Borchers, Badrul Sarwar, Jon Herlocker, and John Riedl. Combining collaborative filtering with personal agents for better recommendations. In *AAAI '99/IAAI '99: Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence*, pages 439–446, Menlo Park, CA, USA, 1999. American Association for Artificial Intelligence. ISBN 0-262-51106-1.

Saikat Guha, Bin Cheng, and Paul Francis. Challenges in measuring online advertising systems. In *Proceedings of the 10th annual conference on Internet measurement*, IMC '10, pages 81–87, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0483-2.

David J. Hand and Keming Yu. Idiot's bayes—not so stupid after all? *International Statistical Review*, 69(3):385–398, 2001.

Conor Hayes and Padraig Cunningham. Context boosting collaborative recommendations. *Knowledge-Based Systems*, 17(2-4):131–138, 2004.

Martin Helmhout, Craig Saunders, Allan Tomlinson, John A. MacDonald, Jame M. Irvine, and Alisdair McDiarmid. Instant knowledge: a secure mobile context-aware distributed recommender system. In *ICT-MobileSummit 2009*, Santander, June 2009. IEEE.

Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 230–237, New York, NY, USA, 1999. ACM. ISBN 1-58113-096-1.

Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1): 5–53, 2004. ISSN 1046-8188.

Thomas Hofmann. Probabilistic latent semantic indexing. In *SIGIR*, pages 50–57. ACM, 1999.

Thomas Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 42(1/2):177–196, 2001.

Tzvetan Horozov, Nitya Narasimhan, and Venu Vasudevan. Using location for personalized POI recommendations in mobile environments. *Applications and the Internet, IEEE/IPSJ International Symposium on*, 0:124–129, 2006.

Thorsten Joachims. A probabilistic analysis of the rocchio algorithm with TFIDF for text categorization. In *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 143–151, 1997.

Peter Leitner and Thomas Grechenig. Social networking sphere: A snapshot of trends, functionalities and revenue models. In *IADIS International Conference on Web Based Communities*, pages 187–191, 2008.

Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, 2003. ISSN 1089-7801.

David Maltz and Kate Ehrlich. Pointing the way: active collaborative filtering. In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 202–209, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co. ISBN 0-201-84705-1.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008. ISBN 0-521-86571-9, 978-0-521-86571-5.

Steve Mansfield-Devine. Hacktivism: assessing the damage. *Network Security*, 2011(8): 5–13, 2011. ISSN 1353-4858.

Stuart E. Middleton, David C. De Roure, and Nigel R. Shadbolt. Capturing knowledge of user preferences: Ontologies in recommender systems. In *In Proceedings of the First International Conference on Knowledge Capture (K-CAP 2001), Oct 2001*, pages 100–107. ACM Press, 2001.

Bradley N. Miller, Joseph A. Konstan, and John Riedl. Pocketlens: Toward a personal recommender system. *ACM Trans. Inf. Syst.*, 22(3):437–476, 2004. ISSN 1046-8188.

Thomas P. Minka. Estimating a dirichlet distribution. Technical report, Microsoft, 2003.

Koji Miyahara and Michael J. Pazzani. Collaborative filtering with the simple bayesian classifier. In *In: Proceedings of the 6th Pacific Rim International Conference on Artificial Intelligence*, pages 679–689, 2000.

J. Pouwelse, P. Garbacki, J. Wang, A. Bakker, J. Yang, A. Iosup, D. H. J. Epema, M. Reinders, M. van Steen, and H. Sips. Tribler: A social-based peer-to-peer system. In *5th Int'l Workshop on Peer-to-Peer Systems (IPTPS'06)*, Santa Barbara, CA, February 2006.

Tim Reichling and Volker Wulf. Expert recommender systems in practice: evaluating semi-automatic profile generation. In *Proceedings of the 27th international conference on Human factors in computing systems*, CHI '09, pages 59–68, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-246-7.

Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *CSCW '94: Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186, New York, NY, USA, 1994. ACM. ISBN 0-89791-689-1.

Paul Resnick and Hal R. Varian. Recommender systems. *Commun. ACM*, 40(3):56–58, 1997. ISSN 0001-0782.

Víctor Robles, Pedro Larrañaga, José M. Peña Sánchez, Oscar Marbán, F. Javier Crespo, and María S. Pérez. Collaborative filtering using interval estimation naïve bayes. In Ernestina Menasalvas Ruiz, Javier Segovia, and Piotr S. Szczepaniak, editors, *AWIC*, pages 46–53. Springer, 2003.

G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18:613–620, November 1975. ISSN 0001-0782.

Gerard Salton and Chris Buckley. Term weighting approaches in automatic text retrieval. Technical report, Ithaca, NY, USA, 1987.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Reidl. Item-based collaborative filtering recommendation algorithms. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 285–295, New York, NY, USA, 2001. ACM. ISBN 1-58113-348-0.

Badrul M. Sarwar, Joseph A. Konstan, Al Borchers, Jon Herlocker, Brad Miller, and John Riedl. Using filtering agents to improve prediction quality in the grouplens research collaborative filtering system. In *CSCW '98: Proceedings of the 1998 ACM conference on Computer supported cooperative work*, pages 345–354, New York, NY, USA, 1998. ACM. ISBN 1-58113-009-0.

Rossano Schifanella, André Panisson, Cristina Gena, and Giancarlo Ruffo. Mobhinter: epidemic collaborative filtering and self-organization in mobile ad-hoc networks. In *RecSys '08: Proceedings of the 2008 ACM conference on Recommender systems*, pages 27–34, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-093-7.

Albrecht Schmidt, Michael Beigl, and H. Hans-W. There is more to context than location. *Computers and Graphics*, 23(6):893–901, 1999.

Karl-Michael Schneider. Learning to filter junk E-mail from positive and unlabeled examples. In Keh-Yih Su, Junichi Tsujii, Jong-Hyeok Lee, and Oi Kwong, editors, *Natural Language Processing IJCNLP 2004*, volume 3248 of *Lecture Notes in Computer Science*, pages 426–435. Springer Berlin / Heidelberg, 2005. ISBN 978-3-540-24475-2.

U. Shardanand and P. Maes. Social information filtering:algorithms for automating word of mouth. In *Proc. Conf. Human Factors in Computing Systems*, pages 210–217, 1995.

Ryan Singel. Netflix cancels recommendation contest after privacy lawsuit. `http://www.wired.com/threatlevel/2010/03/netflix-cancels-contest/`, March 2010. Retrieved on Wednesday 16th February 2011.

L Story and B Stone. Facebook retreats on online tracking. *The New York Times*, 2007.

Latanya Sweeney. k-anonymity: a model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10:557–570, October 2002. ISSN 0218-4885.

Amund Tveit. Peer-to-peer based recommendations for mobile commerce. In *WMC '01: Proceedings of the 1st international workshop on Mobile commerce*, pages 26–29, New York, NY, USA, 2001. ACM. ISBN 1-58113-376-6.

Mark van Setten, Stanislav Pokraev, and Johan Koolwaaij. Context-aware recommendations in the mobile tourist application COMPASS. In *Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 235–244. Springer Berlin / Heidelberg, 2004.

Vassilios S. Verykios, Elisa Bertino, Igor Nai Fovino, Loredana Parasiliti Provenza, Yucel Saygin, and Yannis Theodoridis. State-of-the-art in privacy preserving data mining. *SIGMOD Rec.*, 33:50–57, March 2004. ISSN 0163-5808.

Jun Wang, Johan Pouwelse, Jenneke Fokker, Arjen P. Vries, and Marcel J. Reinders. Personalization on a peer-to-peer television system. *Multimedia Tools Appl.*, 36(1-2): 89–113, 2008a. ISSN 1380-7501.

Jun Wang, Stephen Robertson, Arjen P. Vries, and Marcel J. Reinders. Probabilistic relevance ranking for collaborative filtering. *Inf. Retr.*, 11(6):477–497, 2008b. ISSN 1386-4564.

Wolfgang Woerndl and Georg Groh. Utilizing physical and social context to improve recommender systems. In *WI-IATW '07: Proceedings of the 2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology - Workshops*, pages 123–128, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-3028-1.

Po-Wah Yau and Allan Tomlinson. Towards privacy in a context-aware social network based recommendation system. In *SocialCom 2011, 3rd IEEE International Conference on Social Computing*, Boston, October 2011. IEEE Computer Society.

Xiaojin Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.